КАК СТАТЬ ПРОГРАММИСТОМ

С НУЛЯ

после 30
без опыта
без образования

Борис ProIt

Борис ProIt

КАК СТАТЬ ПРОГРАММИСТОМ С НУЛЯ

ISBN 978-5-6046301-4-3

Москва, 2022

Выражаю свою благодарность моей семье, которая всегда поддерживает меня и верит в меня. Моей жене, сыну и родителям, без которых у меня ничего не получилось бы.

Также благодарю каждого из своих подписчиков — за обратную связь, за мотивацию и время, уделенное моим материалам. Я верю, что каждый из вас может достичь своей цели, если будет к ней идти.

И отдельно хочу выразить благодарность моей верной собаке, которая прошла со мной весь этот путь. Она была рядом в те моменты, когда я только начал изучать код, когда работал, набирался опыта и писал эту книгу.

Оглавление

От автора

Приветствую, дорогие читатели. Меня зовут Борис, по специальности я Senior Software Engineer (старший разработчик) на платформе Android. Проще говоря, я разрабатываю мобильные приложения и этим зарабатываю на жизнь. В своих соцсетях я активно помогаю ребятам, таким же, как я, которые хотят войти в IT и стать программистами, но не имеют опыта и технического образования. Кроме того, я создал бесплатный курс по программированию, который простым и доступным языком вводит человека в мир кода и логики. Об этом курсе я расскажу чуть позже.

Многие думают, что для работы в этой области нужно обладать рядом навыков и получить специальное образование. Но я не учился этому в университете, не программировал самостоятельно в детстве и в целом не обнаруживал в себе склонности к точным наукам: мои школьные отметки по математике всегда болтались где-то между двойкой и тройкой. Как вы можете заметить, я не подхожу под стандартное описание программиста — такого, каким его обычно себе представляют. Именно поэтому моя история может быть вам полезна и интересна. Ведь если вы тоже никогда не программировали, не имеете технического образования и всегда больше любили гуманитарные предметы, но очень хотите войти в IT и получить все бонусы этой профессии, то эта книга точно для вас.

На ее страницах я последовательно расскажу о каждом пройденном мной этапе, начиная с момента, когда в голове только зародилась идея стать программистом, и до того, как я устроился на первую работу. Я с удовольствием поделюсь своими личными техниками и секретами, которые позволят вам в несколько раз увеличить шансы дойти до своего первого заработка.

Дочитав эту книгу до конца, вы узнаете:

- о том, как я выбирал специальность;
- какие ошибки я совершал, выбирая курсы по программированию, и как сделать выбор, чтобы не потерять время и деньги;
- как я справился со всеми страхами и неуверенностью, ведь психологический аспект является одним из важных условий для достижения итоговой цели;
- как, работая на двух работах и будучи семейным человеком с маленьким ребенком, я оптимизировал свое время, чтобы успевать работать и учиться, одновременно уделяя внимание семье.

Сейчас перед вами стандартная версия книги, но в перспективе планируется выпустить и дополненный, расширенный вариант, который, помимо всего вышеперечисленного, будет включать в себя следующие главы:

- углубленный анализ основных направлений в программировании;
- описание моего подхода к выбору специальности, основанного на личном опыте;
- рассказ об опыте работы программистом, где в первой части главы я поделюсь советами о том, как легче и быстрее найти работу человеку без технического опыта и образования, а во второй части расскажу о тех моментах, с которыми столкнется новичок в первые недели;
- главу о поиске и всех тонкостях работы в иностранных компаниях. Расскажу про то, когда уже можно будет начинать искать работу, в деталях опишу, как происходят ее поиск и процесс обсуждения с работодателем.

Сейчас, оборачиваясь назад, я могу четко утверждать, что решение стать программистом стало одним из самых важных и правильных в моей жизни. Этот путь был долгим и сложным, но он однозначно стоит того, чтобы его пройти. Почему? Потому что, овладев навыком программирования, ты становишься востребованным во всем мире. Ты можешь жить практически в любой стране. На тебя уже не действуют ограничения и сложности местного рынка труда, ведь ты можешь работать на любую компанию в любой точке земли. Я бы назвал это невероятной свободой. Это свобода выбора, которая дает огромные возможности вне зависимости от города, где ты родился, от социального статуса и языка, на котором ты разговариваешь.

Помимо этого, программирование дает стабильность, ведь востребованность в квалифицированных специалистах постоянно растет, а это значит, что вы всегда сможете заработать. Программисты действительно неплохо зарабатывают, получая возможность жить на достаточно высоком уровне. Вы сможете спокойно спать по ночам, не переживая о кризисах или о том, что может случиться с компанией, где вы работаете. Стабильность и свобода — вот те награды, которые я получил в конце этого пути.

Когда я учился программированию, меня всегда возмущало, насколько сложно преподносится информация на курсах, в книгах и в обучающих видео. Я раз за разом пересматривал эти видео, где учитель с невозмутимым видом каким-то совершенно космическим языком объяснял очевидные для *него* вещи. Потом он писал код, запускал на компьютере, и у него все работало, а у меня постоянно выскакивала какая-нибудь ошибка. При этом в заголовке видео было четко указано, что оно предназначено, цитирую, «*для новичков*». Если вы уже пробовали изучать программирование, то это чувство собственной беспомощности вам наверняка знакомо. Я угадал? Так вот, цель моей книги в том, чтобы помочь вам сориентироваться в этих джунглях технической терминологии и в

бесконечных лабиринтах курсов по программированию. Сейчас я уже могу с уверенностью констатировать, что никакого волшебства здесь нет. Программирование — это обычный навык, которому может научиться абсолютно любой человек без каких-либо специальных знаний и опыта. И моя задача заключается в том, чтобы открыть для вас волшебную коробку, в которой лежит чудо-инструмент под названием «программирование», достать его и показать, что это обычный молоток, с помощью которого «строят» программы.

Вот почему, дочитав эту книгу до конца и изучив мой личный опыт, вы сможете взглянуть на этот процесс совершенно по-новому. Он уже не будет казаться вам чем-то страшным и непонятным. Вместо этого вы сможете с ясной головой оценить, где вы сейчас находитесь и куда двигаться дальше. Мой опыт, советы и рекомендации дадут вам возможность спокойно пойти к намеченной цели, причем самым коротким путем, ведь длинный путь я уже прошел за вас. Какой смысл его повторять.

Тем не менее, уважаемые читатели, я бы хотел сразу предупредить, что в руках у вас не волшебная палочка, которая одним махом сделает из вас высококлассных специалистов. Моя книга — это просто карта, которая позволит вам понять, откуда и куда нужно идти, как двигаться быстрее, как не свернуть на полпути. Но идти вы должны будете самостоятельно. И я обязан сразу предупредить, что до результата дойдут не все. Это нормально. Путь программиста не усыпан розами, это не легкая прогулка, а скорее долгий марафон по бездорожью под палящим солнцем. Так что если вдруг, начав обучение, вы увидите, что это все-таки не ваше, что совсем нет времени или желания продолжать, то прошу об одном: не мучайте себя. Жизнь дана нам для того, чтобы наслаждаться. Делайте так, как комфортно вам. Возможно, у вас сейчас просто нет нужных

ресурсов, но они появятся позже. Или вам следует поискать себя в чем-то другом.

Тем же, кто твердо намерен дойти до конца, несмотря на все трудности, я желаю удачи. И помните, что вы не одни в этом путешествии.

Итак, поехали!)

Кто такой программист на самом деле

Когда мы пытаемся ответить на вопрос, кто такой программист, в голове чаще всего возникает образ молодого крутого парня из фильмов, который сидит перед шестью мониторами с миллионами бегущих по ним строк и с энтузиазмом без остановки стучит по клавиатуре, выдавая десятки строк готового кода в минуту. Разумеется, все это без проблем запускается и работает.

Но в реальной жизни все далеко не так. Программирование — это не только создание кода. Большую часть времени программист думает о том, какие команды написать, как что-то починить или оптимизировать. Ключевое слово здесь — «думает». Действительно, очень часто бывает такое, что программист за несколько часов не пишет ни одной строчки кода, а вместо этого ковыряется в документации и ищет варианты решения проблемы на сторонних ресурсах. Бывает, что работа уже заканчивается, а ты все еще размышляешь о коде и пытаешься придумать решение задачи. Звучит не так романтично, как показывают в фильмах, правда? Но это действительно ближе к истине. Как минимум половину времени, которое вы проводите за компьютером, вы будете тратить на починку поломок и поиск причин, по которым ваша программа не запускается. Примите это за аксиому: у всех и всегда код будет вести себя так, как хочется ему, а вы будете сидеть и исправлять это, независимо от того, опытный вы программист или начинающий. С ростом вашей экспертности будет расти и сложность задач, но принцип останется тем же. Поэтому на начальном этапе вам придется свыкнуться с тем, что программа работает не так, как на экране преподавателя. И когда что-то сломается и вы начнете это

исправлять, вы уже можете себя поздравить: по сути, вы занимаетесь тем, чем занимается настоящий программист — ковыряетесь в коде, чините поломки, улучшаете и оптимизируете то, что делалось до вас.

Почему я так заостряю внимание на этом моменте? Потому что именно от этого я в начале испытывал настолько сильный стресс, что мне хотелось все бросить. Очень многие из-за подобного стресса не доходят до конца обучения, так как реальность не совпадает с ожиданиями. Ваша главная задача — не научиться быстро писать новый код, а доводить до совершенства уже имеющийся и быстро решать постоянно возникающие проблемы. Вот кто такой программист на самом деле: человек, который решает проблемы.

Но давайте не будем забегать вперед. Расскажу по порядку, как я проходил весь этот путь и с какими трудностями встречался.

Как все начиналось

Я помню этот день. На дворе стоял промозглый ноябрь. Я прозябал на стандартной работе, где в основном приходилось работать руками и выполнять задачи, поставленные руководством. На пути к одному из клиентов я заехал на заправку, заплатил за бензин и мимоходом заглянул в свой кошелек. Там лежала всего одна купюра небольшого номинала. На эти деньги можно было купить домой немного продуктов, которых хватило бы на пару дней.

Так было не всегда. Когда-то у меня имелась хорошая должность, были подчиненные и работа, которая хорошо обеспечивала. Но жизнь — это такая штука, когда все может измениться, и придется начинать все заново, с нуля. Поэтому в тот момент, стоя и глядя на свой пустой кошелек, я понимал, что что-то идет не так и нужно опять пробиваться наверх. Но как это сделать, я еще не осознавал.

Вздохнув, я продолжил свой путь к клиенту. Подъехав на место, я на какое-то время задержался у входа в его дом. Это был очень аккуратный особняк, но не такой огромный, в которых живут голливудские звезды, а обычный, стильный и привлекательный. Такие обычно показывают в западных фильмах.

Меня приветливо встретил хозяин, на вид обычный парень в очках. Войдя внутрь, я оценил обстановку: большие окна, пропускающие много света, лаконичный интерьер. В просторном зале перед диваном располагался камин, на ковре перед которым, раскладывая свои игрушки, сидели две маленькие дочки хозяина дома — к моменту, как я приехал, отец уже забрал их из школы. Я подумал, что ему повезло: когда все сидят в офисах, он уже дома и может побыть со своими детьми.

Я стоял и внимательно впитывал в себя каждую увиденную деталь. Я не то чтобы завидовал тому парню, своему клиенту, но уютная обстановка его просторного дома произвела на меня очень сильное впечатление. Вернувшись на работу, я спросил у своего руководителя, не знает ли он, чем занимается тот парень, чтобы иметь такой особняк и при этом возвращаться с работы так рано. «Да какой-то там программист. Не знаю, в IT, в общем!» — ответил мне руководитель.

Возвращаясь в тот день домой, я не мог избавиться от мыслей об увиденном. Снова и снова я прокручивал в голове те эмоции и ощущения, которые испытал, находясь в доме своего клиента. Я понял, как сильно хочу, чтобы моя семья тоже жила в таком уюте, а я имел бы возможность находиться дома в течение дня и чаще видеть своих родных, а не приходить в девять вечера, чтобы полчаса пообщаться с сыном перед тем, как он отправится спать.

Весь вечер я думал о том, что мне нужно сменить профессию. Это было похоже на ощущение, будто ты несешься по туннелю. Будто все внимание только на конечной цели, а то, что происходит вокруг, совершенно не важно. Так же было и у меня. Вечер в семейном кругу в тот день пролетел мимо меня, ведь в голове была только мысль: *я хочу начать этот путь*.

Я очень хорошо помню этот момент. Я сидел в темной комнате, слегка освещаемой светом от экрана монитора. Было уже поздно, все спали, а я копался в интернете и искал истории людей, которые смогли добиться успеха в сфере программирования. Я ежесекундно задавал себе вопрос: смогу ли я дойти до конца? Ведь мне уже далеко не двадцать лет, есть семья и обязательства по отношению к ней, а основной профессиональный опыт сосредоточен совершенно в другой сфере. Я не имел технического образования и понимания,

как это все работает. В голове витали бесконечные сомнения и страхи.

И в этот момент я сказал себе, что пусть лучше я попробую и у меня не получится, чем не попробую и буду думать: «А вдруг получилось бы?»

Так я и сделал. В тот же вечер я купил свой первый курс. То есть на раздумья у меня ушло всего *несколько часов*.

Но давайте остановимся на этом моменте немного подробнее. Ведь не все принимают решения таким образом. Я и сам иногда хожу в раздумьях некоторое время, перед тем как начать действовать. Скажу больше: пообщавшись с моими подписчиками и теми, кого знаю лично, я вижу, что огромное количество людей, решившись изменить свою жизнь и войти в IT, остаются именно на этом этапе и долгое время пребывают в подвешенном состоянии. Они не начинают идти к своей цели, но в то же время и не отказываются от нее.

Думаю, причина этому — страхи и сомнения, в частности, страх больших изменений и многие другие, о которых мы еще поговорим в этой книге.

Страхи — неотъемлемая часть любого пути. Уже будучи состоявшимся программистом и общаясь со своими подписчиками, я понял, что, хотя все люди и разные, их страхи часто совпадают. Многое из того, о чем рассказывали мне подписчики, я проходил и сам. Это стало для меня открытием.

После проведения опросов у себя на странице в социальных сетях, а также бесед в личных сообщениях я смог составить список распространенных страхов, которые испытывает человек при входе в новую профессию. Благодаря этим выводам я смог ответить себе

на вопросы, почему один человек в сорок лет может получить новую профессию, а другой нет, и в чем разница между этими людьми.

Давайте рассмотрим ситуацию на примере курсов по программированию. Она показывает, насколько неправильно мы в целом подходим к процессу обучения с помощью курсов. Это выглядит следующим образом: мы записываемся на курсы и ждем результатов. Основной упор в обучении идет на написание кода. И со временем народу в группе остается все меньше и меньше. В итоге до конца доходят не все. К этому принято относиться как к данности, никто не задается вопросом, почему так происходит. Но я все-таки не раз задумывался: почему одни доходят до результата, а другие нет? Чем же первые отличаются от вторых?

Я сам дошел до конца и убежден в том, что умение писать код — это лишь треть успеха. Помимо этого, нужно уметь учиться и, конечно же, справляться со своими страхами и сомнениями, не забывая работать над собственной мотивацией. Я помню, как развивался в каждом из этих аспектов, и могу точно сказать: без проработки имеющихся сомнений дойти до конца будет очень сложно. Потому что на этом долгом пути мне потребовались ясный ум и психологическое равновесие. Без этих двух составляющих я не смог бы изучать сложный материал на протяжении длительного времени.

Вот некоторые из страхов, с которыми я сталкивался:

- страх изменений в жизни и потери стабильности;
- страх, что материал окажется слишком сложным для изучения;
- страх оказаться в чем-то хуже других;
- страх того, что для работы не хватит технического образования и опыта;
- страх того, что ожидания не оправдаются.

Как бы странно это ни звучало, но избавиться от этих опасений мне во многом помог еще один страх, который перевешивает все вышеперечисленные. Это страх испытать чувство сожаления. Больше всего я боюсь угрызений совести насчет того, что я мог попробовать и не попробовал, что я мог дойти до конца и не дошел. Поэтому когда ко мне в голову приходят мысли, что я не смогу или мои ожидания будут не оправданы, я говорю себе: «Попробую, а там разберусь. Кто-то уже достиг этого, значит, есть инструкция, как это сделать». Поэтому я просто пробую, для того, чтобы избавиться от этого противного внутреннего чувства угрызений совести.

Когда я думаю об этом, в моей голове возникает картинка в виде айсберга (см. рис. 1). Ведь, по сути, мой процесс обучения состоял из нескольких уровней. Видимая и очевидная часть — это изучение кода. Но были и невидимые части — страхи и умение учиться, о которых тоже не нужно забывать. Основная мысль, на которой я хочу сконцентрировать ваше внимание, заключается в следующем: для того чтобы успешно и быстрее изучать код, необходимо отработать собственные страхи, а также оптимизировать и прокачать навык обучения. При таком подходе процесс изучения кода станет намного легче и быстрее.

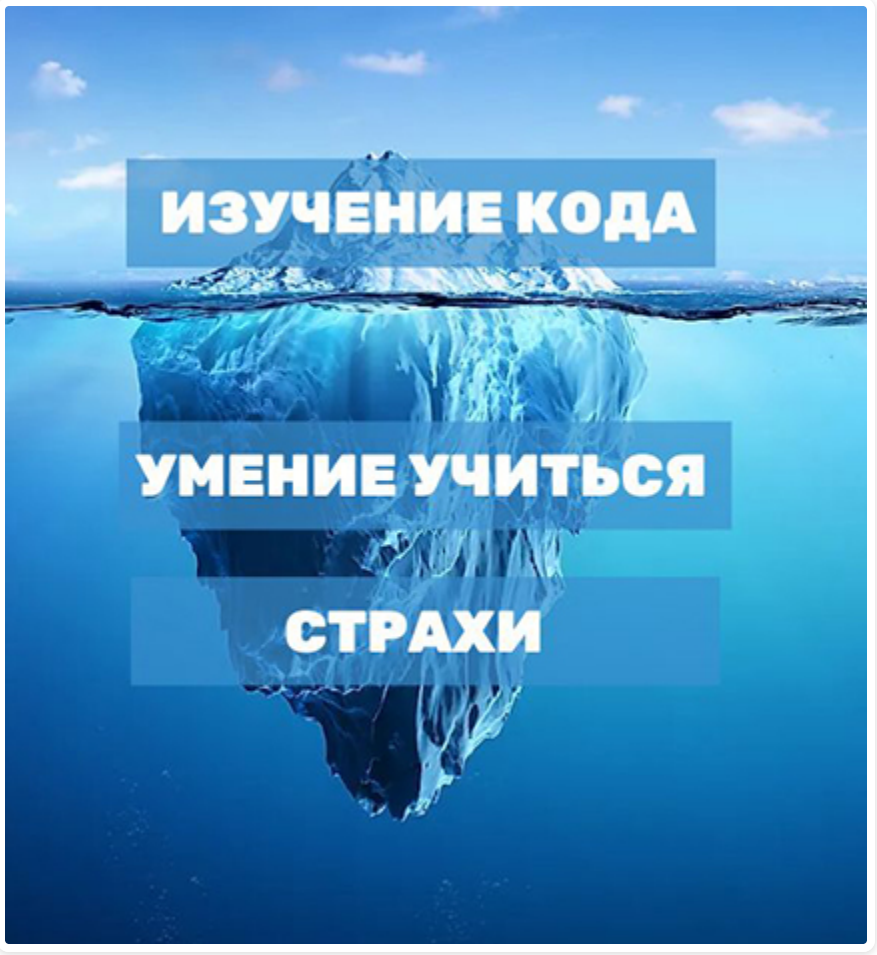


Рис. 1. Процесс обучения и страхи

А пока давайте вернемся в тот судьбоносный вечер, когда я, сидя в темной комнате, принимал одно из самых важных в моей жизни решений. В голове все переворачивалось. Было очень много вопросов и очень мало ответов. Я сидел и искал в интернете разную информацию, чтобы сделать верный выбор. В тот момент у меня уже не было сомнений, что я сегодня начну свой путь программиста. Но я хотел начать его правильно и допустить минимальное количество ошибок. На ряд своих вопросов я мог найти ответ в интернете. Но были и такие факторы, с которыми всемирная паутина не могла мне помочь. Давайте более детально разберемся, как я преодолевал свои страхи и психологические барьеры.

Первым страхом, с которым я столкнулся, был страх больших изменений.

Страх больших изменений

На своем жизненном пути я принимал много разных решений, но интересно здесь не их количество, а то, что я их принимал, как правило, очень быстро. Так же было и с решением стать программистом. Еще в три часа дня у меня и мысли не было о чем-то подобном, в семь вечера, по дороге домой от клиента, я подумал об этом, а уже в одиннадцать часов я сидел и осваивал свой первый курс по Android.

Для меня такое поведение всегда казалось нормой, но оказалось, что так происходит далеко не у всех.

Как-то при общении с моим близким другом он сказал мне, что не может так быстро принимать решения. Я задумался над тем, почему так происходит. После некоторых размышлений я понял: вся причина в том, что я не осознаю весь масштаб изменений, к которым приведет мое решение, и не заглядываю вперед.

То есть, начиная что-то новое, я не вижу всей длины этого пути и тем самым не оцениваю масштаб последствий сделанного выбора. Обычно я даю себе слово просто попробовать что-то сделать. Это затягивается на какое-то время, а потом бросать уже жалко.

Насколько эффективен такой подход? Я не знаю. Я столько всего начинал и бросал, что уже и не вспомнить. Но, несмотря на это, я все равно считаю, что в таком подходе больше хорошего, чем плохого. Потому что любой провал дает бесценный опыт, он учит лучше любых тренингов и курсов. Мысль «зато я попробовал» придает нам сил и приносит внутреннее чувство удовлетворения. Это прямо противоположно тому ощущению, когда мы не решились

осуществить что-то, а потом каждую ночь перед сном терзаемся вопросом: «А что, если бы я попробовал?»

Так вот, принимая решение, я просто выбираю наименьшую боль из двух. Как это работает? Если при появлении какого-то желания я не решаюсь действовать, то запускается процесс раздумывания. И этот процесс я ненавижу еще больше, чем угрызения совести от вопроса «А что, если бы я попробовал?». Поэтому когда в моей голове возникает идея или цель, я не люблю долго вынашивать ее в себе. Я либо отказываюсь от нее и живу дальше, либо начинаю реализовывать.

Меня раздражает это состояние подвешенности, когда изо дня в день ты носишь мысль у себя в голове и откладываешь принятие решения. Потому что в действительности состояния «посередине» нет, это не более чем ловушка для мозга. Если я не выбираю *делать*, то, пока я думаю, я фактически выбираю *не делать*. Переключатель работает только в две стороны: либо да, либо нет. И если я говорю какой-либо идее «Нет», то я просто осознанно отказываюсь от нее и живу дальше.

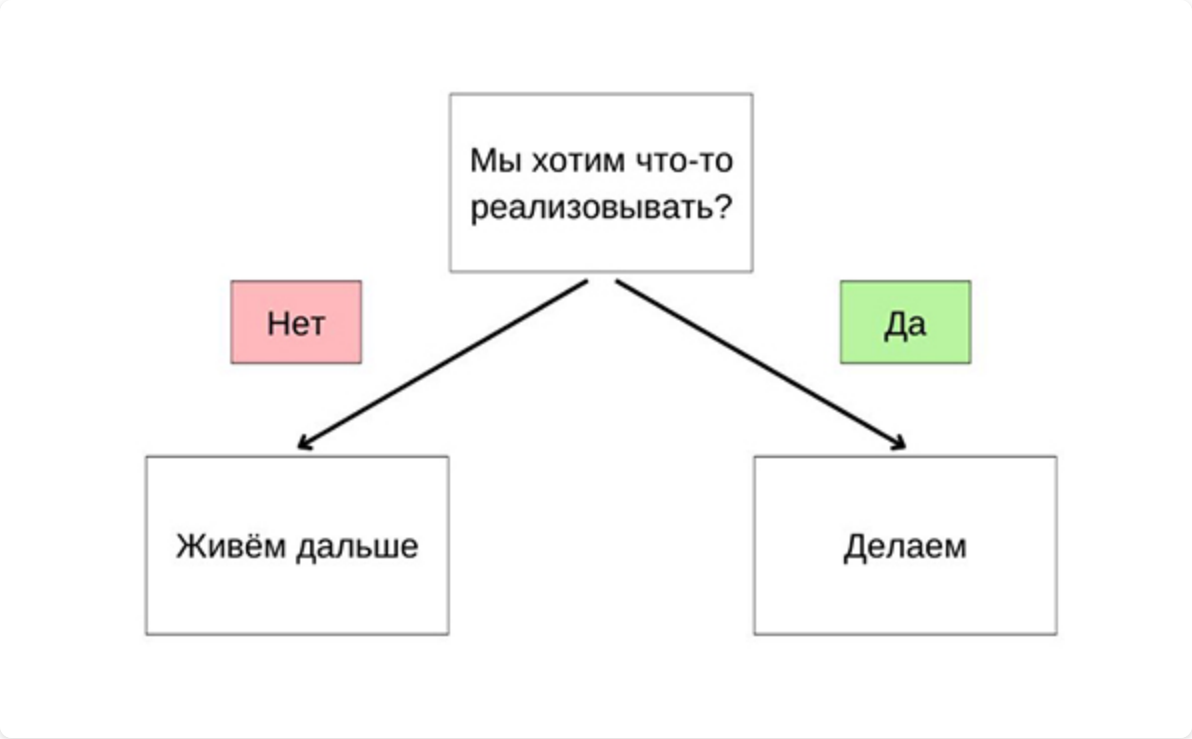


Рис. 2. Схема принятия решения

Такой подход кажется жестким: всегда существует риск случайно отказаться от того, от чего отказываться не стоило. Не всегда мы имеем возможность принять серьезное решение быстро, иногда требуется дополнительная информация. В таком случае работает все та же схема, что представлена на рисунке 2, но в положительном сценарии «Я попробую» появляется дополнительный этап, который я называю «Сбор информации». Цель этого этапа — собрать как можно больше данных для принятия взвешенного решения.

Например, я, делая выбор перед тем, как принять решение, задавал себе следующие вопросы:

- Будет ли мне хватать времени? Чтобы понять это, я должен поговорить со своей семьей и убедиться, что они смогут на какой-то период освободить меня от некоторых домашних обязанностей, пока я посвящаю себя обучению.

- Не очень ли сильно мое обучение ударит по семейному бюджету? Есть ли в наличии денежные ресурсы для покупки курсов и ноутбука?
- Есть ли у меня вообще шансы войти в эту сферу на том этапе жизни, на котором я нахожусь? Все-таки я уже не студент, и мне уже не восемнадцать лет. Хватит ли у меня сил на такой путь?

После сбора информации картина станет ясна и принять решение будет намного легче. И даже если я решу не начинать этот путь, меня уже не будет мучить внутренний голос, потому что я буду знать объективные причины, по которым я отказался от своих намерений.

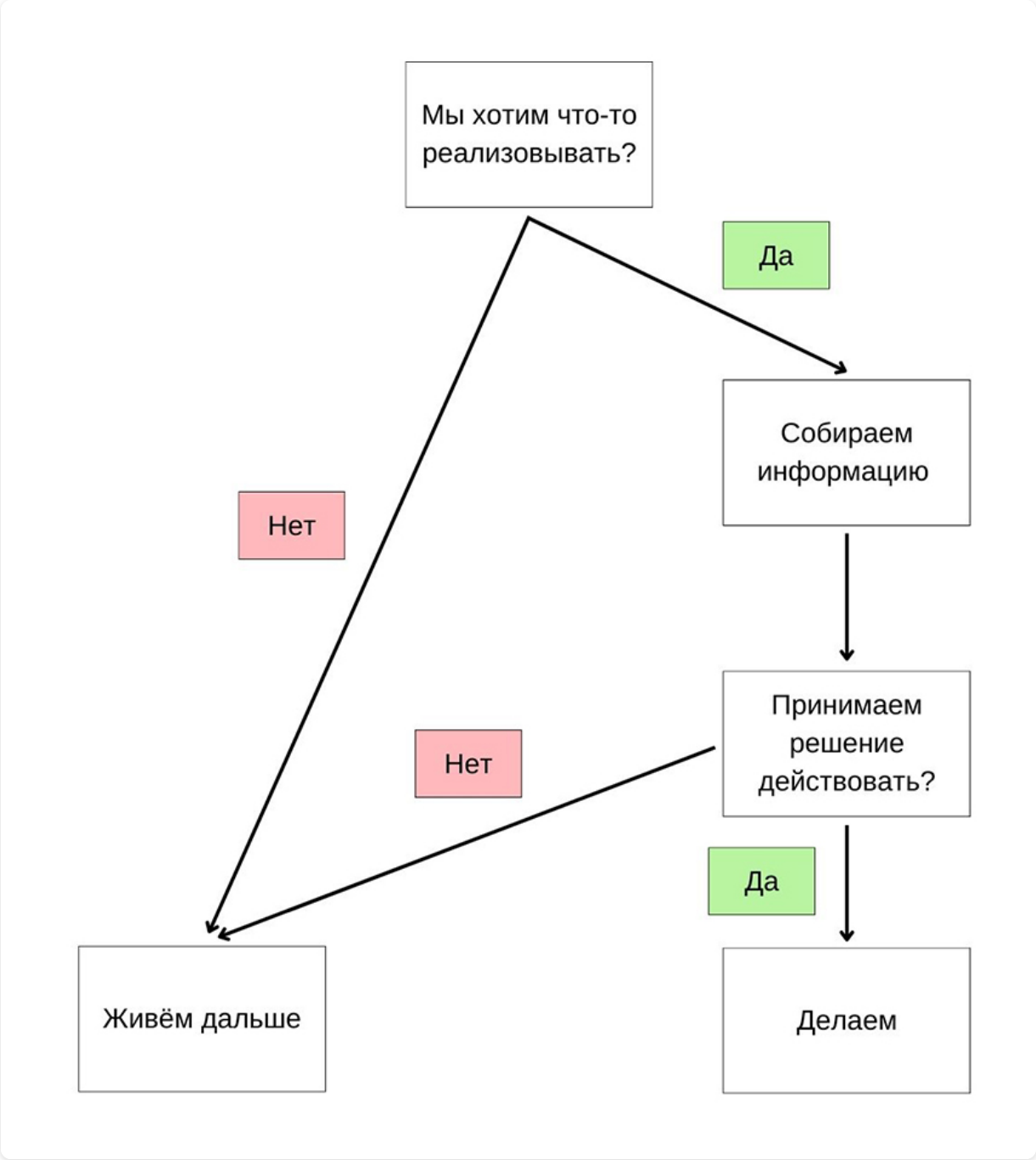


Рис. 3. Усовершенствованная схема принятия решения

При таком подходе страх перед большими изменениями не проявляет себя так ярко (во всяком случае, так было у меня). Ведь он обычно появляется тогда, когда мы находимся в подвешенном состоянии, в так называемом статусе «Я подумаю». А когда мы

осознанно делаем выбор — либо да, либо нет, — мы четко понимаем, в каком направлении мы двигаемся, а главное, четко определяем свое место в этом процессе.

Любая неизвестность внушает страх и неуверенность в своих силах, поэтому, едва мы избавляемся от неизвестности и наполняемся уверенностью в себе и решимостью сделать первый шаг, страх отступает, и возникает четкий, понятный план действий.

Вообще, если так задуматься, каждый день мы многократно делаем выбор, часто даже не осознавая этого. Но важно помнить: если мы решаем подумать, вместо того чтобы начать действовать, фактически мы принимаем решение ничего не делать. Никакого «посередине» не существует. Помня об этом, я предпочитаю принимать решения быстро и не оставлять себе времени на раздумья. Да, эти решения не всегда оказываются верными. Зато по ночам меня не мучают мысли и вопросы «А что, если бы я все-таки попробовал?».

Ну хорошо, скажете вы, и все-таки, почему именно программирование? В мире есть много других профессий, где можно самореализоваться и добиться финансовой независимости.

Думаю, пришло время пояснить, почему я остановился именно на этой специализации.

Почему люди меняют профессию и становятся программистами

До того как сменить профессию, я в основном работал на должностях, где нужно было взаимодействовать с людьми. В роли менеджера мне приходилось нанимать персонал и управлять им. Также я имел большой опыт в продажах, и это тоже подразумевало общение с людьми, только теперь не с подчиненными, а с потенциальными покупателями. Я уже говорил, что был бесконечно далек от написания кода: моей образовательной специальностью был менеджмент, в школе по математике я имел максимум тройки, да и в университете был примерно такой же уровень. То есть, думаю, понятно, что склад ума у меня далеко не технический.

Так каковы же были мои личные мотивы на момент принятия решения о смене специальности? Я хотел иметь навык, который позволял бы:

- создавать какой-то продукт, чтобы результатом моего труда было нечто осязаемое;
- хорошо зарабатывать;
- быть востребованным во всем мире;
- и при всем при этом работать удаленно.

Но я — это я, а у любого из вас могут быть свои интересы и цели. Понимая это, я задумался о том, что вообще движет людьми, которые решаются на смену профессии. И обнаружил очень интересное исследование от компаний GeekBrains и ResearchMe[1].

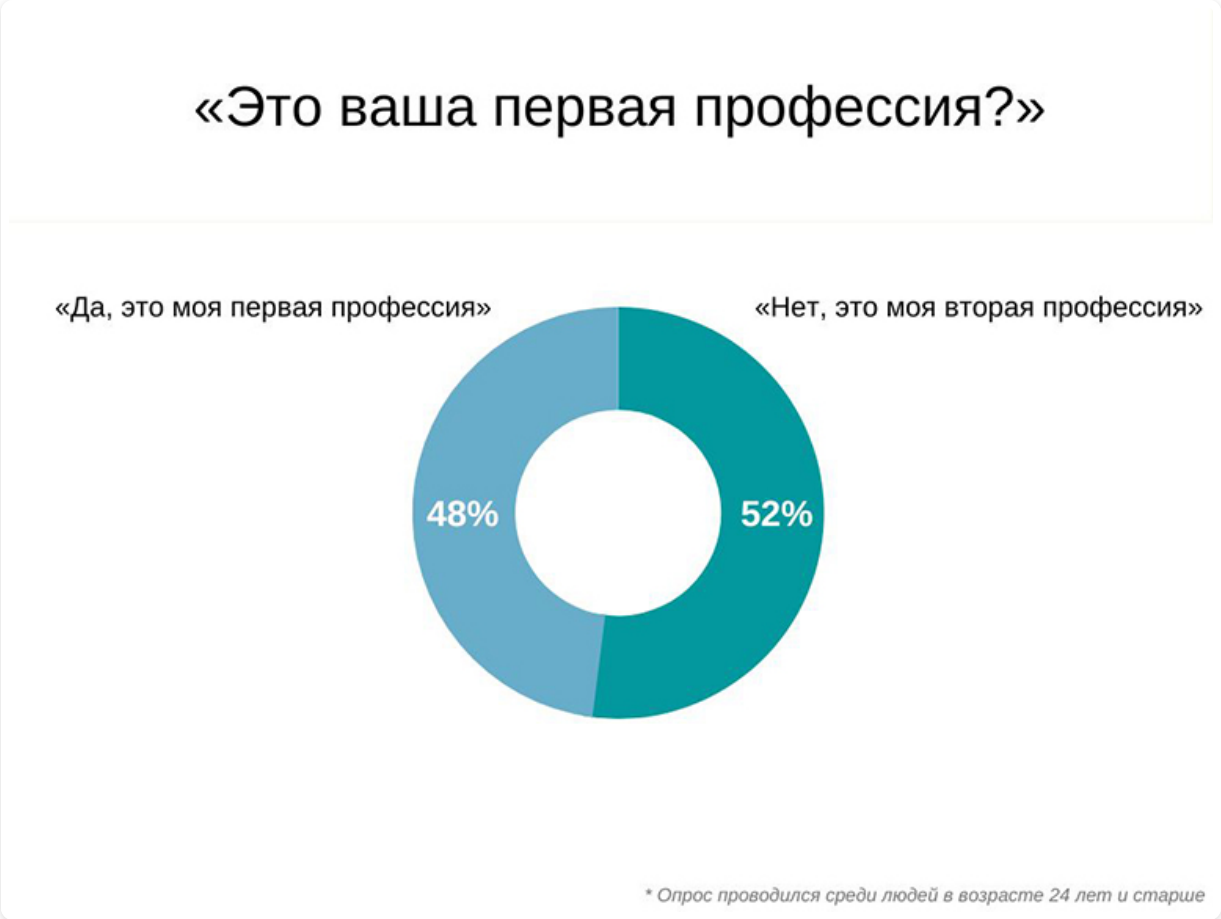


Рис. 4. Результаты опроса о смене профессии после 24 лет

В правой части диаграммы отображено количество людей, для которых IT было не первой сферой их работы. И таких людей оказалось целых 52%. Это означает, что почти половина людей, работающих в IT, минимум один раз меняли профессию. Поэтому если вам кажется, что смена профессии — это редкое явление, то данные исследования вполне успешно опровергают это мнение. Я и сам встречал достаточно много людей, сменивших направление работы и решивших уйти в IT-сферу. И это не обязательно написание кода, это может быть любое другое направление, например, web-дизайн, рекрутинг, управление проектами или продажа IT-продуктов. Но о специальностях мы поговорим чуть позже. Основная мысль, которую я хочу до вас донести, заключается в том, что огромное

количество людей пришло в IT из других сфер, и не все эти люди стремились стать именно программистами. Они становились специалистами в других направлениях.

Но что вообще сподвигло их решиться на важный шаг и поменять специализацию?

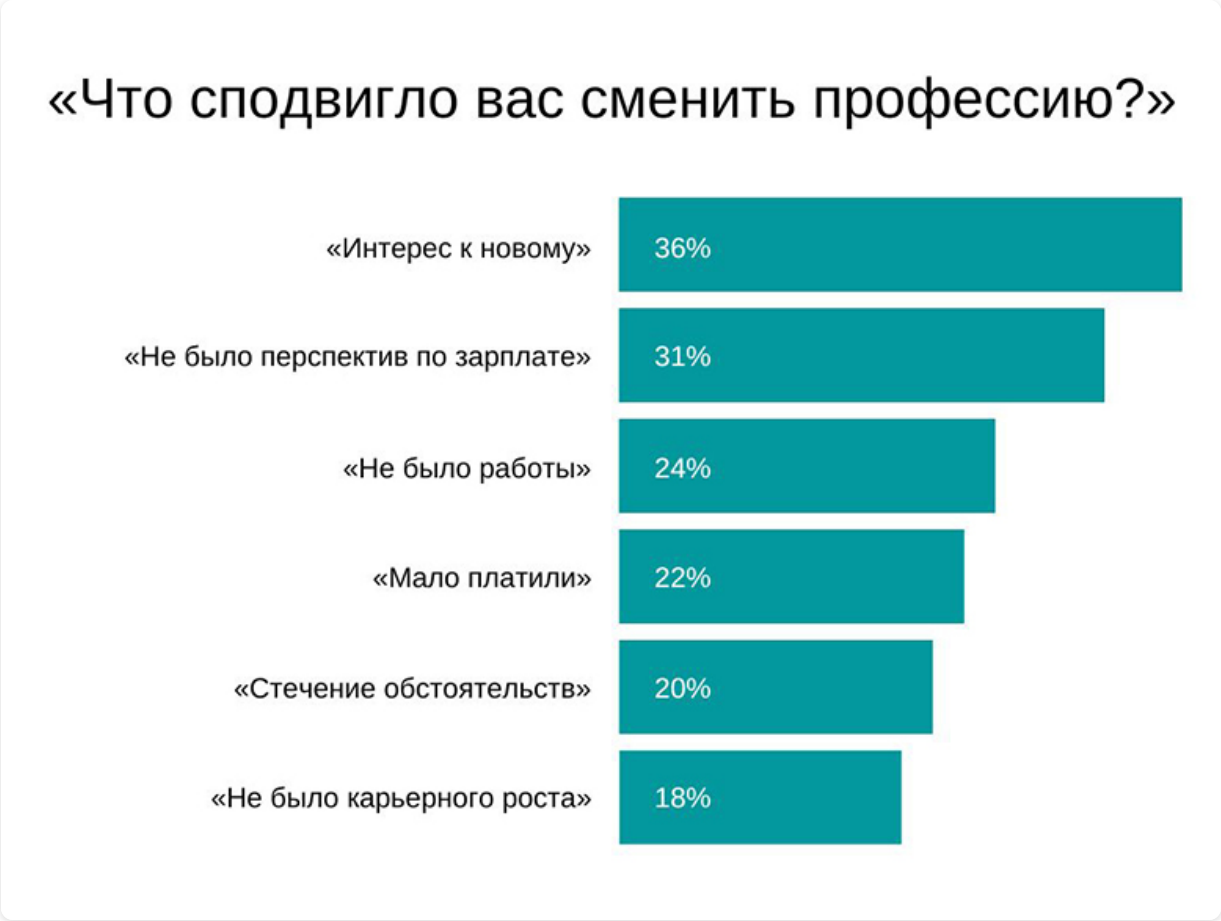


Рис. 5. Основные причины смены профессии

Если посмотреть на рис. 5, то мы увидим, что большинством людей при смене профессии двигало желание попробовать себя в чем-то доселе неизведанном, обрести новые навыки и новые возможности.

Вторая по популярности группа — это люди, желающие больше зарабатывать. В целом группы два, три и четыре объединяет то, что в

той специальности, где человек работал изначально, возникли какие-то проблемы. И именно эти проблемы сподвигли человека уйти в IT. Все остальные причины вы можете посмотреть на том же графике. В итоге мы можем увидеть, что большинство людей уходят в IT из-за проблем в текущей профессии или из простого любопытства.

Как я уже говорил ранее, не все становятся программистами. Я еще раз повторюсь, что в IT существует огромное количество специальностей. Поэтому места хватит всем. И если вы чувствуете, что у вас нет тяги к написанию кода или нет столько времени на обучение, то вы всегда можете выбрать другие, более доступные направления.

На рис. 6 приведены данные о том, в какие направления обычно уходят люди, решившие сменить профессию на IT. Я не буду вдаваться в детали и описывать каждое направление, но некоторыми мыслями хотелось бы поделиться. Во-первых, любопытно, что первые три строчки занимают направления, для работы в которых не нужно быть программистами. Профессия программиста занимает лишь четвертое место в данном списке. И во-вторых, благодаря этому списку можно убедиться: в IT настолько много различных специальностей, что каждый при желании сможет найти себе направление, которое можно освоить исходя из количества свободного времени и понимания, чем именно человек хочет заниматься и что ему действительно интересно.

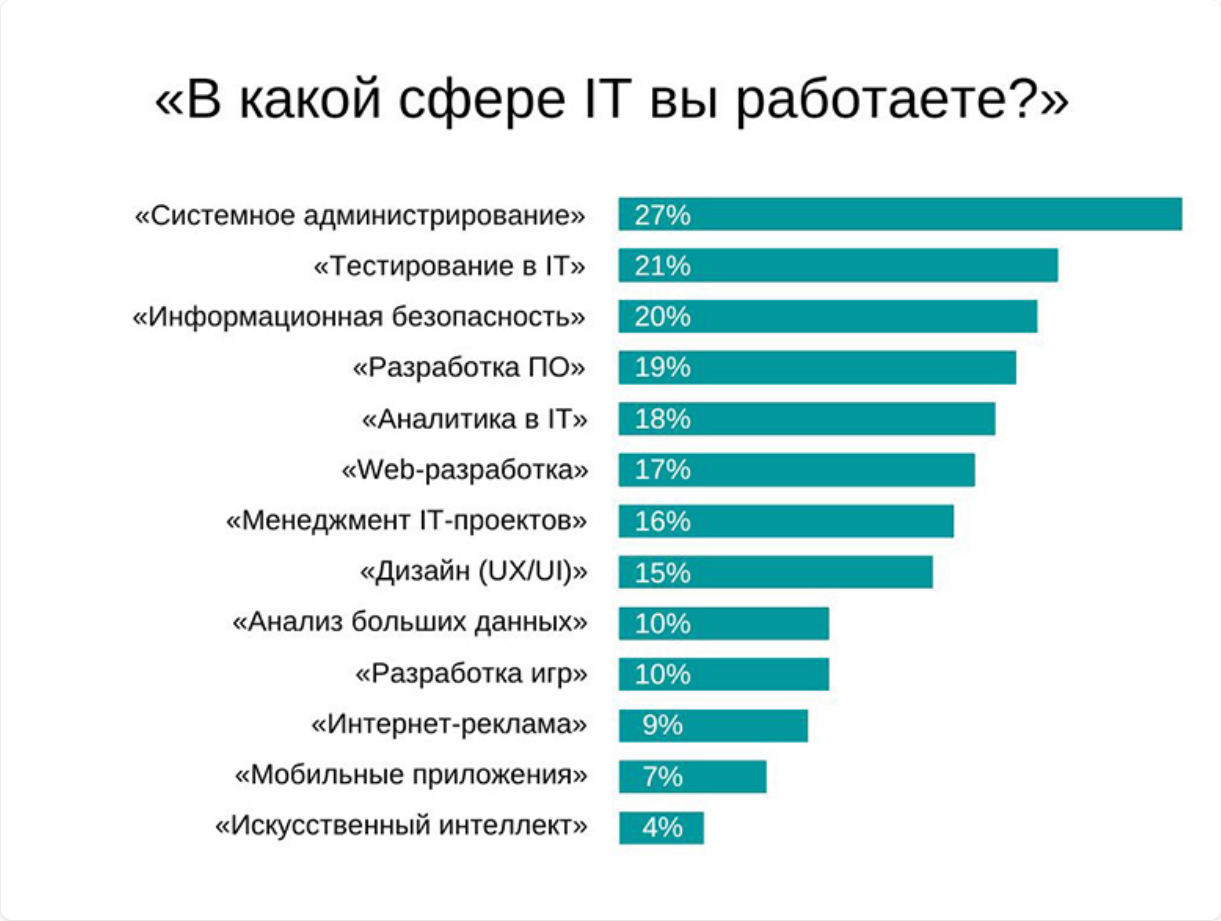


Рис. 6. Направления IT по популярности

Возвращаясь к исследованию от GeekBrains, одним из вопросов, которые задали людям, была просьба назвать чувства, которые они испытали при переходе в новую профессию (см. рис. 7). И я соглашусь со всеми ними.

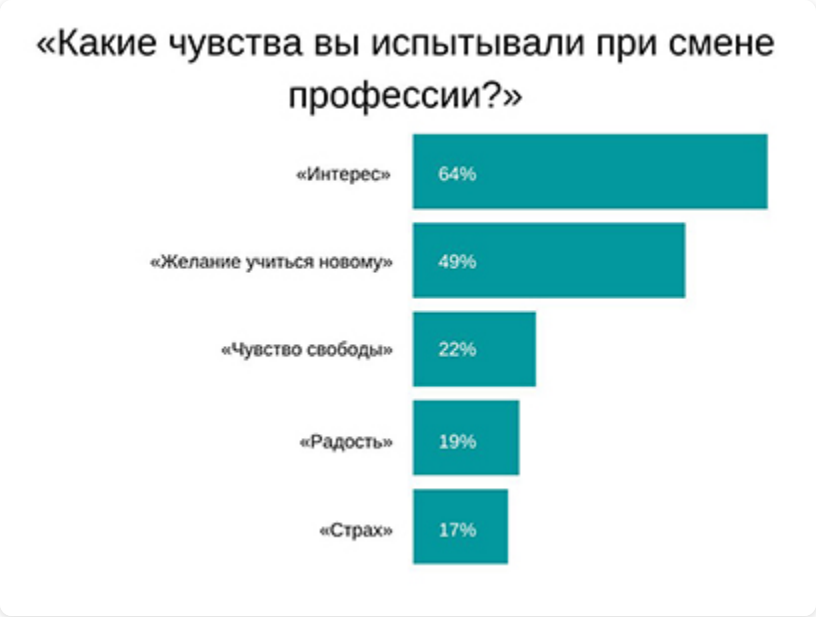


Рис. 7. Топ-5 чувств, которые испытывали люди при смене профессии

В этом списке я объединил бы чувство страха и интереса и поставил их на первое место. Потому что это безумно интересно — входить в мир профессиональной разработки, особенно если вы никогда не работали в IT. Это может вызвать море положительных эмоций. Но вместе с этим накрывает жуткий страх неизвестности и неуверенности в себе. Это еще называется синдромом самозванца. Признаюсь честно, будучи уже профессиональным разработчиком, преодолев большое количество проблем и решив кучу сложных задач, я все равно испытываю это чувство. Этому явлению я посвятил целый раздел, который мы в подробностях рассмотрим далее.

А пока, подводя краткий итог анализу этого исследования, я хочу еще раз подчеркнуть некоторые моменты: ситуация, когда человек уже в достаточно зрелом возрасте решается кардинально изменить свою жизнь и получить новую профессию, — далеко не редкость в современном мире. Более половины людей из сферы IT — это люди, которые раньше работали по другой специальности. Надеюсь, эта информация поможет вам принять правильное решение.

Для того чтобы войти в эту сферу, не обязательно иметь технический склад ума и учиться программировать, есть множество смежных направлений. Лично я выбрал для себя разработку мобильных приложений на Android. И, забегая вперед, скажу, что ни разу не пожалел о своем выборе. Почему я остановился именно на этом направлении? Расскажу подробнее.

Почему я выбрал Android

В тот вечер, когда я сидел в этой комнате с купленными курсами, я еще не осознавал того, что сделал один из самых важных первых шагов в своей жизни. Но почему я остановился именно на этом направлении? Ведь в списке доступных курсов были web-разработка, разработка на iOS и многое другое.

Когда-то понятие «программирование» ассоциировалось у меня с чем-то вроде волшебства. Программисты казались мне отдельной кастой людей, у которых есть огромное количество возможностей. И я не мог даже представить, что когда-то тоже буду зарабатывать на жизнь написанием кода. Если я и допускал мимолетную мысль о том, что когда-нибудь стану разработчиком, то думал, что буду создавать web-сайты.

До того как стать программистом, я имел опыт управления собственным интернет-магазином и на уровне пользователя представлял, как он примерно устроен. Я управлял сайтом, создавал странички, вписывал заголовки и описания, умел добавлять изображения. Но честно говоря, понятия не имел, что происходит «под капотом». А мобильные приложения и вовсе были для меня чем-то далеким и непонятным. Я никогда не имел собственного мобильного приложения, поэтому, в отличие от web-сайтов, даже приблизительно не представлял, как все это работает.

Поэтому, когда возник вопрос о том, какое направление в программировании выбрать, я начал диалог с самим собой. Вот, казалось бы, сайты — самое логичное направление. Собственные интернет-странички сейчас нужны всем. Сложно представить даже маленькую компанию, которая будет работать без web-сайта.

В этот момент я вспомнил, что довольно давно, еще когда я работал в сфере менеджмента, мне нужно было сделать анализ цен на разработку сайтов. И я обнаружил, что предложений от компаний по разработке было очень много. При этом цены были невысокими. Затем я решил проверить расценки фрилансеров. Я зашел на одну из крупных фрилансерских платформ и создал заявку на разработку сайта. Мне сразу посыпались отклики с достаточно низкими ценами. Я удивился, что даже у фрилансеров с большим количеством отзывов и крутым портфолио цены все равно остаются на весьма невысоком уровне.

И тогда я случайно заметил чье-то задание на разработку мобильного приложения. Увидев ценник, я испытал шок. Цена была в разы больше, чем за web-разработку. А заявок от программистов, желающих взяться за заказ, было в разы меньше.

Сейчас, по прошествии времени, я уже понимаю, что на самом деле уровни разработки и там и там могут быть разными. Разработка сайта легко может стоить дороже разработки приложения, да и техническая его реализация может быть сложнее. Поэтому мое сравнение на тот момент было абсолютно необъективным. Но тогда я об этом не знал. Я просто увидел цену на одно и на другое и сделал вывод, что мобильная разработка дороже создания web-сайтов. Это отпечаталось в моей памяти и всплыло в голове в момент принятия решения.

Я продолжал проговаривать про себя варианты. Поиск в интернете показал, что при создании сайтов используется огромное количество технологий. Тогда я еще ничего не знал о языках программирования, но подозревал, что вариаций может быть много. И меня всегда мучал вопрос: зачем все так усложнять? Неужели нельзя создавать web-приложения, используя какой-то один язык программирования и один подход? Поэтому в размышлениях об изучении web-

разработки я пришел к выводу о том, что мне придется разбираться во всем этом многообразии вариаций и технологических комбинаций.

С другой стороны, я знал, что на iPhone используется платформа iOS, а на большинстве остальных гаджетов — платформа Android. Не понимая всех технических тонкостей и не разбираясь во всем этом, я все же увидел довольно ясную картину. Web-разработка — много и непонятно. Мобильная разработка — тоже ничего не понятно, но хотя бы нужно освоить одну технологию.

Поэтому, взвесив эти три пункта: «спрос/конкуренция», «уровень заработка» и «объем информации для изучения», я принял решение, что хочу двигаться в направлении мобильной разработки.

Сначала я решил стать iOS-разработчиком. Но, как оказалось, для разработки приложений под iOS надо иметь макбук, которого у меня не было. В результате я понял, что выбора нет, и начал изучать Android.

Сейчас, поработав в этой сфере уже какое-то время, получив опыт Android-разработки, а также небольшой опыт взаимодействия с iOS и web, я могу сказать, что где-то в своих предположениях, анализе и подходах я допустил ошибки и был неправ. Но это нормально, ведь тогда у меня не было тех знаний, которые есть теперь. Возможно, обладая тогда этими знаниями, я бы более вдумчиво подошел к выбору направления и потратил бы на это больше времени. А тогда на это ушло всего несколько часов.

Но тем не менее я не жалею, что выбрал именно эту специализацию. Иногда я даже рад, что не сделал глубокий анализ Android-направления, потому что, быть может, тогда я бы вообще отказался от идеи изучать Android и в результате не был бы там, где я нахожусь.

Но в любом случае, хотя я и не делал глубокого анализа направлений и определился достаточно быстро, основываясь на логике, личном опыте и интуиции, я рекомендую вам потратить время и изучить вопрос более серьезно. Это позволит вам сделать правильный выбор в долгосрочной перспективе, ведь вы выбираете профессию на много лет вперед, а не новые ботинки, которые выбросите через год. Но что еще важнее, от направления зависит, насколько долго вам придется учиться. И Android-разработка — это не самый короткий путь, но он того стоит.

Я отдельно хочу оговориться, что не ставил своей задачей убедить вас, что web-направление — это что-то сложное и непонятное, что там мало платят и туда не стоит идти. Нет! Это очень классное направление, и оно однозначно стоит вашего внимания, но прежде чем туда идти, я рекомендую разобраться в деталях и тонкостях. Как я сказал выше, в своем анализе я допустил некоторые ошибки, поэтому сейчас, может быть, я бы и пошел в web, но уже с другим подходом и другими ожиданиями.

На одной из допущенных мной ошибок я хочу остановиться подробнее. Этому никто не учит на курсах, а знакомых программистов, которые посоветовали бы мне поступить иначе, у меня не было, поэтому я испытал больше стресса и потратил в итоге больше времени на достижение цели. Но вы, прочитав эту книгу, сможете подойти к вопросу более организованно и сэкономить время и нервы.

Что именно я сделал не так: как и многие другие, я начал свой путь с того, что зашел в поисковик и ввел «самый популярный язык программирования».

Почему я считаю это ошибкой? Представьте, что вы планируете стать врачом. Разве вы выбирали бы специальность только исходя из ее популярности? Нет, кто-то мечтает работать с детьми и стать

педиатром, кто-то хочет стать психиатром, а кто-то — хирургом. А все потому, что в момент выбора врачебного направления мы понимаем, какой врач чем занимается. А при выборе IT-специальности мы понятия не имеем, какая между ними разница. Поэтому первое, что нужно сделать, — это разобраться в каждом направлении, и уже потом выбирать, кем вы хотите стать. И после этого автоматически решится вопрос с выбором языка программирования. К сожалению, когда я выбирал свое направление, я не мог так рассуждать, но у вас есть шанс не повторить моих ошибок.

Вот мои рекомендации, которые могут помочь вам в выборе направления.

Сначала изучите мир IT в целом. Я уже обращал ваше внимание на то, что в IT работают не только программисты. Поэтому если вы твердо решили, что хотите именно писать код, то тогда вбивайте в поисковик «самые популярные направления в программировании». Если же вы еще не решили, хотите ли быть программистом или же просто начать карьеру в IT, в таком случае в поисковике нужно вбивать «направления в IT». Важный момент: даже если вы точно хотите стать программистом, изучить специальности в IT все равно будет полезно, потому что вам придется с этими людьми работать. Но это вы еще успеете сделать. Когда я уже начал учиться, такого рода информацию я изучал обычно за рулем, в дороге. Я просто включал обзор различных профессий и слушал, кто чем занимается. Это не техническая информация, для ее понимания не нужно находиться за компьютером. Но это в конечном итоге тоже способствовало расширению моих познаний об IT-сфере. Не жалейте на это времени. Я всегда говорю, что успех — это как пазл, и развитие системных знаний — один из важных его кусочков.

Когда направление будет выбрано, я рекомендую просмотреть вакансии по нему в вашем регионе. Это очень важная информация, которую вы должны принять во внимание. Иначе может получиться, что вы выучитесь и потратите на обучение деньги и время, а работу потом найти не сможете. Если на местном рынке труда вакансий нет, то всегда есть возможность поискать удаленную работу. Но нужно учитывать, что чем ниже уровень разработчика и чем меньше у него опыта, тем больше конкуренция. Поэтому новичкам найти удаленную работу сложнее, чем опытным специалистам. Я знаю людей, которые на начальном уровне смогли найти проект и начать работать удаленно, и не утверждаю, что это невозможно. Мы говорим лишь об общих тенденциях.

А пока вернемся к страхам и психологическим барьерам. Еще одним из страхов, которые беспокоили меня на начальном этапе, стало опасение потерять финансовую стабильность.

Страх потерять финансовую стабильность

Я человек семейный, поэтому смена профессии — это серьезное решение, которое касается всех моих близких. Для меня очень важно, чтобы люди, с которыми я живу, чувствовали себя счастливыми, потому что это напрямую влияет на мое состояние. Если дома постоянно ссоры и недопонимание, это забирает всю энергию и лишает любой мотивации что-то делать. Я знаю, как тяжело моей жене воспитывать ребенка и обеспечивать наш быт, сколько времени и сил это отнимает. Кроме того, никто не отменял дела по дому, поэтому финансовое обеспечение жизнедеятельности нашей семьи я изначально взял на себя. В результате, когда я принимал решение сменить профессию, мне нужно было тщательно обдумать финансовые риски.

Да, так сложилось, что на момент смены специальности мое финансовое положение было не самым радужным. Но тем не менее я контролировал ситуацию и имел план по ее улучшению. Поэтому, хотя на тот момент денег было не много, я был спокоен и ясно видел всю картину происходящего. А когда все под контролем, волнуешься не так сильно.

Но переход в другую профессию — это шаг в темную комнату, тем более когда речь идет об одной из самых сложных специальностей. Это кардинальная смена гуманитарного направления на техническое, где я не имел никакого опыта, знаний и компетенций. Поэтому никаких гарантий у меня не было, так же как и понимания того, как вообще это все будет происходить.

Если хорошо подумать, то страх потери финансовой стабильности — это не что иное, как страх потери контроля. Мы боимся оказаться в ситуации, когда у нас не останется никаких прогнозов, не будет какого-либо запасного плана. По сути, мы боимся, что *ничего не получится*.

Но насколько реален этот страх и насколько вообще велик шанс, что совсем ничего не получится? Давайте поразмышляем об этом вместе.

Я бы сравнил страх смены профессии со страхом темноты. У страха темноты есть официальное название — никтофобия. И согласно исследованиям, человек, одержимый никтофобией, боится не самой темноты, а того, что именно может в ней скрываться. Он опасается тех событий, которые могут с ним произойти, пока он пребывает в темноте.

Со сменой профессии то же самое. Я не боялся менять саму профессию, но всерьез опасался того, что может произойти в результате. Я боялся неизвестности, которая ожидала меня как во время обучения, так и после него. Поэтому был велик соблазн оставаться в зоне комфорта, где меня не все устраивало, но зато все было ясно и понятно.

Я проходил через эту темноту, поэтому хочу поделиться способом, который, надеюсь, поможет и вам.

Для того чтобы не бояться войти в темноту, нужно включить свет. Так же и со сменой профессии. Следует нащупать нужный путь, чтобы идти было не так страшно. Я называю это «методом фонарика».

Объясню поэтапно ход своих рассуждений, которые в итоге позволили мне решить проблему со страхом неизвестности. Этот алгоритм поможет не только избавиться от опасений, связанных со

сменой профессии, но и вообще спланировать решение многих жизненных ситуаций.

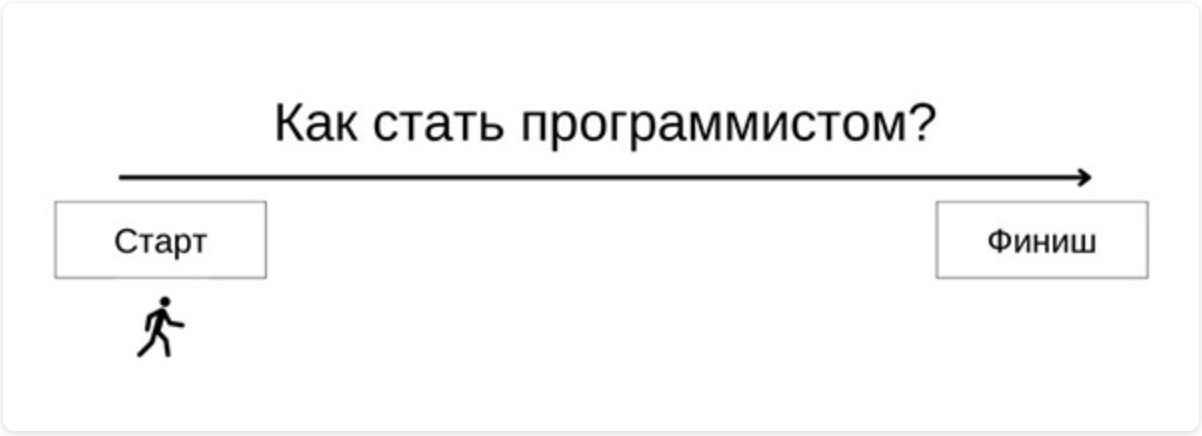


Рис. 8. Как я рассуждал изначально

На рис. 8 вы видите изначальную картинку в моей голове. Так в сыром виде выглядит задача, над решением и планированием которой я собираюсь подумать. На этом этапе у меня есть только вопрос «Как стать программистом?», у которого нет четко сформулированного начала и конца. Это и есть вопрос из «темноты», в нем ничего нет, он состоит из пустоты, поэтому нужно внести сюда конкретику.

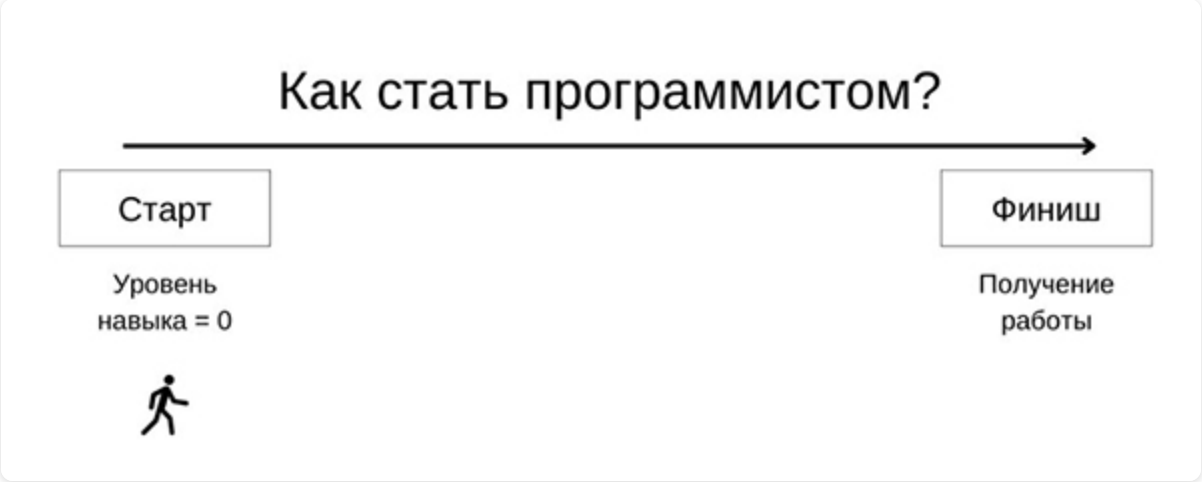


Рис. 9. Этап 1. Определяем начальную и конечную точки пути

Этап 1

Первым шагом, который я сделаю, будет формулировка условий, из которых я исхожу, и установка четкого критерия достижения цели, к которой я буду идти.

В данном случае в самом начале пути я определяю, что мой навык программирования равен нулю. Благодаря прокачке этого навыка я смогу в конце пути претендовать на вакансию программиста. То есть на правом конце прямой у меня появляется четкий показатель того, что можно считать успехом, — это получение работы. Заметьте, что я не считаю успехом само по себе освоение навыка программирования и способность самостоятельно делать приложения. Именно получение работы — тот критерий, который будет обозначать финишную ленту в этом забеге.

Таким образом, осуществив эти маленькие изменения в карте моего пути и четко сформулировав финишную цель, я уже могу понимать направление, в котором должен двигаться. А когда ты знаешь направление, ты уже не чувствуешь себя потерянным в невесомости. Я ясно вижу выход, к которому нужно идти. Осталось только начать движение.

А чтобы сделать этот путь легче и пролить еще немного света на эту «карту», мне нужно добавить больше деталей.

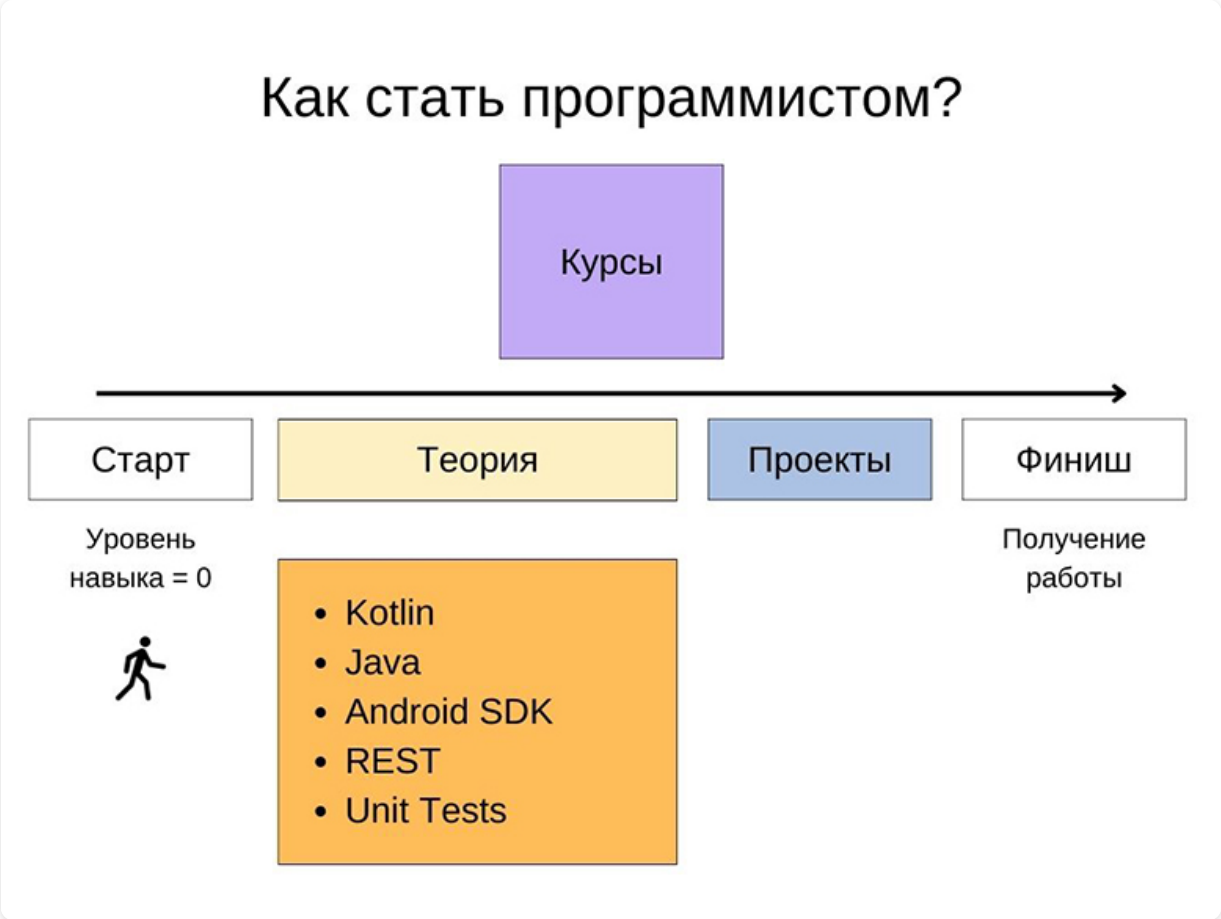


Рис. 10. Этап 2. Добавляем детали

Этап 2

На этом этапе я продумываю нюансы, которые помогут мне достичь результата. Вот примерный алгоритм действий (у вас он, конечно, может быть совершенно другим, здесь важно уловить суть).

В самом начале я иду на сайты с вакансиями, которые мне интересны. В моем случае это вакансии для Android-разработчиков. Я выписываю основные технологии, которые потребуются для работы. На рис. 10 я поместил их в оранжевую «коробку». И на основе этого списка я планирую прокачивать блок теории (он обозначен желтым цветом).

Саму прокачку я планирую осуществлять с помощью различных курсов (фиолетовая «коробка»), потому что в университет я поступить не могу, на это у меня нет времени и ресурсов.

Один из основных критериев выбора курсов — в конце должны быть реальные проекты, которые можно будет показывать на собеседовании (синий блок). Поэтому итоговая схема, которая у меня получилась, состоит из блока с технологиями, которые мне нужно освоить для начала работы Android-разработчиком. Исходя из этого списка с помощью фиолетового блока «Курсы» я прокачиваю блоки «Теория» и «Проекты», что в итоге должно помочь мне дойти до финиша и получить работу.

Составив подробный план пути, я уже не чувствую себя в темноте. Я вижу конкретную цель, которую мне нужно достичь, и понимаю конкретные шаги, которые мне нужно предпринять для ее достижения. Благодаря такому подходу я избавляюсь от страха неизвестности.

Подобный «метод фонарика» я использую абсолютно везде. Если есть желание чего-то добиться, то я просто обозначаю конкретную итоговую цель и расписываю этапы ее достижения.

Конечно, жизнь вносит свои коррективы, и я на личном опыте убедился, что планы не всегда совпадают с реальностью. Где-то ты изначально делаешь неправильные расчеты и не учитываешь всех нюансов либо случаются события, которые ты не можешь контролировать. Это абсолютно нормально, ведь все мы люди, и в этой жизни может случиться все что угодно. Но суть не в том, чтобы все четко и детально спланировать, а в том, чтобы избавиться от еще одного сдерживающего страха и начать движение вперед. И такой подход решает эту задачу в моем случае на сто процентов.

Конечно, не только планирование помогает избавиться от страха финансовой нестабильности, а еще и осознание того факта, что программист даже на начальном этапе своей деятельности зарабатывает вполне нормальные деньги, которых хватит на удовлетворение основных потребностей. А иногда даже больше, чем специалисты с серьезным опытом из других областей. Поэтому начинающему программисту должно хватать на жизнь. А если принять во внимание тот факт, что продвинуться по карьерной лестнице можно очень быстро, то на начальную зарплату не придется долго жить, и спустя короткий промежуток времени уже можно будет получить хорошую прибавку.

Но вернемся к вопросу о страхе потерять финансовую стабильность и о моих сомнениях. Побороть эти сомнения мне помогло осознание того, что стабильность — это вообще понятие довольно иллюзорное. Порой, когда мы вроде бы находим свое место в этом мире, нам кажется, что так будет всегда. Но мы забываем о множестве факторов, которые могут все изменить. Простой пример — появление COVID. Кто знал, что он так глобально перестроит весь мир, да и вообще что такая перестройка возможна? COVID перевернул весь мир с ног на голову, и кто-то оказался наверху, а кто-то внизу. Те бизнесы, которые еще вчера считались самыми стабильными и серьезными, с приходом COVID испытали жуткие потрясения, некоторые и вовсе прекратили существование. В результате куча людей осталась без работы.

То есть вся эта история с пандемией показала, что любой бизнес имеет слабости и в любой момент могут возникнуть проблемы. А уж если ты трудишься по найму, то неважно, на какой должности, ты в любой день можешь остаться без работы.

Однажды я вспоминал свой опыт работы в различных сферах и задумался, насколько мало я всегда мог контролировать ситуацию и

как легко все могло измениться. Например, когда я трудился по найму продавцом, я полностью зависел от работодателя и от успешности компании. Вдобавок к этому моя специальность позволяла работодателю в любой момент уволить меня и нанять другого. Я был легко заменяем. Так и случилось. Компания закрылась, и я был уволен.

Когда у меня появился собственный интернет-магазин, мой бизнес зависел от того, предоставит ли поставщик хорошие скидки. Это и погубило меня. Как только начались проблемы с поставками товара, мне не хватило опыта и знаний, чтобы найти решение, поэтому бизнес пришлось закрыть.

Был и такой случай, что однажды утром, завтракая и готовясь выезжать в офис, я получил СМС от начальника, где было написано, что я уволен. Просто так. Увольнение по СМС. Прошлым вечером, до этого, я разговаривал с ним у него в кабинете, обсуждая рабочие дела, и он ни слова мне не сказал, что собирается уволить. И вот с утра я получаю такие новости.

Это три примера из моей собственной жизни, когда все было вроде бы хорошо, но в одночасье ситуация изменилась. И я уверен, что у вас тоже найдутся такие истории, когда все было вроде бы стабильно, но в один момент что-то пошло не так, и вы ничего не могли с этим поделать.

Поэтому я понимал, что наличие навыка программирования может стать для меня тем инструментом, который позволит мне не беспокоиться о завтрашнем дне. И вот почему:

- этот навык имеет большой спрос, и найти компанию, желающую за этот навык заплатить, довольно просто;
- программист всегда может работать удаленно, а это значит, что он открыт для работы по всему миру. Этот факт напрямую

влияет на показатели стабильности. Ведь даже если в стране проживания все плохо, нет работы и беднота, то всегда можно найти работу за рубежом;
- если нет желания работать по найму, то есть и другие варианты: найти работу на сайтах фриланса либо начать создавать собственный IT-продукт и развивать свой стартап.

Все перечисленные мной факторы способны повысить уровень стабильности и дать возможность планировать свою жизнь. Конечно, я согласен, и у программистов могут быть свои проблемы: можно потерять работу либо компания закроется. Но тем не менее, даже если ситуация нестабильна, всегда есть мягкие сценарии выхода из кризиса. То есть при правильном подходе решение все равно быстро найдется, в отличие от ситуации, когда у человека, например, есть свой бизнес, и ничего кроме этого. Или когда он живет в маленьком городке, где есть один завод, работает на этом заводе лет десять, а потом предприятие закрывают, и непонятно, что делать дальше. В обоих случаях из создавшегося положения выйти очень сложно. Именно поэтому, когда я начинал свой путь программиста, я вообще не смотрел на заработки на начальном уровне. Моей целью было начать получать профессиональный опыт.

Теперь, когда мы более-менее разобрались со страхами, которые посещают нас на этапе принятия решения, я продолжу свой рассказ о том, как я обучался новой профессии.

Первые шаги. Покупка курсов и выбор ноутбука

Итак, в тот же вечер, когда я принял решение заняться разработкой приложений на Android, я купил свои первые курсы, потратив перед этим пару часов на сравнение предложений и поиск отзывов.

Выбрав один из вариантов, я сразу заплатил за него, тем самым отрезав себе путь к отступлению.

Это, кстати, еще один очень важный мотивационный момент. Дело в том, что когда я учусь бесплатно, я воспринимаю это как занятие чисто для себя. Нет никаких обязательств и четкого пути продвижения. Пробуешь то, пробуешь это... Но с таким подходом прогресса лично у меня либо не происходит, либо он идет намного медленнее, чем когда есть четкий курс и ясное понимание целей. А без прогресса и успеха существует большой риск забросить это дело. Плюс отсутствует часть мотивации, которая важна для движения вперед.

Но стоит мне заплатить деньги, как мой подход к обучению сразу меняется. Я беру ручку, покупаю новую тетрадь и выделяю достаточно времени на освоение программы. Помимо этого, я снимаю с себя ответственность и становлюсь ведомым. Так гораздо легче. Меня ведут по пути познания, а я просто плыву по течению и двигаюсь из пункта А в пункт Б.

Однако просто купить курсы недостаточно, нужен еще нормальный ноутбук, чтобы можно было выполнять различные задания. Мне в личные сообщения приходит много вопросов о том, какой ноутбук выбрать для изучения программирования. Почему-то многим

кажется, что ноутбук обязательно должен быть самым лучшим. Многие даже берут кредиты, чтобы приобрести Apple MacBook.

Мой личный опыт показал, что не обязательно сразу, на старте покупать дорогое «железо». Поскольку я решил заниматься разработкой на Android, мне достаточно было ноутбука с системой Windows. Денег на тот момент было совсем немного, поэтому я зашел в интернет и выбрал самый дешевый ноутбук фирмы Lenovo. Почему именно Lenovo? Потому что именно эти ноутбуки имеют максимальное количество оперативной памяти при минимальной цене. Во многих компаниях техника этого бренда используется для выполнения повседневных офисных задач.

Решающим критерием для покупки стали 16 Гб оперативной памяти. Этого было вполне достаточно, чтобы запустить эмулятор, необходимый для разработки приложений на Android. Единственный минус, о котором я не знал на тот момент, — что выбранный мною ноутбук создан на базе процессора AMD, который не имел функции виртуализации. Простыми словами, он не мог запускать эмулятор, на котором я тестировал бы код. Но об этом я узнал уже после того, как начал обучение. Возвращать ноутбук в магазин смысла не было, потому что аналоги стоили намного дороже. Поэтому я нашел решение в виде эмулятора Genymotion, который позволяет запускать Android на компьютерах, где эта возможность изначально не предусмотрена. Впрочем, с тех пор прошло немало времени. Некоторые подписчики писали мне, что даже на процессорах AMD уже возможна виртуализация, но я лично не проверял.

Самое главное, что за те деньги, которые я потратил на покупку ноутбука — а это было примерно сто долларов, — я остался очень доволен выбором. Я программировал на том ноутбуке до самого конца обучения, и он меня никогда не подводил.

Поэтому, если вы решаете, какой ноутбук следует покупать, я рекомендую не тратить на это огромные средства. Я бы взял ноутбук по доступной цене, с максимальным количеством оперативной памяти. Главное условие — чтобы он был абсолютно новым. Я также не рекомендую использовать этот ноутбук для других целей, потому что если вы начнете смотреть на нем фильмы, скачивать программы, устанавливать игры, это все может запустить различные процессы, которые будут работать в фоновом режиме, пока ноутбук включен. Возможно, вы не будете даже знать о них, но они со временем начнут тормозить скорость работы компьютера. Именно поэтому я покупал абсолютно новый ноутбук и ничего, кроме тех программ, которые мне нужны были для разработки, на нем не устанавливал.

Курсы, по которым я занимался в начале обучения, имели видеоформат. Я просто смотрел видео, повторял то, что говорит учитель на экране, и надеялся, что когда запущу приложение, оно будет работать так же, как и у преподавателя. Но очень часто, несмотря на то что я внимательно смотрел видео и повторял каждый символ, в итоге у меня все равно выскакивала какая-то ошибка, с которой мне приходилось возиться самостоятельно. Забегая вперед, хочу сказать, что такое может происходить на любом курсе, хоть на бесплатном с YouTube, хоть на дорогом, в классе с реальным учителем. Такова природа программирования. Более того, этот сценарий — часть обычной, будничной работы программиста. Поэтому, сталкиваясь с подобным во время обучения, вы практикуете то, с чем вы будете иметь дело каждый день: программа или приложение не будут сразу работать как надо, будут вылезать какие-то ошибки, на поиск и решение которых вам придется потратить время. В этом и заключается суть работы программиста.

Выбор курсов. Личный опыт

Но продолжим тему обучения. На самых первых выбранных курсах я учился в течение шести месяцев, и, должен заметить, это было довольно непросто. Во-первых, потому что, как я говорил ранее, у учителя всегда все работало, а у меня то и дело выскакивали какие-то ошибки. Во-вторых, мне было сложно воспринимать информацию: казалось, что преподаватель оперирует исключительно техническими терминами. Я, конечно, понимал, что программирование — это наука о технологиях, но некоторые учителя настолько увлекались использованием специализированной лексики, что порой казалось, будто я не на курсе для новичков, а где-то на лекции для продвинутых программистов.

И так продолжалось на протяжении всего обучения. За все время своего пути я поменял несколько курсов, пересмотрел огромное количество различных YouTube-каналов, переслушал десятки различных преподавателей и в итоге могу сказать, что везде было все примерно то же самое, независимо от уровня сложности самого материала.

Здесь я хочу сделать небольшое отступление и рассказать о своем предыдущем опыте обучения. Мне кажется важным сравнить стандартный подход к образованию с тем опытом, который я получил, обучаясь на курсах.

Когда я учился в школе или в университете, я ходил на занятия с понедельника по субботу в фиксированное время, с фиксированным расписанием. Такой формат обучения, безусловно, дисциплинирует. За человека уже продуман график занятий, расписание экзаменов, и ему остается только плыть по течению. Как и многие из нас, таким

образом я получил полное среднее и высшее образование со степенью бакалавра.

После того как я закончил университет, у меня появилась возможность открыть собственный интернет-магазин. Не буду вдаваться в подробности, скажу лишь, что поначалу клиентов было мало. Нужно было как-то привлекать посетителей на сайт. И в тот момент я впервые столкнулся с мотивированным самообразованием[2].

Как именно это выглядело: я вставал в пять утра, чтобы самостоятельно изучать принципы работы алгоритмов Google и «Яндекс», чтобы продвигать свой сайт в поисковиках и получать своих первых посетителей. Это называется SEO (Search Engine Optimization — оптимизация под поисковые системы). Благодаря бесплатным урокам на YouTube я получил свой первый заказ и смог продвинуть интернет-магазин на первые места в поисковой выдаче. В дальнейшем поток заказов стал постоянным. Что касается рекламы, то с помощью мотивированного самообразования и различных курсов, в основном бесплатных, я смог полностью изучить все актуальные на тот момент инструменты интернет-продвижения и получить такие сертификаты, как Google Adwords Specialist, Google Analytics Specialist, Google Analytics E-Commerce Specialist. А также несколько сертификатов по SEO-продвижению.

После этого, следуя все тем же принципам мотивированного самообразования, я углубился в тонкости менеджмента. Благодаря этому у меня получилось устроиться в хорошую организацию и создать с нуля отдел продаж, в котором мог работать абсолютно любой человек без какого-либо опыта. С помощью всех алгоритмов, скриптов и систем продаж, которые были внедрены, я мог зайти в любое незнакомое место и продать услугу незнакомому человеку. Таким же образом могли поступать люди, которые проходили

обучение по специальной системе и продавали по специальным скриптам. И этому всему я научился на курсах, в книгах и видео на YouTube. Мотивированное самообразование помогло мне найти информацию, чтобы достичь моих целей, и я их достиг.

К чему я все это рассказываю? К тому, что имею обширный опыт получения образования во всех классических вариациях, кроме буткэмпа (учебный лагерь для интенсивного изучения программирования). То есть я учился и по стандартной схеме (как в школе или университете, где есть своя система, материалы, ученики и учителя), и по системе мотивированного самообразования, когда изучается только конкретный материал для достижения конкретной цели.

В связи с этим, основываясь на собственном опыте, хотелось бы акцентировать ваше внимание на нескольких важных моментах.

1. Никакой курс обучения не даст вам стопроцентных знаний по предмету.

Какой бы дорогой или именитый курс вы ни выбрали, в ходе обучения вы все равно столкнетесь с тем, что нужно будет искать дополнительные материалы. Где-то придется просто найти объяснение какого-то термина, а где-то — полное описание того или иного решения/метода. Иногда, чтобы сдвинуться с места в основном курсе, мне приходилось покупать дополнительные мини-курсы на других платформах или искать материалы из бесплатных источников. Я просто хочу настроить вас на то, что это нормально, так будет на любом курсе, с которым вы будете иметь дело. Я пишу это для того, чтобы, когда вы столкнетесь со сложностями, у вас были силы искать решение, вместо того чтобы расстраиваться, что курс плохо составлен.

2. Помните, что практика — залог успеха.

Здесь все просто: чтобы научиться программировать, нужно писать много кода (я говорю в первую очередь о программировании). Если на курсе вам дают только теорию и хотя бы половину времени вы не занимаетесь написанием кода, это плохой курс. Справедливость этого тезиса я ощутил на себе в начале пути, когда только постигал основы программирования. Я чувствую это и сейчас, когда изучаю что-то новое для работы. Теоретический материал по программированию в отрыве от практики усвоить очень сложно. Поэтому сейчас я иногда сразу перехожу к написанию кода и уже по ходу дела обращаюсь к теоретическим материалам. А если вы новичок, то теория просто будет входить в вашу голову, вызывать перегруз и благополучно ее покидать. Во всяком случае, так было у меня. Поэтому я считаю, что курс обучения должен состоять преимущественно из практики.

И еще одно важное замечание. Вообще в описании почти к любому курсу, с которыми я сталкивался, обычно было указано, что он основан на практике. Но в действительности я бы разделил практику на две категории. Первая категория — это когда вы практикуетесь в написании кода на том или ином языке программирования. Вторая — это когда вы занимаетесь разработкой собственного проекта.

Приведу конкретные примеры из собственного опыта. Один из курсов, который я приобрел в интернете, начинался с изучения языка программирования Java (на тот момент Kotlin еще не был в обиходе). И первые уроки начинались с переменных и написания примеров по типу: «int a = 1, int b = 2, int c = a + b. Чему равно c?» Потом я тренировался на различных задачах, но все равно испытывал недоумение: я не понимал, что и зачем я делаю, как именно я смогу применить эти знания, помогут ли они мне в создании реального проекта.

Другой курс начался с того, что мы сразу попробовали разработать приложение. Да, вот так просто. Это было примитивное приложение с одним экраном, текстом и задним фоном. Оно называлось «Поздравительная открытка», и цель его состояла в том, чтобы поздравить друга с днем рождения, написав на экране текст. Я не нашел, кого поздравить с днем рождения, поэтому просто написал приятные слова для своей жены, поставил красивый фон и дал ей прочитать.

И в том и в другом случае мы можем говорить о том, что я практиковался. Но подход был настолько разный, что темп обучения ощущался совершенно по-иному. Давайте попробуем сравнить две ситуации. Представьте, что вас на час заперли в комнате, где нет ни окон, ни мебели, забрали мобильник, сказали просто ждать и, выходя, выключили свет. И вы, сидя на полу, мучительно проживаете каждую секунду, которую вы проводите в этой темной комнате. Время тянется медленно, и вы хотите только одного: чтобы эти мучения как можно скорее прекратились.

А теперь представьте, что вас посадили в комфортное массажное кресло перед большим телевизором, включили ваш любимый фильм и рядом поставили вкусную еду и напитки. В этом случае время пролетит просто молниеносно. Вы не почувствуете, как пройдет этот час.

В обучении то же самое! Там, конечно, не будет столь сильного контраста эмоций, но суть примерно такая же.

Поэтому практика бывает разной. Если я сейчас при выборе курса вижу, что мне предлагают только слушать, смотреть и ничего не пробовать создать, я просто не продолжаю. Когда вы изучаете практику, работая над чем-то конкретным, осязаемым, вы понимаете цель ваших действий. Это имеет смысл и доставляет удовольствие вашему мозгу, даже когда процесс идет с трудом. Я помню, когда я

писал «int a = b + c», я не понимал, как это связано с Android-разработкой, мой мозг отторгал эту информацию. А когда я видел, как на моих глазах появляется простенькое приложение, теоретический материал усваивался намного эффективнее.

Работоспособность такого подхода я доказал и собственным примером, выпуская видео на YouTube. В целом могу утверждать, что сам метод обучения через чистую практику работает намного лучше, чем поглощение огромного количества теоретических материалов.

Благодаря тому что материал курса изложен простым и доступным языком, он становится понятен любому человеку, даже тому, который никогда раньше не писал код и думает, что это какая-то страшная магия, доступная только людям, закончившим технический университет. Доказательством этому является множество успешных историй, когда люди, поверив в себя и используя мои рекомендации, смогли дойти до конца и осуществили свою мечту, став программистами, не имея опыта и технического образования и, что немаловажно, не потратив ни одной копейки. В этом вы можете убедиться сами, посмотрев интервью с некоторыми из этих людей у меня на канале, со ссылками на их социальные сети, где они рассказывают о том, как шли к своей цели и самостоятельно становились программистами.

Поэтому обязательно обращайте внимание на то, какую практику предоставляет курс, иначе вы потеряете много времени. А то и вовсе, «сидя на холодном полу в комнате с выключенным светом», решите, что программирование не для вас, и бросите это дело, хотя нужно было просто перейти в другую комнату.

3. Фокусируйтесь на создании реального портфолио.

Продолжая разговор о практике, хочу поделиться своим подходом, который оправдал себя полностью и помог мне при поиске работы. Он заключается в том, чтобы все время обучения программированию складывалось вокруг создания реальных проектов, которые можно будет впоследствии включить в резюме. Не решений задачек или алгоритмов (на это у вас нет времени, по крайней мере, в Android-разработке[3]), а создания собственных проектов для резюме.

Моей первой целью было создание трех приложений, которые можно будет разместить на PlayStore. И все мои действия были сосредоточены вокруг этой цели. Вот несколько основных вещей, которые я постоянно держал у себя в голове:

- мое время ограничено. Пока я учусь, время играет против меня, потому что чем дольше я учусь, тем выше вероятность того, что я брошу это дело. Я могу перегореть;
- проекты, которые я разрабатываю во время обучения, — это основа моего резюме. Мало кто будет смотреть на курсы, которые я прошел, в основном все будут обращать внимание на реализованные мной проекты. Поэтому основное время я тратил только на них, а все остальное — на обучающие материалы, призванные помочь мне в их реализации.

4. Программированию должны учить люди с опытом обучения или публичных выступлений.

И последний пункт, который во время обучения вызывал у меня больше всего эмоций и негодования, — это квалификация преподавателей по программированию. В рекламе тех или иных курсов я часто видел завлекающую фразу, что их ведут действующие практики-программисты из известных компаний (Google, Facebook, «Яндекс» и других). Но потом, включая урок, я слушал монотонную речь, напичканную техническими терминами, и просто записывал

слова учителя в тетрадь, не понимая, зачем это все нужно. Это было сущее мучение. Я считаю, что хороший программист не обязательно будет хорошим учителем. У меня есть некоторый опыт выступлений перед аудиторией, полученный до того, как я стал программистом. Я изучал различную литературу о том, как это делается, и редко видел эти приемы в обучающих материалах по программированию.

Держать внимание аудитории — это целое искусство, и этому нужно учиться точно так же, как и писать хороший код. Я не специалист в области выступлений перед аудиторией, но все же приведу маленький пример из того, что я запомнил. Например, разговаривая с человеком, нужно постоянно переключать его внимание, особенно если обсуждается сложный материал. Сначала человек может просто слушать информацию, но потом нужно перейти на визуальное восприятие, дать какие-то слайды или иллюстрации. Потом можно опять переключить внимание и вовлечь слушателя в диалог, например, задать вопросы, чтобы он подумал, и затем обсудить ответ. И таким образом, человек не перегружается: пока один орган восприятия занят, остальные отдыхают. Этим способом мы помогаем слушателю держать фокус на сути материала. Подобных приемов работы с аудиторией масса, и они совершенно не связаны с программированием. Они изучаются отдельно от навыков написания кода.

Поэтому если я вижу такую рекламу, то всегда уточняю, имеется ли у преподавателя опыт работы с аудиторией. Ведь если я попаду к преподавателю, не умеющему объяснять сложные вещи простым языком, это может очень сильно повлиять на мое отношение к учебе.

Этот момент особенно важен, когда вы не ходите на занятия, а изучаете материал дома. Мало того что вы постигаете сложные вещи, сидя за компьютером, к этому добавляются постоянные отвлекающие факторы в виде мобильного телефона, родных, с

которыми вы живете, и т. д., в то время как вам необходимо соблюдать дисциплину. По сути, в этом случае все держится только на вашем собственном энтузиазме. В таких условиях качество и стиль подачи материала становятся очень важным моментом, который может влиять на успех достижения главной цели.

Лично для себя я решил, что абсолютно не важно, учусь ли я в классе или смотрю видеоуроки. Главное здесь — подача материала. Энергетика, манера речи и лексикон рассказчика очень важны. И если с самого начала я вижу, что в материале присутствует большое количество технических терминов, для меня это ненормально. Раньше я думал, что раз я изучаю сложные технические темы, курс должен включать в себя много непонятных терминов. Но, став программистом и поработав в этой сфере, я понял, что так быть не должно. Конечно, совсем без технических терминов в обучении не обойтись, но их появление должно быть плавным и равномерным. А если я слышу формулировку «Здесь мы используем вот это, но вам это пока не нужно знать, мы это будем изучать позже», то это плохой знак. Это значит, что курс плохо структурирован либо преподаватель не может объяснить материал простыми словами. В таких случаях я прохожу еще максимум два-три урока и, если такое продолжается, просто перестаю тратить время на этот курс.

Сейчас, будучи уже программистом, я могу с уверенностью сказать, что в самом начале материал для новичков вообще может состоять из минимума технических терминов и формулировок. На старте обучение должно быть легким и понятным. Дети уже сейчас учатся программировать с помощью игр начиная с пяти лет. И тот факт, что вы изучаете сложную техническую дисциплину, не должен оправдывать то, что преподаватели с самого начала забрасывают ученика сложными формулировками и монотонной подачей материала, убивая на корню все желание продолжать учиться.

Курсы и дополнительные материалы для обучения

Продолжая тему, хочу рассказать о своем личном опыте: по каким курсам я обучался и к каким источникам обращался в процессе. Сразу хочу уточнить, что я не использовал эти источники по очереди. Я обращался к ним параллельно, исходя из своей задачи. Ресурсы для обучения были как платные, так и бесплатные, они отличались друг от друга форматами и способом подачи информации, но в конечном итоге каждый из них позволил мне приблизиться к своей цели. Давайте пройдемся по каждому пункту отдельно (я отсортировал их по значимости).

1. *Udacity* — это основной источник, по которому я обучался. Все обучающие материалы там полностью на английском языке. На платформе есть как платные курсы, так и бесплатные. Вообще на момент моего обучения платные курсы во многом дублировали содержание бесплатных, поэтому в принципе можно было бы и не тратить деньги. Но в платном варианте, помимо обучающих материалов, предоставляется поддержка в виде ментора и доступа к форуму, где можно задавать вопросы. Преподаватели также проверяют код и дают обратную связь. Что касается самих материалов, то я бы сказал, что в их содержании есть и минусы, и плюсы. Какая-то часть материалов, которые я изучал, уже устарела, где-то мне не нравилась подача, а где-то все было хорошо. Из явных плюсов хочу отметить, что проекты, которые мы делали во время обучения, были очень интересными, и каждый из них имел смысл. То есть все, что я делал в этих проектах, в итоге мне потом пригодилось в работе. Поэтому если хотите найти идею для своего собственного проекта, вам вполне подойдет платформа Udacity.

2. *YouTube* — основной источник видеоконтента. Вся информация бесплатна. В основном я использовал этот хостинг для поиска решений конкретных проблем, а также для того, чтобы посмотреть на материалы основного курса с другого угла. То есть если на основном курсе мне было что-то непонятно, то я искал аналоги на YouTube, чтобы иметь несколько вариантов объяснения.

3. *Web-сайты из поисковой выдачи* — это бесплатный источник информации, но в текстовом формате. В основном я использовал сайты stackoverflow.com и документацию по Android от Google. В целом изучать технические статьи и блоги мне почему-то нравится меньше, чем работать с видеороликами. Поэтому даже примеры написания кода я предпочитал в первую очередь искать на YouTube, где человек наглядно на экране показывал и объяснял свои действия.

4. *Udemy* — это платный источник информации, хотя там есть возможность найти и бесплатные курсы. В среднем стоимость курса на этой платформе варьируется от 10 до 200 долларов. Есть курсы практически на любую тематику. Для меня это идеальный вариант, и я очень люблю эту платформу, потому что на YouTube не всегда сходу получается найти что-то стоящее и бесплатное, быстро отыскать развернутый ответ на свой вопрос, а тут много хорошей информации, и при этом цена не слишком велика. Плюс еще в том, что у каждого курса есть рейтинг и отзывы, поэтому вероятность того, что время и деньги будут потрачены зря, снижается. Тем не менее Udemy я никогда не использовал как основной источник обучающих материалов, скорее как дополнительный. Например, Android я изучал на Udacity, но на Udemy дополнительно прошел несколько уроков по дизайну, потому что в основном курсе Udacity этот вопрос рассматривался поверхностно.

5. *Javarush* — этот ресурс я использовал для того, чтобы познакомиться с языком программирования Java. Напомню, что на тот момент в Android-разработке язык программирования Kotlin еще не использовался, поэтому мне нужно было учить Java. Самое начало курса мне очень понравилось. Благодаря материалам этого портала многие вещи в моей голове встали на свои места. Но где-то после середины я завяз в задачках и бросил это дело. Как оказалось, бросить нужно было намного раньше, потому что для Android-разработчика было достаточно пройти только первую часть, которая включала обучение основам Java.

6. *Репетитор*. Как видите, он стоит в самом конце моего списка, поскольку он нужен был мне только тогда, когда у меня возникала проблема или ошибка, которую я не мог исправить самостоятельно. Обратите внимание, что репетитор не учил меня программированию, он только помогал справиться с трудностями. Поэтому я прибегал к его помощи очень редко.

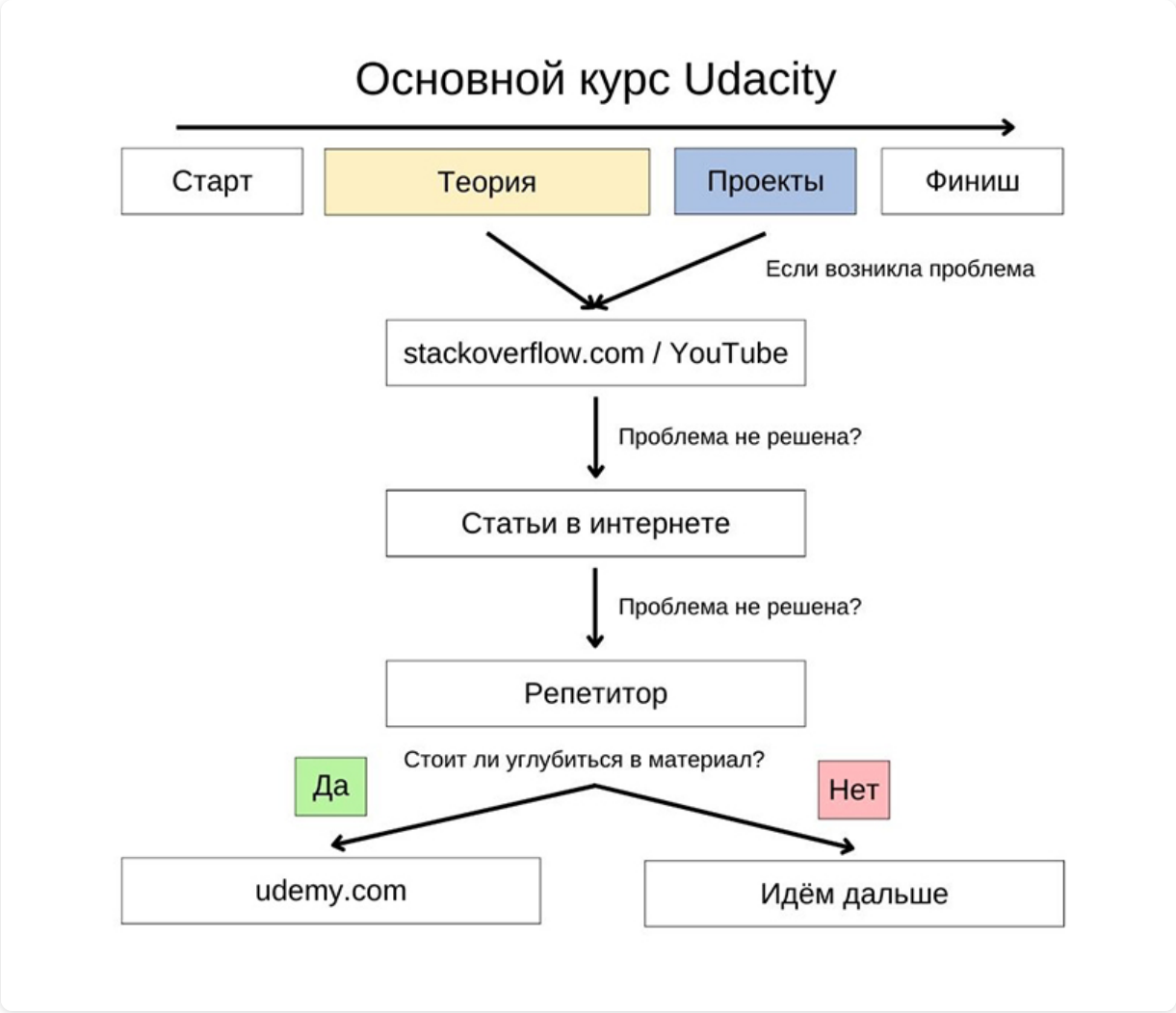


Рис. 11. Алгоритм моего обучения

На рис. 11 я показываю, к каким источникам я обращался в тот или иной момент обучения. Так выглядит итоговый алгоритм моих действий. Изначально я шел по материалам основного курса. Если же возникала проблема или я не до конца понимал материал, я начинал искать видео по данным темам на YouTube или текстовое описание с обсуждением на stackoverflow.com. Если же и на этих площадках я не находил объяснения, тогда я пытался найти уточняющие материалы в блогах разработчиков. Обычно к тому моменту у меня уже имелось общее понимание темы и оставалось найти какие-то детали, которых не хватало.

Несколько раз бывали случаи, что и после изучения всех источников моя проблема все равно не решалась. В таких критических ситуациях я прибегал к помощи репетиторов, которых искал в интернете. Площадок, где можно найти индивидуального учителя, огромное количество. Если же и после урока с репетитором оставались вопросы, я понимал, что дальше двигаться не имеет смысла, нужно остановиться и углубиться в материал. Поэтому я шел на Udemy и покупал специализированный курс на какую-то конкретную тематику. В ходе изучения Android я прибегал к покупке дополнительных курсов на Udemy только два раза: когда мне углубленно пришлось изучать Gradle для Android и Material Design. Это было хорошее решение, потому что, отойдя от основной программы курса, я сменил фокус и смог взглянуть на некоторые темы под другим углом.

Как я победил первый эмоциональный спад

Прошел примерно месяц с момента, как я начал учиться. Этот месяц был наполнен разного рода эмоциями. Поначалу у меня горел огонь в глазах, хотелось с разбегу ворваться в этот IT-мир и в скором времени добиться успеха. Но когда проходит первая эйфория, неминуемо наступает эмоциональный спад, и все чаще и чаще бывают дни, когда с трудом удается заставить себя продолжить начатое.

Я помню один из таких дней. Это была суббота. Я договорился с семьей, что в этот день займусь обучением и никто не будет меня отвлекать. Поэтому моя супруга взяла ребенка, и они уехали в гости. Я остался один, и, по сути, это были идеальные условия для учебы: я выспался, чувствовал себя бодрым и не испытывал никакой тревоги, мне не нужно было никуда идти.

И вот я сел смотреть очередное видео и писать код. После первого просмотра я понял, что ничего не понял, поэтому решил пересмотреть ролик еще раз. Но разобраться снова не удалось. Тогда я решил, что просто начну писать код вслед за преподавателем. Я стал смотреть видео с самого начала в третий раз, периодически ставил его на паузу и медленно повторял действия, которые совершал преподаватель. Так постепенно я дошел до финала. Но когда я нажал на кнопку запуска проекта, чтобы посмотреть результат, то приложение просто не запустилось. Я перешел в программу для написания кода Android Studio и увидел, что вылетела какая-то ошибка. Это не на шутку разозлило меня. Я не мог смириться с тем, что у меня не получается писать код даже

тогда, когда для этого есть все условия. Мне захотелось все бросить, я подумал, что, наверное, это все-таки не мое. Такие мысли возникали у меня и раньше, но на этот раз я уже не мог просто так отмахнуться от них. Мне пришлось лицом к лицу столкнуться с неприятным фактом: я не вижу в себе сил продолжать обучение.

Весь процесс мне напоминал квест, где тебя запирают в комнате, и, чтобы пройти дальше, нужно правильно решить какие-то задания и выполнить определенный порядок действий. И если поначалу мне казалось, что выбраться из комнаты и перейти на другой уровень в целом вполне выполнимая задача, то с каждым уровнем условия казались мне все более сложными и запутанными. Постепенно меня стали посещать мысли о том, что мне начинает надоедать эта «игра». Возникли сомнения, стоит ли вообще мне заниматься тем, что не приносит удовольствия, тем более что приз в конце довольно абстрактный, кажущийся далеким и недосягаемым. Проще говоря, с повышением сложности материала моя мотивация начала падать. Поэтому я осознал, что, помимо улучшения навыка написания кода, мне нужно поработать над мотивацией. В тот момент мне очень помогло мое чувство осознанности, которое позволило взглянуть на этот процесс со стороны и без эмоций оценить ситуацию.

Как я говорил выше, до изучения программирования я имел опыт построения собственного бизнеса и уже испытывал похожие чувства и эмоции ранее. А когда ты бывал в какой-то ситуации, то во второй раз проходить через это проще. Такие эмоции испытывает любой, кто когда-либо начинал заниматься новым, незнакомым для себя делом. Этот путь никогда не бывает прямым и легким. Если изобразить его схематически, то визуально рисунок будет напоминать кардиограмму: волнообразные поступательные движения вверх, вниз, потом опять вверх и снова вниз.

Поэтому, чтобы превозмочь эти внутренние сложности и состояние упадка, я просто поступил так же, как уже поступал раньше: сделал паузу, подышал воздухом, успокоился и после этого продолжил обучение. Ведь я знал, что секрет успеха очень прост: если идти вперед, то рано или поздно ты все равно дойдешь.

Посмотрите на рис. 12. На нем я попытался наглядно показать сравнение пути двух людей, желающих выучить программирование. Вертикальная линия (ось ординат) — это образное изображение желания человека двигаться вперед. А на горизонтальной оси (абсцисс) отображено время в условных единицах. У кого-то один отрезок может быть равен месяцу, а у кого-то неделе.

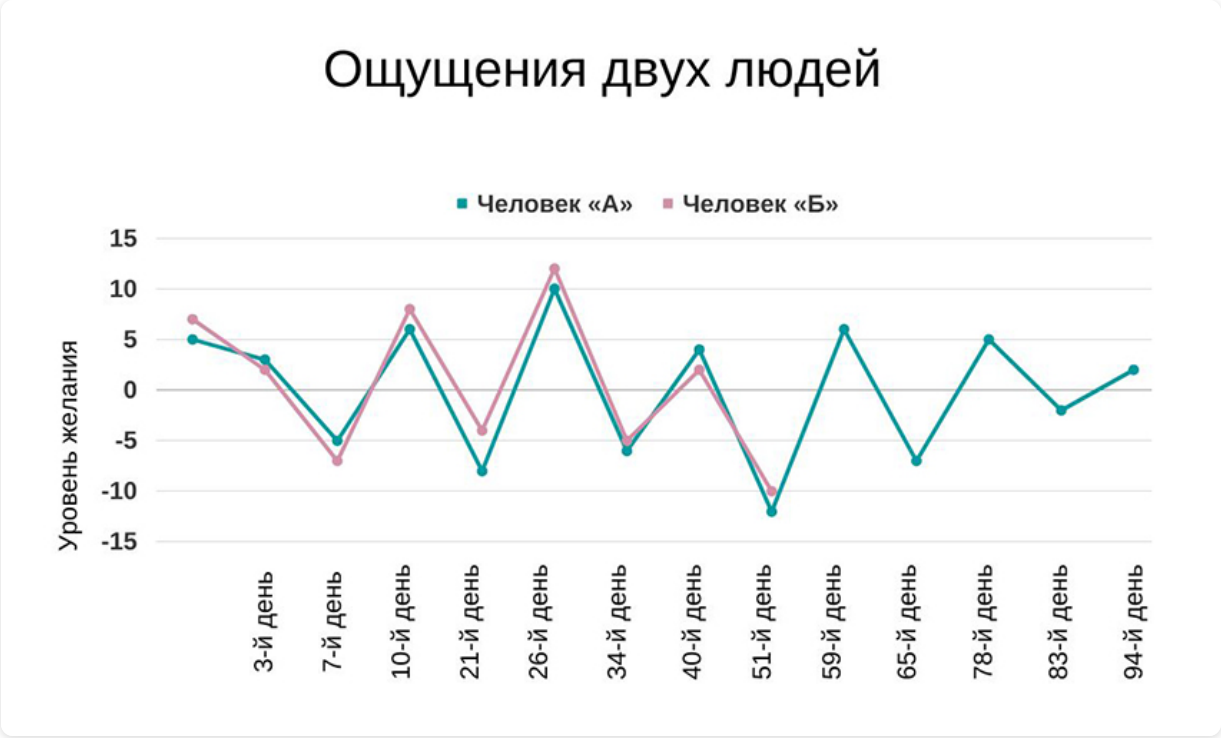


Рис. 12. Ощущения двух разных людей при прохождении одного и того же пути

Давайте теперь разберемся с тем, что происходит на графике. Оба человека начинают свой путь с высоким уровнем желания и энергии,

поэтому красная и голубая отметки находятся на достаточно высоком уровне.

Оба человека начинают движение вперед. И мы видим, что постепенно уровень желания начинает падать. Это естественно, потому что сложность материала растет, а первоначальная мотивация несколько снижается. Так происходит абсолютно с любым человеком.

На седьмой день обучения мы видим, что оба человека сталкиваются с проблемами в понимании материала, и обе линии по горизонтали уходят в минус. Это естественная реакция мозга — стремиться перестать заниматься тем, что не получается. Я называю этот момент этапом отторжения. Нам обычно доставляет удовольствие тот процесс, который нам интересен и в котором мы добиваемся положительных результатов. В противном случае нам приходится преодолевать себя, делать что-то через не хочу.

Обратите внимание, что график пошел вниз у обоих. Потому что рано или поздно каждый человек сталкивается с материалом, который непонятен, или с любыми другими трудностями. И здесь самое главное — дать мозгу стимул, который объяснит, зачем ему нужно продолжать заниматься тем, что ему не нравится.

Этот стимул может быть двух типов: либо получение удовольствия от итогового результата, либо страх от того, что результата не будет. В моем случае я использовал второй вариант. Страх — более сильный мотиватор. Я испугался остаться без работы, испугался ситуации, что моей семье нечего будет есть. Этот страх объяснил моему мозгу, зачем нужно продолжать учиться и почему нужно идти вперед, несмотря ни на что.

Для кого-то лучше сработает именно позитивная мотивация, но для меня это более слабый стимул. Я, разумеется, представлял себе

деньги, которые я смогу зарабатывать, будучи программистом. Воображал, что у меня появятся хороший дом и машина, думал о том, как будет выглядеть мой рабочий день, если я буду работать удаленно. Но это все не мотивировало меня настолько сильно, чтобы в сложный момент заставить себя продолжить разбираться в очередной проблеме с кодом. И только страх давал мне силы идти дальше.

Еще раз подчеркну, что главное в этот момент — состояние осознанности. Для меня это было очень важно. Я понимал, что мне нужно делать и как вывести себя из этого состояния.

Как только мы решаем какую-то сложную задачу, эйфория от этой промежуточной победы начинает вести график желания вверх. Это мы видим на отрезке от седьмого до десятого дня.

Дальше тенденция сохраняется (см. двадцать первый день на графике). Мы справляемся с проблемой, у нас появляются силы и вдохновение, график растет вверх. Но потом новая проблема, и все снова падает. Вот почему я называю это графиком сердцебиения.

Теперь обратите внимание на пятьдесят первый день. На нем вы видите, что оба человека опять испытали стресс от того, что материал непонятен. Этому может быть много разных причин, но лично я чувствовал сильный стресс, когда в моих знаниях обнаруживался пробел и приходилось искать дополнительные материалы вне основного курса. На экране у учителя все работало, и, по его словам, я должен был все это понимать, но в действительности у меня очень часто возникали какие-то ошибки. Поэтому мне приходилось разбираться дополнительно. И чем дольше я застревал на каком-то параграфе, тем ниже падал показатель графика желания.

Представим, что с такой проблемой столкнулись два наших воображаемых человека. И именно на этом моменте человек «Б» решил сойти с дистанции, то есть перестать учиться. А вот человек «А» смог преодолеть эти обстоятельства и решить проблему. Эта победа обогатила его бесценным опытом и придала уверенности в себе, поэтому его график пошел вверх.

По прошествии какого-то времени человек «А» снова столкнется с проблемой, но благодаря тому, что он уже имеет опыт в решении большого количества проблем, его эмоциональное состояние окажется достаточно крепким и стабильным, поэтому его график не просядет, а процесс не прервется.

На оставшемся до финиша пути у человека «А» уже не будет сильных взлетов и падений в его графике. Потому что он просто привыкнет к процессу обучения, а возникающие проблемы не будут вызывать столько негатива и стресса. Решение проблем превратится в привычку. Однако и большой радости и эйфории, возможно, тоже уже не будет, потому что придет понимание: любой процесс состоит из взлетов и падений, и после победы над очередной трудностью рано или поздно появится новая. Однако если не сходить с дистанции и в размеренном ритме решать все вопросы, то есть все шансы дойти до вожделенной цели. Так было лично у меня.

С помощью этого графика я хотел объяснить несколько вещей, способных значительно упростить ваш путь. Во-первых, абсолютно все люди будут испытывать проблемы в изучении новой дисциплины, какая бы она ни была. И график желания двигаться вперед будет прыгать вверх и вниз. За очередной победой придет новая проблема, потом найдется решение и появится чувство эйфории, следом — новая проблема, и так далее. Это абсолютно нормально.

Во-вторых, важно помнить, что со временем вы привыкнете к такому ритму, и появление новых проблем будет вызывать меньше стресса. Это я наблюдаю сейчас, в своей каждодневной работе. Бывают задания, которые я выполняю с легкостью и радостью, но бывают и такие, которые вызывают только стресс и нервозность. После таких заданий я чувствую себя эмоционально выжатым. Но тем не менее я отношусь к появлению таких проблем абсолютно нормально и уже не переживаю так сильно.

Но основной вывод, которым я хотел бы поделиться, звучит так: не важно, в какой точке пути вы сейчас находитесь, самое главное — это движение вперед. Обучаясь осознанно и помня о цикличности проблем, просто продолжайте идти. Со временем эти эмоциональные скачки станут не такими яркими. Когда вы столкнетесь с очередной сложностью, вспомните мой график и решите, кто вы на нем — человек «А» или человек «Б».

Страх перфекциониста

Прошло примерно три месяца с момента, как я приобрел первый курс. За это время я постепенно начал чувствовать, как учеба превращается в рутину. Никакой эйфории больше не было. Розовые очки тоже разбились вдребезги. Во время обучения я постоянно чувствовал себя слишком глупым, недостаточно подготовленным. Учителя с преувеличенной бодростью рассказывали материал, и мне казалось, что я с такой же легкостью должен его воспринимать. Более того, когда я заходил в поисках решения той или иной задачи на форумы или в чаты начинающих программистов, я видел, как легко люди разговаривают на эти темы, предлагают решения и прекрасно представляют себе, как все это работает. А я порой вообще не понимал, что происходит. Код казался мне набором непонятных символов, а терминология — дремучим лесом. Мне казалось, что этому надо было учиться еще тогда, когда я был студентом и голова не была забита каждодневными заботами.

Я помню, как однажды посмотрел урок, где показывали, как нужно внедрять обработчик кнопок. То есть как сделать так, чтобы после нажатия на кнопку что-то происходило. Я добавил код, но почему-то у меня ничего не работало. Я решил посмотреть, как решаются подобные задачи, на stackoverflow.com, чтобы попытаться понять суть и найти разницу между моим кодом и кодом учителя. Там я увидел десятки вариантов, предложенных пользователями. Первое, что проскочило у меня в голове: «Зачем столько вариантов? Разве нельзя дать один, чтобы просто работал?» Дальше я начал читать комментарии пользователей. Люди легко обсуждали данную тему, задавая друг другу вопросы и отвечая на них. В тот момент я подумал, что это невозможно понять. Наверное, все эти люди — молодые студенты, которые учатся программированию в

университете. Меня накрыло сильное чувство страха и неуверенности. В тот момент в моей голове была только одна мысль:

«Я всегда буду недостаточно хорош для этого».

В наше время, когда социальные сети являются неотъемлемой частью повседневной жизни, а YouTube уже давно для многих заменил классическое телевидение, страхи и неуверенность в себе стали проявляться еще чаще. Ведь эти инструменты дали нам возможность смотреть на мир намного шире. Мы можем регулярно наблюдать за людьми, которые нам интересны, видеть их жизненные успехи. И естественно, мы начинаем, часто на бессознательном уровне, сравнивать их достижения со своими собственными. Поэтому когда мы, например, становимся свидетелями успешности молодого парня, который разработал одно-единственное мобильное приложение и стал миллионером, в голове невольно всплывают фразы вроде «Я недостаточно хорош», «Я ничего не понимаю», «Мое время уже ушло, я слишком стар для этого», «Сейчас неподходящий момент, у меня куча забот и проблем».

Я справился с этими деструктивными мыслями следующим образом. Посмотрев правде в глаза, я понял, что на самом деле эти мысли — не более чем удобное оправдание для моего бездействия и лени. И я постарался ответить себе честно: если бы не было доступа к интернету и я бы не видел всех этих успешных картинок, стал бы я действовать или нашел бы другую причину остановиться? И если мой ответ «Нет, я не стал бы ничего делать», значит, причина только в моей собственной лени. Какое мне дело до чужих достижений и мнений? Ведь у меня свой путь и собственная жизнь. Поэтому я просто взял себя в руки и предпочел хоть что-то делать, не оглядываясь по сторонам.

Второй фактор, который также постоянно тормозил меня, не давая что-то начать, — это постоянное ожидание идеального момента. Но в

итоге я пришел к выводу, что такой момент никогда не настанет. Все истории великих ученых, бизнесменов и спортсменов свидетельствуют о том, что в их жизни тоже не было идеальных обстоятельств для того, чтобы все сложилось. Наоборот, многие истории успеха начинаются именно с ситуации, когда у человека было огромное количество проблем и сложностей. Найдя в себе силы именно на пике проблем, они поняли, что у них есть энергия и мотивация дойти до своей цели.

Что касается меня, то, как я писал уже ранее, угрызения совести от того, что я не попробовал, намного сильнее страха вероятного провала. Поэтому я просто пробую, делаю что-то с максимальной отдачей и измеряю результат не тем, получилось или нет, а тем, какие уроки я вынес и какой полезный опыт я приобрел. Любая моя деятельность — тому подтверждение.

Например, мой блог в социальных сетях не идеален. Когда-то там не было ничего, кроме идеи. Но я планомерно и терпеливо двигался вперед, публикуя посты, которые могли понравиться незнакомым людям. Сначала в блоге появились десять незнакомых людей, которым было интересно читать мои мысли, потом их стало сто, потом тысяча. Сейчас у меня несколько тысяч подписчиков, и их количество растет. Но всего этого не было бы, если бы я сидел и ждал идеального времени для старта или если бы решил потратить большое количество часов на обучение, перед тем как начать, или если бы я думал, что у меня не хватит времени. Я просто публиковал посты, отслеживал реакцию аудитории и по ходу дела исправлял свои ошибки, учился и продолжал идти дальше. Если бы с самого начала я начал сравнивать себя с другими, то я бы так и не рискнул вести блог. Ведь в соцсетях уже столько крутых блогеров, у которых миллионы подписчиков, по сравнению с ними я — маленький муравей. Однако это меня не пугает.

Так же было и с программированием. Я знал, что опыта и образования у меня нет, времени тоже не то чтобы очень много (во всяком случае, меньше, чем у молодых ребят). Быть может, у меня меньше возможностей и гибкости. Но нужно было что-то делать. И, подойдя к этому вопросу с умом, я в итоге дошел до своей цели.

Точно так же я поступаю во всех своих остальных начинаниях, будь то новые проекты, канал на YouTube или эта книга. Кстати, написание книги стало одним из самых сложных дел, которыми я когда-либо занимался. В детстве и в юности я читал не так много, как хотелось бы, поэтому даже подумать не мог о том, чтобы написать свою книгу. Но вот я пишу ее, и мне неважно, будет ли она лучше или хуже других изданий на аналогичную тему. Я думаю только о том, что должен продолжать и дописать ее до конца. И если хотя бы одному человеку она поможет достичь своих целей и изменить жизнь к лучшему, став программистом, значит, мои усилия не напрасны.

Итак, чтобы преодолеть свой перфекционизм, я рекомендую просто перестать обращать внимание на возможные неблагоприятные факторы и условия. Нужно просто видеть впереди цель и идти к ней понемногу, маленькими шагами, но каждый день. Это как идти по туннелю, видя свет в самом его конце. Не обращайте внимания на другие мнения и чужой успех. Делайте шаг вперед, а потом еще и еще. Как говорится, дорогу осилит идущий. А лучший момент, чтобы начать, — сейчас.

Страх, что программирование слишком сложное

Прошло еще какое-то время. Я начал привыкать к ритму, с которым я продвигался вперед. На программирование я теперь смотрел немного по-другому. Если в самом начале оно казалось мне чем-то из области фантастики, то сейчас пыль улеглась, и я понял, что программирование похоже на обычный предмет, который можно изучать. Но это понимание пришло только со временем. Поэтому я считаю, что, начав учиться программированию, нужно просто перетерпеть хотя бы два месяца, чтобы мозг адаптировался и начал ориентироваться в этой теме. Как именно перетерпеть? Так же, как это сделал я: просто не останавливаясь, медленно, но ежедневно, по графику изучать теорию и продолжать писать код. Даже одна строчка в день — это лучше чем ноль. А через пять дней это уже пять строчек кода, и это уже на пять больше, чем ноль. Поэтому, поставив себе цель просто не останавливаться и двигаться вперед, я добился того, что программирование уже не казалось мне таким страшным.

Я не утверждаю, что программирование — простой предмет, но, освоив его, понимаешь, что никакой магии нет, нужно просто изучать теорию, писать много кода и делать это на протяжении определенного времени, на регулярной основе. Пройдя эту дорогу, я понял, что сложно не само программирование, а скорее процесс изучения. Причин этому несколько. Я уже говорил о них: это то, что программирование часто преподают специалисты без педагогических навыков, а также то, что в этом предмете огромное количество технических терминов. Как быть с ними? Я думаю, что не нужно пытаться выучить их все как можно быстрее, достаточно просто понимать, о чем идет речь, и уметь пользоваться

поисковиком. А лучше просто чаще и больше писать код, и тогда понимание придет само по себе. Без практики теория бесполезна.

Также считаю нужным напомнить еще раз, что важно работать над проектами, которые вам понятны, которые вам интересны и смысл которых вы хорошо осознаете. Пусть это будут не очень сложные проекты: приложение прогноза погоды, калькулятор, чат и так далее. То есть когда вы слышите название проекта, в голове у вас должна всплывать картинка того, как должен выглядеть результат.

Что еще? Во время обучения важно ставить правильные краткосрочные задачи. Не ставьте себе задачу «выучить Android», эту цель нельзя измерить, и она слишком обширная. Вместо этого поставьте себе цель «сделать экран с одной красной кнопкой в центре», она более проста и понятна. И из таких маленьких, понятных, легко реализуемых целей, будто из маленьких кирпичиков, будет строиться процесс вашего обучения.

Возможно, теперь благодаря моим советам вы сможете взглянуть на программирование несколько шире. И если со сложностью программирования вы ничего поделать не сможете, то повлиять на процесс обучения и сделать его более комфортным вполне в ваших силах.

Принцип приоритетности кода

Когда изучение программирования вошло в привычку, ушло чувство тревожности и процесс стал рутинным. Мое сознание прояснилось. Больше не было ощущения, что я нахожусь в каком-то хаосе, где все непонятно. Это дало мне возможность посмотреть на весь процесс обучения со стороны. И вот какие выводы я сделал. Во-первых, я понял, что преодолел сложный этап первых сомнений и теперь планомерно иду к своей цели. Это уже было достижением для меня. Во-вторых, я осознал, что хотел бы ускорить прогресс, поэтому начал анализировать свой процесс обучения и фанатично искать в нем изъяны, которые я мог бы исправить. Можно сказать, что с этого момента началось мое по-настоящему *осознанное обучение*. Я уже не просто слепо следовал инструкциям, которые давал учитель, а мог осознавать свой уровень понимания материала и прибегать к некоторым уловкам, которые позволили мне упростить восприятие информации и ускориться в своем продвижении вперед. Это просветление пришло благодаря тому, что я разобрался со своими изначальными страхами. Я обработал их, взял ситуацию под контроль и таким образом смог сконцентрироваться на написании кода.

Как я уже говорил ранее, мой процесс изучения программирования я представлял в виде айсберга. Так вот, после того как я обработал все первоначальные страхи, я взялся за оптимизацию процесса обучения.

Во время прохождения курсов у меня бывало такое, что я хотел подробно и на сто процентов понять материал, перед тем как двинуться дальше. Я пытался досконально разобраться в информации и углубиться в детали. Бывало, что я тратил много

времени, погружаясь в нюансы с головой, и в итоге уходил в них слишком глубоко.

Отчасти это было необходимо: нельзя просто поверхностно пробежаться по курсу обучения, сделать код по принципу «лишь бы работал» и идти дальше. Потому что, не поняв материал, вы рискуете в дальнейшем столкнуться с теми же проблемами, от которых пытались уйти.

Так как же поступить? Я вам не смогу дать точной инструкции на этот случай, но поделюсь своим личным опытом и техниками, которые мне помогают не погружаться слишком глубоко, но и не бежать по верхам, не понимая сути изученного.

Первым делом я рекомендую начать подходить к обучению более вдумчиво. Поясню, что я имею в виду. Скорее всего, вы, как и я, выберете какой-то один курс, и если он качественный, то его материалы нужно изучать полностью. Но в большинстве случаев их будет недостаточно, поэтому в ходе практики написания кода вы, вероятно, будете пользоваться сторонними ресурсами. Так вот, ваша задача — не уйти в них с головой.

Поясню на одном из примеров, с которыми я столкнулся в ходе обучения: мне нужно было реализовать процедуру создания нового пользователя и сохранения его данных на сервере. Другими словами, обеспечить процесс регистрации, который каждый из нас проходил хотя бы раз в жизни. Данный функционал часто используется в web-, Android- или iOS-разработке.

Важный момент: здесь и ниже вам встретятся технические описания, которые, возможно, могут вызвать некоторые затруднения. Пусть вас это не пугает, на данном этапе техническая информация выполняет чисто ознакомительную функцию, и вам не обязательно досконально разбираться во всех тонкостях и особенностях внедрения

технических решений. Нужно просто уловить основную идею и ход мысли.

Первое, что я сделал, — это разделил основную задачу на несколько этапов. Разбиение задачи на этапы позволяет систематизировать процесс, находя отдельное решение на каждом из них.

Первый этап. Создание полей ввода электронной почты и пароля, а также кнопки регистрации аккаунта, которые пока не функциональны (см. рис. 13).

Рис. 13. Поля ввода данных и кнопка «Создать»

Второй этап. После ввода данных и нажатия кнопки адрес электронной почты и пароль отправляются на сервер для сохранения (см. рис. 14).

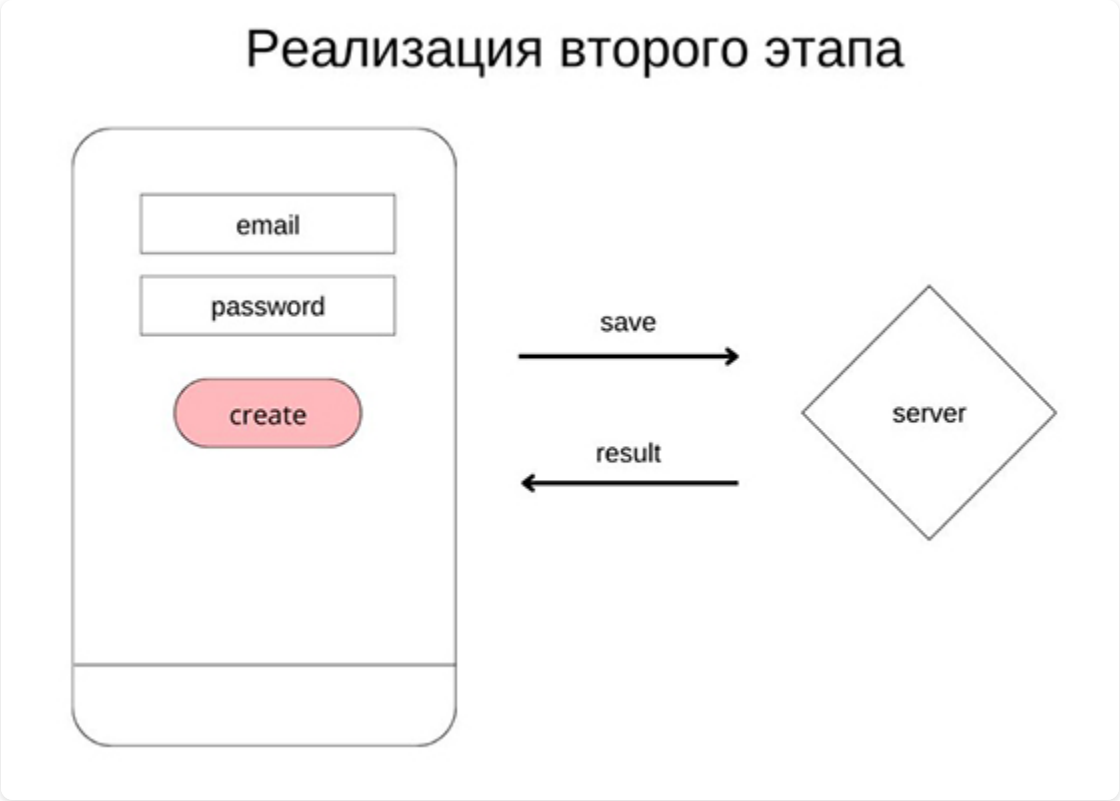


Рис. 14. Отправка данных на сервер

Третий этап. Сервер сохраняет все данные пользователя у себя и возвращает мне на телефон отчет о том, была ли процедура сохранения успешна. При получении отчета пользователю демонстрируется надпись «SUCCESS!» и тем самым сообщается о том, что регистрация прошла успешно (см. рис. 15).

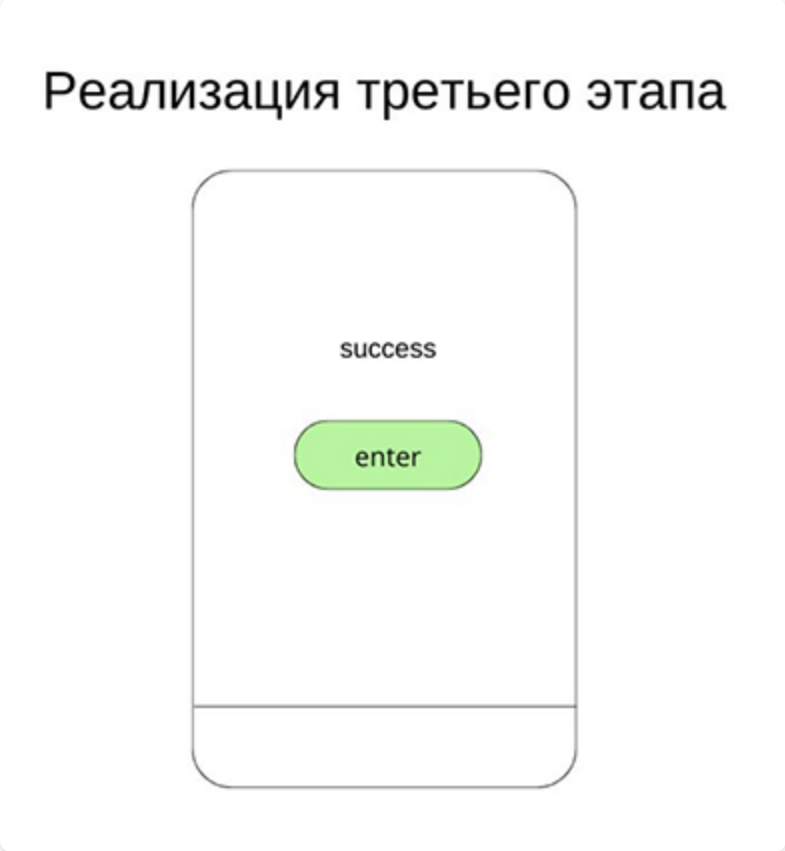


Рис. 15. Экран пользовательского устройства с сообщением об успешной регистрации

Итак, моей задачей было написать код, чтобы все это заработало. Я посмотрел первую часть обучающих материалов и успешно создал первый экран с формой. Первый этап пройден, ура! Я почувствовал воодушевление, потому что у меня все получилось и результаты учителя полностью совпали с моими. У него работает, и у меня работает. Все счастливы!

Затем я приступил ко второму этапу. И тут мой темп продвижения начал снижаться. Это произошло потому, что сложность материала возросла. Делая паузы в процессе просмотра видеоролика, я строчку за строчкой повторял код, который написал учитель. И тут выяснилось, что у него все работает и он собирается переходить к следующему этапу. А у меня при отправке данных приходит ответ с ошибкой: «500 Internal Server Error».

Что делать? Конечно, я пошел в поисковик и начал изучать причины возникновения этой ошибки. Я расстроился, поскольку полностью повторял то, что учитель писал на экране, но результаты у нас разные. В такие моменты я чувствовал досаду, что курс не может предусмотреть подобные сценарии и ошибки, которые могут случаться, если что-то пошло не так.

Я проходил через это огромное количество раз. Более того, когда я уже сам был профессиональным программистом и создал свой курс по разработке на Android, я пытался разобрать самые распространенные проблемы в своих видео и акцентировать внимание учеников на том, как искать решение проблем, если я о чем-то не рассказал. Тем не менее ошибки, причину которых им не удавалось понять, все равно появлялись. Причем не где-то в середине курса, когда человек уже пишет сложный код, а даже на самых первых уроках, когда мы просто устанавливаем программу для разработки на компьютер.

Поэтому я сделал вывод, что появление непредвиденных ошибок — естественная часть процесса обучения.

Но вернемся к моей проблеме. Столкнувшись с ошибкой «500 Internal Server Error», я начал разбираться в этом вопросе, пытаясь все починить. И в один прекрасный момент мне это удалось. Но, смотря видео про причины возникновения ошибки 500, я выяснил, что существуют и другие типы ошибок:

- 501 HTTP Error (Not implemented),
- 502 HTTP Error (Bad Gateway Error),
- 503 HTTP Error (Service Unavailable),
- 504 HTTP Error (Gateway Timeout Error),
- 505 HTTP Error (HTTP Version Not Supported).

И тут у меня возникла мысль, что было бы неплохо изучить и остальные типы ошибок — на всякий случай, чтобы в дальнейшем иметь о них представление и уметь с ними справляться.

Вот тот момент, ради которого я и пишу эту главу. Потому что здесь очень важно вовремя остановиться и не углубиться слишком сильно в материал, который не относится к курсу. Вам нужно двигаться дальше. Максимум, что я порекомендовал бы сделать, — это посмотреть одно видео, где просто делают краткий обзор всех ошибок, но не начинать изучать каждую из них по отдельности. Другими словами, если ваша ошибка исправлена и вы можете продвигаться вперед — лучше именно так и поступить.

Подводя итог этой главы, отмечу несколько основных моментов.

- Обучающие материалы можно условно разделить на две части: материалы курса, по которому вы двигаетесь, и дополнительные, которые позволяют продвигаться дальше.
- Если курс был выбран правильно и вы хотите дойти до конца, то материалы основного курса должны быть изучены полностью. Перескакивать через них нельзя. А вот изучение дополнительных материалов не обязательно, поэтому при работе с ними нужно контролировать себя и задавать себе вопросы: «А действительно ли мне сейчас нужно тратить на это время и как это поможет продвинуться дальше?»
- Сценарий, когда возникают непредвиденные ошибки либо когда результат на вашем экране не соответствует результатам на экране преподавателя, — это нормально. Так будет случаться постоянно, поэтому относитесь к этому как к части обучения. Могу сказать больше: даже когда вы уже станете профессиональным программистом, вы все равно будете постоянно с этим сталкиваться.

Мой алгоритм поиска решений проблемы

Оптимизировав процесс изучения материала и научившись вовремя говорить себе «Стоп», я почувствовал, что стал очень избирательно относиться к тому, на какой материал я трачу время и стоит ли он того. В итоге это позволило мне в течение долгого времени сохранять скорость продвижения на оптимальном уровне. То есть если поначалу я постоянно тормозил и топтался на одном месте целыми днями, а то и неделями, то теперь я практически не снижал скорость до нуля и не тратил на поиск решения проблемы больше времени, чем требовалось.

Проблемы, как я уже говорил ранее, — часть каждодневной работы программиста, и с этим нужно свыкнуться. Но незаметно для себя я обнаружил, что в процессе ежедневной практики у меня выработался свой алгоритм поиска и решения проблем. Оптимизировав этот процесс тогда, я до сих пор продолжаю совершенствовать его в своей профессиональной практике. И в этой главе я хочу детально и с примерами рассказать о нем. Этот алгоритм действий подходит как для новичков, так и для тех, кто потратил какое-то время на обучение и уже имеет небольшой опыт. Освоив этот алгоритм единожды, вы сможете использовать его в своей практике снова и снова.

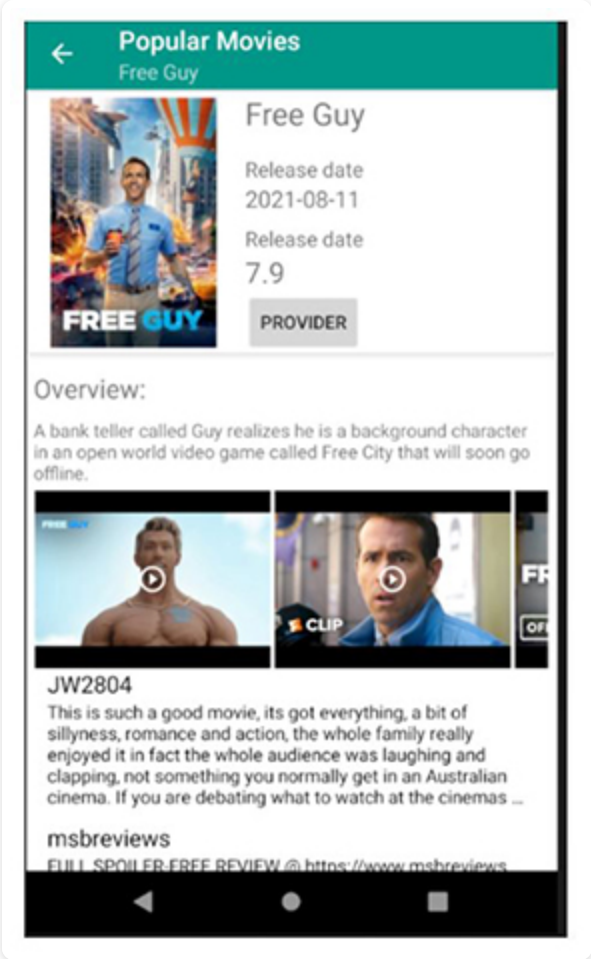


Рис. 16. Приложение Popular Movies

На рис. 16 вы видите одно из приложений, процессом работы над которым я делюсь в своем бесплатном курсе на YouTube. На экране представлена детальная информация о выбранном фильме: постер, подробное описание, видеоматериалы к фильму, дата выхода и рейтинг.

В обучающем видеоролике я не показываю, как добавить трейлеры на экран (они находятся в самом центре экрана и имеют вид горизонтального списка). Я даю ученикам возможность сделать это самостоятельно, но на этом примере я покажу, как решал бы эту проблему, если бы столкнулся с ней впервые. Прошу вас максимально вдумчиво и осознанно прочитать написанное ниже.

Эта информация очень важна, и если вы ее поймете, то сэкономите огромное количество времени и нервов во время обучения.

Как я уже говорил, процесс решения задачи я обычно разделяю на две части. Первая — это поиск инструмента, который позволит мне получить тот результат, который я хочу. И вторая часть — настройка этого инструмента, чтобы он работал так, как мне нужно. Под инструментом я в данном случае подразумеваю кусок кода, который позволит мне создать список на экране.

Тут сделаю небольшое лирическое отступление для тех читателей, которые еще не пробовали писать код. Во время разработки вы будете в основном настраивать готовые куски кода, созданные другими разработчиками. Этот процесс напоминает починку машины: вы не разрабатываете детали для ремонта с нуля, а используете готовые. Для того чтобы прикрепить деталь, вы также воспользуетесь готовыми шурупами и болтами. Так же и в программировании: есть уже разработанные инструменты, которые нужно настроить, чтобы решить ту или иную задачу.

Поэтому для отображения списка мне нужно сначала найти инструмент (кусок готового кода), который это поможет сделать. А потом, на втором этапе, когда я буду уверен, что он делает то, что мне нужно, я буду искать инструкцию, как этот инструмент настроить.

То есть еще раз: сначала нашли нужный инструмент, потом нашли инструкцию по его настройке.

Сначала нужно попытаться понять, как вообще отобразить трейлеры списком. Поэтому я сформулирую эффективный поисковый запрос. В самой задаче уже есть подсказка: если я буду работать со списком, значит, мне нужно узнать, как реализовать его на Android. Искать лучше сразу на английском, поскольку подавляющее большинство инструментов программирования имеет название только на этом

языке. А вот информацию, как использовать тот или иной инструмент, уже можно искать на русском языке либо воспользоваться переводчиком и искать сразу все вместе на английском. Вы это лучше поймете на примере, приведенном ниже.

Научиться правильно формулировать поисковый запрос очень важно, поскольку многие неопытные разработчики, не найдя ответа в выдаче, сразу расстраиваются и думают, что решения нет. Даже во время работы у меня бывали случаи, что я тратил несколько часов на поиск решения и ничего не мог найти. Но потом обращался за помощью к коллеге, он формулировал поисковый запрос по-другому и находил ту статью, которая решала мою проблему. Это было очень неловкое чувство, когда ты жалуешься на то, что решения нет, а коллега находит его в один клик. Поэтому навык правильного формирования запроса очень важен. Он приходит со временем и с приобретенным опытом. Осознанно прокачивайте это качество вместе с навыком написания кода.

Но вернемся к нашему примеру. Для внедрения списка видео я сформулировал следующий запрос.

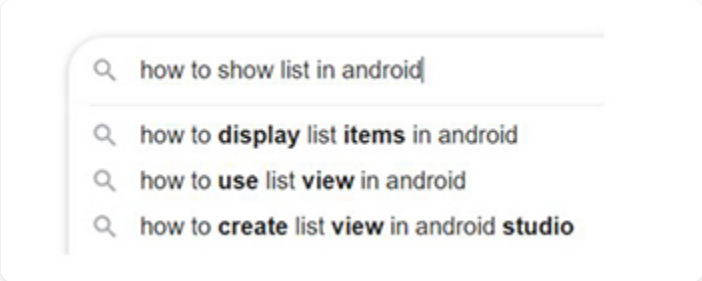


Рис. 17. Поисковый запрос

Давайте разберем эту фразу по словам.

На этапе поиска решения я еще не знаю, какой именно инструмент буду использовать, поэтому мне нужно спросить у поисковика, как это сделать, используя в начале запроса ключевое слово *«how to»*

(*«как»*). Я называю эту часть запроса статической, поскольку она актуальна практически для любого запроса.

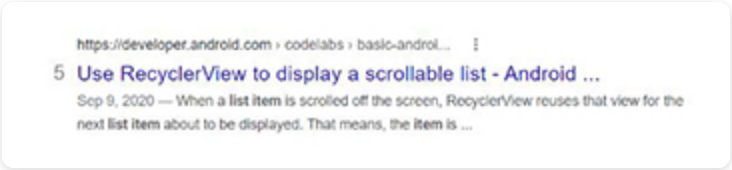
Дальше идет динамическая часть — это та задача, которая передо мной стоит, в данном случае показать список (на английском — *«show list»* (*«показать список»*)). И в конце я обязательно делаю уточнение, что я ищу решение для Android, потому что если не уточнить, то поисковик может выдать решения для web- или iOS-разработки, которые нас не интересуют. В итоге формула выглядит следующим образом:

How to **do something** **in Android**

Как **сделать что-то** **в Android**

СТАТИЧЕСКАЯ **ДИНАМИЧЕСКАЯ** **СТАТИЧЕСКАЯ**

После того как запрос сформирован, выпадает список подсказок. Я выбираю первую, потому что она полностью соответствует моим нуждам, и дальше получаю список результатов с ответом на мой запрос.



18. Один из результатов запроса

На рис. 18 вы видите результат выдачи поисковика с пятого места. Почему я выбрал именно этот результат? Дело в том, что в Android реализация списка может быть осуществлена с помощью двух инструментов: ListView и RecyclerView. ListView — это более примитивный вариант, и он редко где используется, в отличие от RecyclerView — гибкого и удобного инструмента. Это как отвертка и

шуруповерт: и тем и другим можно прикрутить шуруп, но в первом случае это придется делать вручную, а во втором достаточно нажать кнопку.

Тем не менее, когда я, впервые столкнувшись с задачей, искал ответ на вопрос, как сделать список, я начал внедрять именно ListView, потому что он отобразился выше в поисковой выдаче. Но оказалось, что ListView имеет много технических ограничений, и мне пришлось все переделывать. Поэтому в итоге я сделал вывод, что первое место в выдаче не всегда гарантирует качественный ответ. Теперь у меня есть собственный список сайтов, к которым я обращаюсь в первую очередь, вне зависимости от того, на каком месте они находятся. Я уверен, что на этих сайтах больше вероятность того, что я найду качественный ответ на свой запрос.

В частности, я отдаю предпочтение сайту stackoverflow.com. Это самый главный сайт вопросов и ответов для всех разработчиков в мире. Все дороги ведут туда. Обычно там можно найти большое количество примеров, особенно это касается вопросов от новичков. Также имеются уточнения от опытных специалистов, которые интересовались этой проблемой (см. рис. 19). Обратите внимание, что существует и русская версия данного сайта. Вопросы и аудитория английской и русской версий разные, поэтому можно одновременно спрашивать и там и там. По моим личным ощущениям, русское комьюнити более терпимо и лояльно к вопросам новичков, нежели пользователи английского сайта.

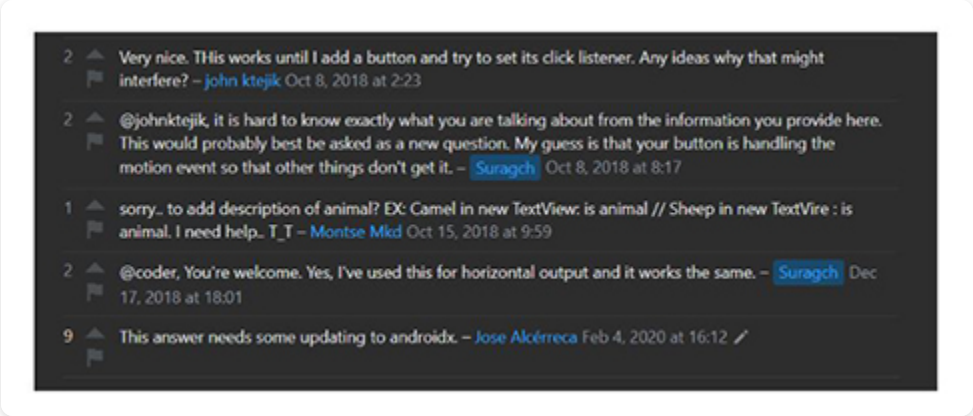


Рис. 19. Комментарии и уточнения пользователей сайта stackoverflow.com

Но самый большой плюс — формат, в котором организована вся информация. Она обычно четкая и по делу, поэтому не придется читать огромную, оптимизированную под поисковики статью с кучей текста.

Во вторую очередь я смотрю документацию от Google. Я ее не очень люблю, потому что там часто бывают устаревшие данные, но меня привлекает краткость, с которой даются те или иные ответы. Честно скажу, что в самом начале обучения мне было сложно ориентироваться в документации.

Полный перечень источников, к которым я обращаюсь в подобных случаях, вы можете найти в главе «Курсы и дополнительные материалы».

Итак, я решил первую часть проблемы: нашел инструмент, который позволит мне отобразить мой список трейлеров на экране. Поэтому теперь я могу перейти к поиску примера, который позволит мне понять, как же этот список внедрить. И тут в самой задаче я опять вижу подсказку: раз я ищу пример, именно так и нужно написать.

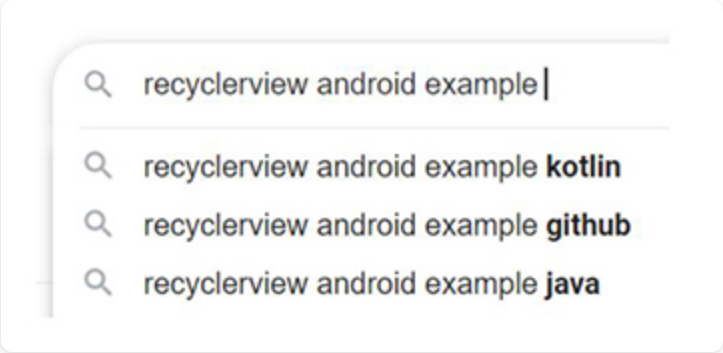


Рис. 20. Запрос для поиска примера

Данный запрос тоже состоит из статических и динамических частей. *«RecyclerView»* (это название инструмента) — динамическая часть, на месте этого слова может быть любая технология. Дальше идет статическая часть из слов *«android example»* (*«Пример Android»*). Так как я работаю на Android, эта связка для меня постоянна. В вашем случае она может выглядеть немного по-другому, в зависимости от того, с какой технологией вы работаете.

Бывает, что после этой основной части нужно добавить уточнения. Допустим, я хочу получить пример, реализованный на языке программирования Kotlin (сейчас используется для разработки на Android), а не на Java (использовался для разработки на Android ранее), поэтому я добавляю слово *«kotlin»* после слова *«example»*. Либо я хочу найти частый пример использования RecyclerView вместе с текстом и картинкой. Поэтому я могу добавить дополнительные слова *«with image and text»* (*«с рисунком и текстом»*), как нам советует поисковик. В итоге формула запроса выглядит следующим образом:

**Android example*

**Пример в Android*

ДИНАМИЧЕСКАЯ* **СТАТИЧЕСКАЯ СТАТИЧЕСКАЯ** *ДИНАМИЧЕСКАЯ

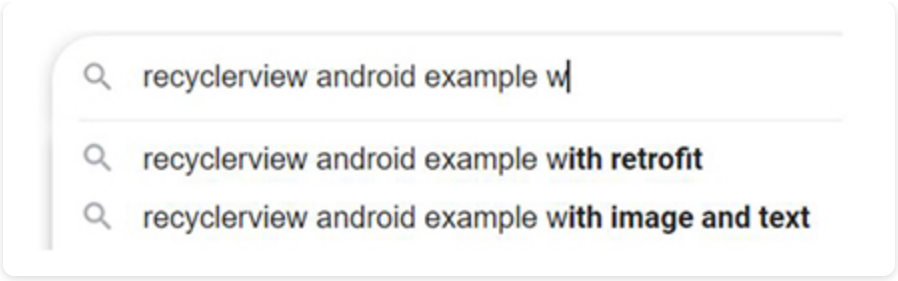


Рис. 21. Формула запроса на поиск примера

Из выдачи я снова выбираю stackoverflow.com, который находится на третьей позиции, и вижу огромное количество примеров (см. рис. 21). Почему я сразу не использую поиск на сайте stackoverflow.com? Просто по привычке ищу все через Google.

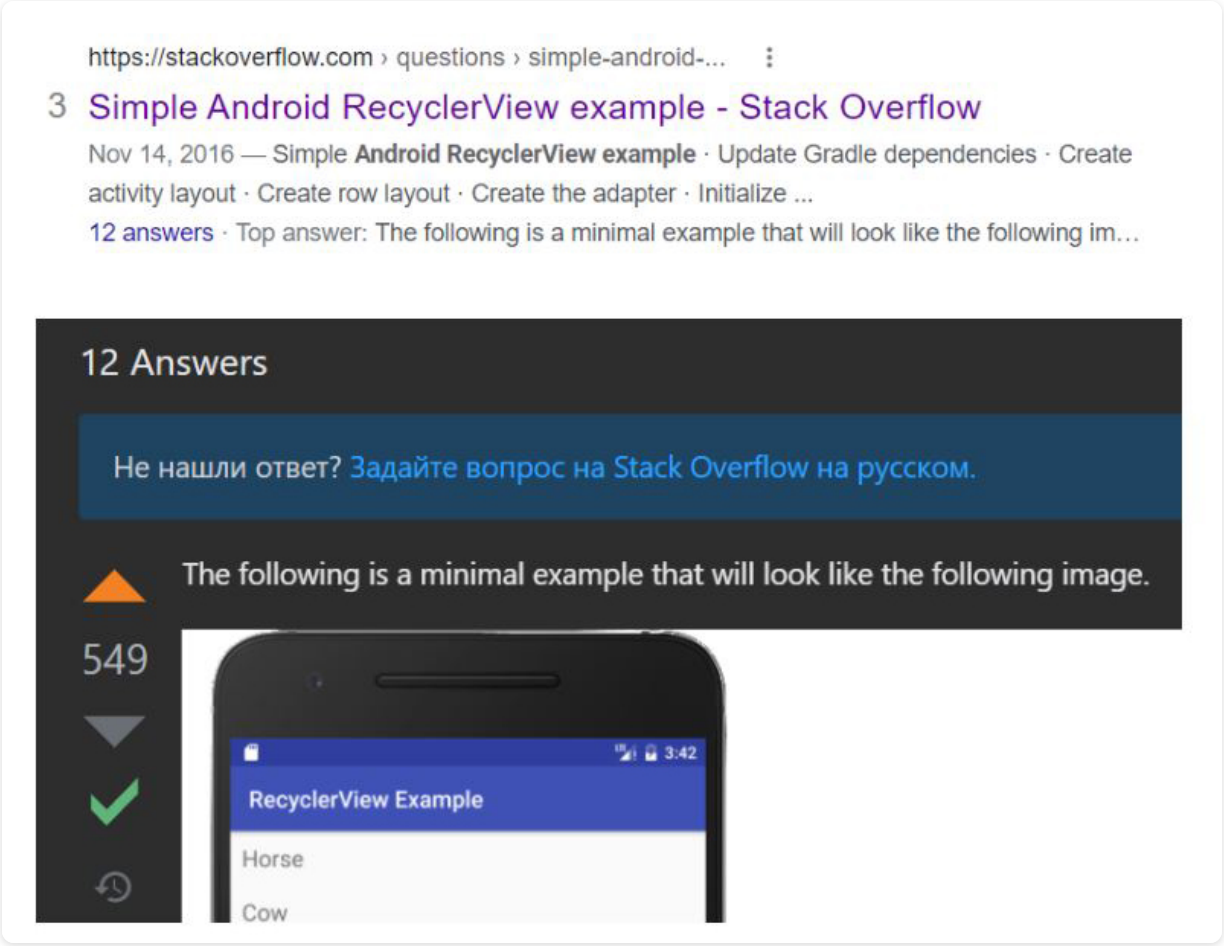


Рис. 22. Ответы на сайте stackoverflow.com

Этот пример с зеленой галочкой, помеченный как правильный ответ, мне кажется привлекательным, потому что содержит довольно детальное описание и хорошо оформлен. Если при внедрении кода что-то не устраивает, то всегда можно посмотреть другие варианты решения по тому же принципу. Их может быть очень много.

Итак, список мы внедрили, но он отображает текст, а нам нужно, чтобы отображалось видео. Вы спросите, а почему я сразу в своем примере не начинаю искать инструмент для встраивания видео, а разбираюсь с текстовым списком. Дело в том, что я предпочитаю внедрять куски кода маленькими порциями, и каждая из них в идеале должна быть настолько простой, насколько это возможно. При таком подходе гораздо легче сначала внедрить очень простой список с текстом, а потом уже адаптировать его под список видео. Кроме того, намного проще сначала внедрить один сложный элемент с простым примером и убедиться, что он работает, а потом внедрить второй, чем одновременно встраивать два сложных и потом искать, что именно сломалось и где закралась ошибка.

Поэтому, успешно внедрив сам список, я перехожу к внедрению видео и возвращаюсь к работе с поисковиком. Ниже будет немного технической терминологии, но если вы ее не поймете, то ничего страшного, главное — понять логику действий.

Свой поиск инструмента я, как и в прошлый раз, начинаю с фразы *«how to»* (см. рис. 23).

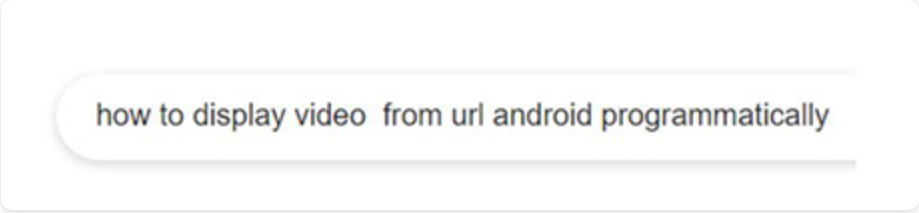


Рис. 23. Запрос на поиск инструмента для встраивания видео

Давайте опять разберем этот запрос по словам. Первая часть, *«how to»*, статична. Следом за ней идет динамическая часть, она описывает суть проблемы, которую должна решать технология. Так вот, в моем случае динамическая часть — это *«display video from url»* (*«отображение видео с URL-адреса»*). Дело в том, что результат с сервера нам приходит в виде ссылки (url), поэтому я использую данное техническое слово в своем запросе. Из собственного опыта могу сказать, что в самом начале, когда еще нет должной технической подготовки, составление динамической части будет занимать много времени, поскольку не совсем понятно, в каком направлении двигаться. Именно эти дополнительные знания я пытался получать, когда ездил за рулем машины, слушая теорию. Я просто включал теоретический материал о том, как это все устроено, узнавал какие-то технические понятия и расширял свой профессиональный лексикон. Поэтому если у вас в самом начале будут проблемы с тем, чтобы корректно составить запрос при поиске решения, это абсолютно нормально, со временем вы сможете это делать намного быстрее, главное — постоянно пополнять копилку своих знаний.

Но вернемся к запросу. Третья часть, которую мы тут видим, выглядит так: *«android programmatically»*. С «Android» мы уже разобрались. Но слово «programmatically» у нас ранее не фигурировало, это что-то новое. Дело в том, что иногда поисковик при запросе, если в нем указано слово «Android», вместо примера кода может выдать примеры сервисов из PlayMarket — в нашем случае он может выдать приложения, которые отображают видео. Слово «programmatically» я часто использую при формулировании запросов, так что можно считать его статичным. В переводе это слово означает «программно», то есть с его помощью мы хотим узнать, как отобразить видео с помощью программного кода. Впрочем, даже задав достаточно точные параметры, я могу

встретить примеры, которые не будут нам подходить, поэтому нужно будет отдельно разбираться в каждом из них.

Не думаю, что следует детально описывать каждый этап поиска. Ведь моя задача — показать алгоритм действий, который повторяется снова и снова, а не научить вас внедрять сам список. Поэтому могу только сказать, что дальше я делаю абсолютно те же самые шаги, то есть ищу примеры кода на stackoveflow.com, внедряю их, возможно, сталкиваюсь с новыми проблемами, которые, в свою очередь, будут требовать решения. И так будет происходить до того момента, пока проблема не будет решена полностью и я не получу работающий список с отображаемыми видеороликами. Так и выглядит цикл написания кода: ищем решение, внедряем, настраиваем и чиним, а затем заново. Если говорить о моем примере, то после всех этих цикличных манипуляций с поисковиком и внедрения RecyclerView остается внедрить в него второй инструмент, Exoplayer, который будет показывать трейлеры на экране.

Подводя итог данной теме, можно сделать несколько выводов.

- Процесс решения той или иной задачи состоит из двух частей: поиска инструмента и поиска примера того, как настроить этот инструмент.
- Каждый запрос состоит из статической части и из динамической. Для каждой технологии будут свои комбинации, но если вы хорошо изучите мои примеры, вам будет от чего отталкиваться.
- Важно обращать внимание на уточнения в запросе. Это могут быть технологии, названия библиотек, используемый язык программирования, операционная система или все что угодно. Главное — чувствовать баланс, потому что если сильно

усложнить запрос, то увеличится вероятность того, что информация не будет найдена.
- Поиск решений — это цикличное действие. Вы постоянно делаете одно и то же с небольшими вариациями. Это часть работы программиста.
- Быстрый поиск информации — это навык, нарабатываемый с опытом. Свыкнитесь с тем, что в самом начале этот процесс может проходить медленно. Я совершал ошибки, читал длинные статьи, в которых не было ответа на вопрос. Бывало и такое, что я тратил много времени на внедрение инструмента, а в итоге оказывалось, что он не подходит. И в результате мне приходилось все переделывать и искать заново. Да, и такое тоже случается, даже с профессионалами. Целая команда полгода внедряет технологию, а потом оказывается, что это не то, что нужно — время и деньги были потрачены зря.

Поиск решений — это очень ценный навык, который по важности не уступает навыку написания самого кода. Поэтому старайтесь прокачивать его с таким же усердием, как и сам язык программирования.

Важность поддержки со стороны семьи

Но давайте вернемся к моей истории. Прошло около четырех месяцев. Жизнь шла своим чередом. С течением времени изучение программирования по вечерам стало чем-то самим собой разумеющимся. Конечно, новый порядок вещей не мог не отразиться на моей семейной жизни и не повлиять на отношения с супругой. К счастью, она восприняла мою цель так же серьезно, как и я. Она знала, что если я решил чего-то добиться, у меня хватит сил и дисциплины довести начатое до конца. И эта вера очень много значила для меня, потому что мне не пришлось тратить силы и энергию на убеждения, что это нам действительно нужно и что это того стоит. Она просто поверила в меня. Поэтому в том, что я достиг своей цели, есть и ее немалая заслуга. Она забирала ребенка из комнаты и полностью занималась им, чтобы я мог учиться. Вечерами я без каких-либо претензий с ее стороны мог спокойно сидеть за компьютером, погружаясь в процесс обучения и не отвлекаясь на внешние факторы. А когда возникали сложности, я ни разу не слышал упреков или сомнений в том, что моя цель того не стоит или что я не смогу дойти до конца. Напротив, я получал огромную поддержку. Моя жена напоминала, как много уже пройдено, и говорила, что назад поворачивать уже поздно и нужно идти до конца.

Собственно говоря, едва решившись изучать программирование, я сразу же обсудил это со своими близкими. Я сделал акцент на двух основных вещах. Первым делом я перечислил все плюсы, которые дает программирование: то, что эта профессия позволит всегда быть востребованным во всем мире и работать удаленно на компании в

любой точке планеты. И конечно же, упомянул о том, что программистам хорошо платят, причем часто в валюте, и есть отличные перспективы роста.

Поговорив о плюсах, я сразу обозначил, что на все это потребуется время, что придется заниматься по вечерам и по выходным. Мне будут нужны уединение и покой для полного погружения в процесс.

Обсудив плюсы и минусы, я прямо сказал, что мы должны принять это решение вместе, потому что оно повлияет на нашу общую жизнь, ведь мы живем под одной крышей. И можно сказать, что решение идти по этой долгой дороге было коллективным. Этот фактор тоже очень сильно повлиял на итоговый результат, потому что я не чувствовал давления и был расслаблен психологически.

Поэтому перед тем, как начать этот путь, я рекомендую поговорить со своими близкими и узнать их мнение на этот счет. А я и моя семья, после того как наше путешествие началось, поставили мое программирование в общий приоритет, и об этом я хочу поговорить отдельно.

Как поставить программирование в приоритет

Как бы банально это ни звучало, но чтобы дойти до результата, нужно идти. Если нет сил идти, нужно ползти. Если не можешь ползти — нужно лечь и лежать в нужном направлении. Поэтому я все время двигался в сторону цели. Представьте себе, что вы птица, которая летит на далекое расстояние. Движения крыльев помогают птице достичь цели. Когда она перестает махать крыльями, ее скорость начинает снижаться, и она планирует вниз либо вообще может упасть и разбиться. Если птица летит над землей, у нее есть возможность сесть на землю для отдыха и потом лететь дальше. Но если птица пересекает море или океан, то ей нужно лететь до конца.

Так вот, представьте, что вы птица, которая летит очень далеко. И как только вы перестаете махать крыльями и снижаете скорость, вы подвергаете себя риску. Как только вы остановились и приземлились, вас может убить охотник, на вас может напасть хищник либо вы можете утонуть в воде. В моем случае, как только я переставал двигаться вперед, я опасался, что мне будет лень вернуться назад в полет. Сразу начинали появляться различные причины, чтобы откладывать обучение на потом. А если отодвинуть обучение на долгое время, желание лететь дальше пропадет вовсе. Поэтому надо постоянно двигаться вперед. Нельзя останавливаться. Ведь чем быстрее и интенсивнее вы будете лететь к цели, тем быстрее вы до нее доберетесь.

Во время моего полета меня накрывали те же заботы, что и вас, тех, кто только готовится встать на этот путь или уже идет по нему. У меня уже был маленький ребенок, и хотя жена взяла на себя

большую часть забот о нем, мне все равно нередко приходилось отложить все дела и идти ей на помощь. Мой срок обучения программированию составил два года, и все это время я работал полный рабочий день. Был даже период, когда я работал на двух работах одновременно. Помимо этого, я еще успевал регулярно заниматься спортом — я ведь понимаю, что без здоровья все равно никаких целей не достигнешь. Поэтому мне тоже всегда не хватало времени, я постоянно был уставшим и хотел отдохнуть. Так что никакой волшебной пилюли тут нет, но я для себя выработал несколько принципов, которые позволили мне продвигаться вперед без остановок.

Первый из них заключается в том, что нужно задействовать для учебы те инструменты, которые в данный момент под рукой. Сейчас мы имеем в доступе огромное количество форматов обучения. Это различные видеоуроки, аудиоподкасты, обучающие мобильные приложения и даже приложения, которые обучают в игровой форме. Более подробно о том, как я разбивал свое время с использованием каждого из этих форматов, я расскажу чуть позже (см. главу «Контроль времени»). Но суть, думаю, ясна: если было удобно смотреть, я смотрел, если было удобно только слушать, я слушал. Любая информация из сферы IT была полезной, даже общая, та, которая не связана напрямую с кодом.

И второй важный фактор, который влиял на то, «долечу» я до конца или нет, — это все-таки сила воли, дисциплина и мотивация. Без твердого намерения достичь своей цели у меня ничего не вышло бы. Если птица живет в теплых странах, где всегда хороший климат, то какой смысл лететь куда-то далеко? Ее и так все устраивает. Но если птица знает, что через месяц в ее местности наступят такие холода, что она не сможет найти себе пропитание, и это приведет ее к смерти, то она полетит в теплые края, несмотря ни на что.

Читая истории подписчиков, разговаривая со своими знакомыми, да даже глядя на самого себя, я сделал вывод, что люди тоже делятся на два таких типа. Вопрос только в том, к какому типу относитесь вы. И нужно быть откровенным с собой: мы можем жаловаться на то, что нас не устраивает качество жизни, не устраивает страна, в которой мы живем, не устраивает работа, на которую мы ходим. И говорить можно все что угодно, но если вы не ищете теплое место, значит, вас все устраивает. Человека определяют поступки, а не слова.

Благодаря осознанию этих двух моментов я смог двигаться вперед. Я полностью оптимизировал свое время и поглощал материалы всеми возможными способами. Я постоянно видел себя со стороны и непрестанно задавал себе вопрос: а что я делаю сейчас, чтобы «лететь» в сторону своей цели? И это мне очень помогло.

Горечь от неоправданных ожиданий

Прошло шесть месяцев. Этот срок был для меня очень значимым. В каком-то смысле он означал своего рода промежуточный финиш для меня и моих родных. Почему именно шесть месяцев? Потому что в рекламе курса, который я покупал на сайте udacity.com, говорилось, что средняя продолжительность его прохождения составит полгода, и полученных знаний будет достаточно, чтобы приступить к поиску работы. Именно столько времени у меня и ушло на освоение курса, учитывая график и наличие основной работы. Я окончил курс Basics of Android Development и был готов начать зарабатывать.

Я начал с того, что собрал три разработанных за время обучения приложения и разместил их на Play Store, после чего перешел к составлению резюме. Это было очень тяжело, потому что приходилось буквально по крупицам вспоминать и собирать технический опыт и навыки, полученные на прошлых работах. В результате у меня получилось очень слабое резюме, в которое я сам верил с трудом.

Еще одним важным пунктом на тот момент было то, что я совершенно не мог измерить уровень моих знаний в программировании. Я учился в одиночестве, поэтому мне не с кем было сравнить свои навыки, чтобы понять, насколько они хороши. Поэтому я просто доверился лозунгу курса и начал искать работу программистом на позицию Junior, то есть на самый начальный уровень. Но я не получил ни одного звонка от рекрутеров. Все это происходило еще в доковидные времена, поэтому большая часть вакансий требовала присутствия в офисе на полную рабочую

неделю. А из-за того что я жил в небольшом городе, количество предложений было крайне ограничено. Кроме того, моя специализация (Android-разработка) сама по себе очень специфическая, и таких вакансий было еще меньше. И наконец, самое главное: я понял, что тех знаний, которые я получил на курсе, было в действительности совершенно недостаточно, чтобы начать работать программистом.

Это стало для меня настоящим шоком. Я ждал звонков целую неделю, но никто так и не позвонил. Я впервые столкнулся с по-настоящему горьким чувством неоправданных ожиданий. Разница между тем, что обещали в рекламе курсов, и реальностью поразила меня. Я думал, что всего за шесть месяцев смогу изменить свою жизнь и стать программистом практически без опыта и технических знаний.

Тогда я понял, что мне предстоит принять еще одно важное решение: либо остановиться на том, что есть, либо продолжать двигаться вперед. Конечно, после такого разочарования я уже не верил ни в какие сроки и лозунги, в моем туннеле обучения как будто погас свет, и я не видел, куда идти. Было ощущение, что нужно просто двигаться вперед вслепую, не зная, насколько длинным может оказаться этот туннель.

Теперь я также осознал, что не знаю, сколько точно времени мне потребуется для достижения цели. Я больше не мог доверять рекламным лозунгам и обещаниям. Но с другой стороны, я понял, что часть пути пройдена, и теперь нужно либо идти до конца, сколько бы времени это ни заняло, либо останавливаться на том, что есть, сделать выводы из полученного опыта и жить дальше.

Оказавшись перед таким сложным выбором, я понимал, что теперь у меня нет никакой, даже примерной картины по поводу сроков достижения результата. Также я осознавал, что вообще не имею

никаких гарантий, что смогу осилить этот путь. Но думаю, что вы, держа сейчас в руках эту книгу, уже поняли, какой выбор я в итоге сделал. Поворачивать обратно я не хотел, поэтому продолжил свою дорогу и вступил в темноту, несмотря на разочарование от того, что мои надежды не оправдались.

После этого разочарования я вывел для себя одно правило, которое использую и по сей день и которое помогает теперь мне во всех моих начинаниях. Обычно, когда я что-то задумываю или планирую, у меня в голове есть три основных расклада:

- негативный — прогнозы не совпадают с реальностью в худшую сторону;
- нормальный — прогнозы совпадают с реальностью;
- позитивный — реальность превосходит прогнозы и ожидания.

Так вот, я первым делом прокручиваю свои эмоции и представляю, какие ощущения я испытаю в случае негативного сценария. Я пытаюсь понять, какие полезные уроки я смогу извлечь из ситуации, если все пойдет не по плану. И потом решаю, устраивает ли вообще меня негативный сценарий либо стоит прекратить двигаться к цели и остановиться на том, что есть. При таком подходе я морально готов к тому, что все пойдет не так, как я хочу. Благодаря этому, если случается что-то непредвиденное, то оно не выбьет меня из колеи. Я смогу справиться со сложностями и пойти дальше.

В целом, если рассматривать именно программирование, то я могу твердо сказать, что никаких розовых очков и ожидания того, что это будет просто, у вас не должно быть. Обучение IT-профессии — это изначально сложная задача. Поэтому я советую сразу настраиваться на тяжелую работу. Но тем не менее я уверен, что она под силу абсолютно любому человеку. Абсолютно любому! Я сам прошел этот путь, поэтому могу оценить его сложность и поделиться с вами

несколькими важными моментами, с которыми вы столкнетесь в процессе обучения.

И напоследок несколько выводов.

- Учить технологию — это сложный процесс. Сама перестройка мозга на «технические рельсы» потребует много времени и усилий. Чем меньше было опыта до этого момента и чем больше ваш возраст, тем труднее будет перестроиться. В самом начале, при появлении первых сложностей, у вас могут возникнуть отторжение и желание бросить это дело, но нужно все равно продолжать учиться, и со временем ваш мозг привыкнет.
- В среднем на обучение уходит от 6 до 24 месяцев, и это немаленький срок. Когда-то я думал, что нет ничего сложного в том, чтобы год-другой потратить на обучение. На словах это легко. Но по ощущениям, изучение технических материалов без какого-либо опыта и материального вознаграждения — это сложно. Необходимость тратить свое свободное время, когда ты и так устал после работы и вынужден делать то, что не приносит удовольствия, — это очень сложно. Делать это на протяжении продолжительного времени, когда есть возможность бросить все и никто не будет против, — это ужасно сложно. Мне порой казалось, что это бесконечная история, у которой нет завершения.
- Времени на обучение всегда будет не хватать, вне зависимости от вашего статуса и состояния. Если вы меняете профессию, значит, вы сможете учиться только в свободные от работы часы. А они будут всегда забиты какими-то заботами, делами или развлечениями. И вставить в этот маленький промежуток своей жизни изучение чего-то сложного, что поначалу вообще может не доставлять удовольствия, кажется невыполнимой задачей. Поэтому вы должны понять, что времени не хватает всем, и так будет всегда!

Быть может, такой подход и привычка заранее готовить себя к негативному сценарию покажутся вам чересчур пессимистичными, но на самом деле, проговаривая это, я помогаю сразу двум типам людей. Кто-то благодаря этим строкам сразу бросит попытки обучиться программированию и тем самым сэкономит себе нервы, время и деньги. Ведь вместо этого он сможет заняться чем-то другим или просто продолжить наслаждаться жизнью. Тем же, кто решит, несмотря ни на что, продолжить свой путь, теперь будет легче представить себе реальную картину и выставить корректную планку ожиданий. И тогда, столкнувшись с трудностями, вы морально уже будете к ним готовы.

Что касается меня, то, как я уже говорил, начиная путь, я понимаю, что впереди меня могут ждать проблемы, и заранее продумываю, что я буду делать при таком раскладе. Когда ты готов упасть, падать не так больно и не так страшно. На пути изменения жизни по-другому быть не может. Падения будут всегда, но если настроить себя на это заранее, в нужный момент у вас хватит сил встать и продолжить путь. Поэтому снимите розовые очки от рекламы, наденьте шлем с налокотниками и отправляйтесь в путь.

Разные дороги — разные результаты

Рис. 24. Две дороги

Говоря о жизненном пути, я всегда представляю себе две дороги. Одна из них, сложная и полная возможных опасностей, ведет через лес, а вторая, ровная и красивая, — через равнину. И по сути, каждый человек выбирает, по какой дороге ему идти. По лесной дороге идут люди, у которых есть желание изменить свою жизнь. Засыпая каждый вечер, они мысленно видят перед глазами свою цель. Они представляют, как она реализовывается, и, просыпаясь, делают все, чтобы достичь результата. Эта дорога опасная и сложная, люди часто получают травмы, выбирая путь через лес.

Дорогу через равнину выбирают люди, которых все устраивает. Либо у них тоже есть мечта, но они ничего не делают для ее реализации.

В социальных сетях можно встретить огромное количество мотивационных роликов и пабликов, которые категорично утверждают, что нужно всегда двигаться к цели, быть совершеннее, жить лучше, зарабатывать больше. Но я считаю, что такой образ жизни подходит не всем. Нет единственно правильного выбора.

Каждый индивидуален, и хорошим выбором будет тот, с которым комфортно жить конкретному человеку.

Но как бы то ни было, смена профессии — это чаще всего дорога через лес. Поэтому в начале пути задайте себе этот вопрос и ответьте честно, готовы ли вы идти через лес или вам и так комфортно двигаться по асфальтированной дороге и просто наслаждаться жизнью. Лично я всегда предпочитаю идти через лес. Но это мой осознанный выбор, он не навязан извне и не продиктован царящим в обществе культом успешности. Я такой от природы. Хотя я и привык постоянно двигаться по тяжелой дороге, наполненной вызовами и препятствиями, и совершать огромное количество ошибок на своем пути, их, тем не менее, не становится меньше. В частности, решившись на смену профессии и начав свой путь программиста, я допустил ту ошибку, которую совершают множество людей (как я понял из общения с моими учениками и подписчиками). Я думал только о начале пути и представлял конечный результат, но не думал о том, как же я буду идти по самой дороге. Теперь же, все-таки преодолев этот путь и оборачиваясь назад, я могу выделить одну главную идею, которая позволила мне дойти до конца. Речь идет о том, чтобы постоянно задавать себе вопрос: двигаюсь ли я вперед или стою на месте?

Я помню эту начальную эйфорию, которая позволяет сделать первый шаг. После того как она заканчивается, скорость движения неизбежно начинает снижаться, порой до нуля. И это может стать настоящей ловушкой. Я помню, что мысленно я находился в пути. Мне казалось, что я продолжаю обучение, но по факту ничего не делал, просто стоял на месте без какого-либо прогресса: откладывал занятия по причине усталости, оправдывал свое бездействие более важными делами и другими причинами, даже порой думал о том, чтобы сойти с дистанции. Но потом я решил делать каждый день хотя бы по одному маленькому шагу, лишь бы постепенно идти

вперед. Такой подход имеет ряд преимуществ. Во-первых, медленное движение лучше, чем отсутствие движения. Поэтому к цели мы идем в любом случае. Во-вторых, при невысоком темпе меньше устаешь и меньше риск выдохнуться и вообще прекратить движение. Мы все знаем случаи, когда начинаешь какое-то дело, впахиваешь по двадцать часов в день в течение трех месяцев, потом выгораешь и перестаешь что-либо делать. А здесь я двигался медленно, не задыхаясь от высокой скорости и нагрузки.

Тем не менее у этого подхода есть и большой минус. При маленькой скорости время прохождения пути очень сильно увеличивается, и можно перегореть. Поэтому идти придется долго, и мотивации потребуется больше.

Я говорю это к тому, что если вы выбрали дорогу через лес, будьте готовы идти долго, время от времени испытывая проблемы с мотивацией, с пониманием материала, со временем. И все проблемы, перечисленные мной выше, — тоже часть пути. И решение этих проблем позволяет привыкнуть к динамичному темпу, научиться легче относиться к сложностям и принять, что навык решения проблем так же важен, как навык написания кода. Потому что путь к достижению цели — это не стометровка, это марафон, порой очень долгий и кажущийся бесконечным. Но если двигаться вперед, то вы точно увидите красную ленточку с надписью «Финиш».

Если же вы решите пойти по второму пути и выбрать дорогу по равнине, то, повторюсь, в этом нет ничего плохого. Если вам так комфортно, то пусть так и будет. Наслаждайтесь тем, что есть, и не надо себя терзать словами «Я должен». Вы никому ничего не должны. Не следует начинать свой путь из соображений «надо».

Но давайте абстрагируемся от моего пути и перейдем к вещам более конкретным. Поговорим о том, сколько же времени в среднем может занять обучение программированию. Я видел много историй, где

люди входят в IT и становятся программистами очень быстро. Знаю человека, у которого это заняло всего шесть месяцев упорного труда. Его зовут Михаил, он живет в Канаде, и он стал web-разработчиком в тридцать пять лет, самостоятельно изучая данное направление. Мое интервью с ним, где он в деталях рассказывает о своем пути входа в IT, вы можете посмотреть у меня на YouTube-канале. Я таким быстрым результатом похвастаться не могу. Мой путь занял намного больше времени: с начала учебы до моего первого реального проекта прошло примерно два года.

Таким образом, четкого и однозначного ответа на вопрос, сколько времени занимает получение профессии программиста, не существует, но я постараюсь дать хоть какую-то информацию, чтобы у вас сложилась примерная картина.

Первый фактор, который напрямую влияет на достижение результата, — это выбранная сфера. Сравнивать мы будем одно из самых популярных направлений (web-разработку) и выбранное мной (разработку приложений под мобильные телефоны с операционной системой Android).

Хочу обратить ваше внимание, что сравнение является полностью субъективным и основано преимущественно на моем личном опыте, на опыте людей, с которыми я общался, а также на информации, находящейся в открытом доступе в интернете. Ваше мнение может не совпадать с моим.

После того как я уже поработал Android-разработчиком, спустя какое-то время я получил второе образование по специальности «разработчик программного обеспечения» и имел возможность поучиться web-разработке. По моим личным ощущениям, на самом старте web-разработка показалась мне более понятной. Допускаю, что на тот момент я не мог объективно оценивать сложность

обучения, потому что уже был профессиональным разработчиком. Но сам путь обучения и концепция показались мне более простыми.

Хочу подчеркнуть: я не утверждаю, что сама web-разработка проще, чем разработка на Android, но сам старт и начальная информация, с которой придется иметь дело новичку, по моему мнению, воспринимаются намного легче.

Огромным плюсом web-разработки является то, что минимальные знания, необходимые для старта работы в данном направлении, более просты для понимания, нежели стартовые знания для того же Android. Да и сам объем этих знаний тоже очень сильно различается.

Помимо этого, само обучение web-технологиям проходит более плавно, с логичными переходами от одного к другому. Это похоже на то, как вы потихоньку заходите в море, где уровень воды постепенно становится все выше и выше.

В Android же все начинается с «прыжка в воду», причем ее уровень уже в самом начале находится где-то в районе пояса. То есть в Android мне сами технологии показались более сложными, и минимальный уровень понимания этих технологий для комфортной работы и обучения потребовался более высокий. Если в web-разработке я мог постепенно изучать одну технологию за другой, то в Android-разработке мне нужно было параллельно изучать несколько направлений, которые тесно связаны между собой.

Чтобы было понятнее, что я имею в виду, я попытался оценить уровень сложности конкретной технологии при изучении новичком, присвоив каждой технологии определенный уровень сложности и сравнив их между собой, чтобы вывести общий средний показатель (см. рис. 25).

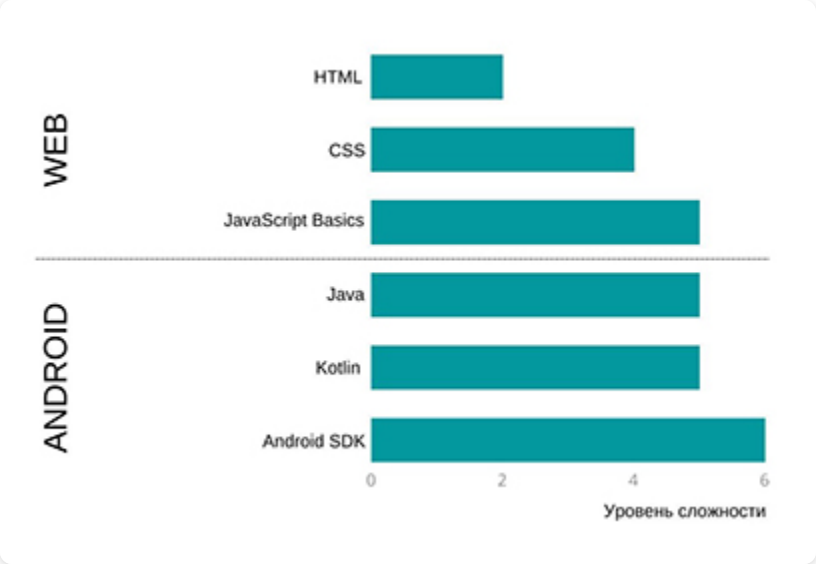


Рис. 25. Сравнение технологий

Первые три пункта на графике сверху — это технологии, которые используются при изучении web-направления и с которыми придется разбираться новичку. Рассмотрим их подробнее. Это:

- построение web-страницы с помощью HTML,
- оформление с помощью CSS,
- программирование с помощью начального уровня JavaScript. С помощью последнего мы как бы вдыхаем жизнь в web-страницу, внедряя в нее логику работы различных элементов, таких как меню, кнопки и т. п.

Так вот, при изучении web-разработки эти технологии вы будете осваивать последовательно, одну за другой.

Последние три технологии в списке (Kotlin, Java, Android) изучаются разработчиками Android. Им предстоит разбираться в языке программирования Java, потому что изначально все приложения писались именно на нем. Но в мае 2019 года, как я уже говорил ранее, официальным языком программирования Android был выбран Kotlin, поэтому в первую очередь вам придется изучать его,

а Java — разве что для общего развития. Помимо этого, требуется еще освоить инструменты для разработки на Android. И самое главное, все эти технологии вы будете изучать параллельно, а не одну за другой, как мы это делаем в web-разработке.

Давайте посмотрим на график и сравним значения сложности, которые я присвоил каждой технологии. В web-направлении получается, что HTML имеет два балла, CSS — четыре, JavaScript — пять. В направлении Android технологии имеют следующие уровни сложности: Android — шестой, Java — пятый, Kotlin — пятый. Таким образом, сложив основные технологии web-разработки (HTML + CSS + JavaScript), мы получим одиннадцать баллов. При сложении технологий Android (Android + Java + Kotlin) мы получим шестнадцать баллов.

То есть мы видим, что направление разработки на Android в целом выглядит более сложным для новичка, а значит, достижение минимального уровня, необходимого для работы над проектом, потребует больше времени. Поэтому, подводя итог, я хотел бы сказать, что выбор направления очень сильно влияет на то, насколько быстро вы будете продвигаться вперед и насколько оперативно сможете приступить к поиску первой работы. Заметьте, я не берусь прогнозировать, насколько быстро вы получите свою первую работу, потому что чем ниже сложность, тем выше конкуренция. К тому же, если вы выбираете более сложное направление, то мотивации для продвижения вперед требуется больше. Все это крепко взаимосвязано, поэтому я смотрел на процесс смены профессии со всех углов, и выбор направления — это один из важнейших моментов, на который я обращал внимание.

Еще один важный фактор, влияющий на скорость перехода в программисты, — стартовые условия. Каждый человек обладает своим уникальным опытом, в том числе опытом работы с

технологиями. Если мы, например, возьмем двух продавцов одежды, один из которых в свободное время любит играть на компьютере, а второй в основном увлекается спортом и туризмом, то первый благодаря своему уверенному владению компьютером все равно будет иметь преимущество, даже если вся его предыдущая деятельность не относилась к программированию. Потому что его навык работы с компьютером более развит, и даже такая мелочь, как набор текста вслепую, не глядя на клавиатуру, может ускорить процесс обучения. Это индивидуальные факторы, которые тоже напрямую влияют на скорость получения результата.

Итак, выбирая направление, нужно просто думать головой, не покупаться на рекламу и понимать, что есть разные уровни сложности. Но в то же время, если где-то проще, значит, там и конкуренция выше, и оплата может быть меньше. Везде есть и плюсы, и минусы. И только вы сможете определить, какой путь подходит именно вам.

Контроль времени. Как успевать работать и учиться?

И снова вернемся к моей истории. Подходил к концу первый год моего обучения, и мой типичный день в ту пору выглядел примерно так.

Шесть утра, пора просыпаться. Я тихо встаю, умываюсь и пытаюсь уйти, никого не разбудив. Со стола хватаю подготовленный с вечера пакет с едой. В то время я не завтракал дома, потому что, во-первых, таким образом я экономил время на сборы и мог поспать лишние несколько минут, а во-вторых, не будил своих родных, ведь жили мы в маленькой квартире, где все сразу проснутся, если ты начнешь расхаживать туда-сюда.

Вот я сижу в машине и еду на одну из своих двух работ. Правой рукой держу бутерброд, левой — руль. В глаза бьет солнце, которое встает над горизонтом. В телефоне звучит какой-то ролик, один из блогеров рассказывает про разные специальности в IT.

Я приезжаю на место, и начинается мой рабочий день, который длится с семи утра до трех часов дня. После этого я еду на вторую работу. До нее всего минут двадцать пути, а начинается она в четыре. Поэтому у меня остается примерно тридцать свободных минут. Я нахожу место для обеда, достаю вторую порцию своей еды и включаю YouTube. Я смотрю на то, как человек объясняет работу с базами данных. Ну вот, обед закончился, пора на работу, которая продлится до девяти часов вечера. Потом домой.

На пути домой чувствую себя уставшим, хочется побыстрее добраться до семьи, горячего душа и ужина. Когда я уходил, все

спали, и солнце только начинало вставать, а когда прихожу, за окном уже темно, а мой маленький сын готовится ко сну.

Наконец-то я дома. Общаюсь с семьей, вместе ужинаем, и вот уже одиннадцатый час ночи. Я сажусь на диван, кладу на колени ноутбук и начинаю изучать программирование. Время за компьютером пролетает незаметно. Я смотрю на часы, на которых уже далеко за полночь. Глаза слипаются, и я понимаю, что мой мозг уже не соображает и не схватывает информацию. Значит, пора спать. На следующий день все сначала.

Я описал один из дней того периода, когда я учился и работал на двух работах одновременно. За два года, пока я обучался, я поменял несколько мест работы, и мой рабочий график тоже, соответственно, менялся. Неизменным оставалось только одно: отсутствие свободного времени. Я помню, что по какому бы графику я ни работал, мне всегда не хватало времени. Однако изучение программирования всегда стояло в приоритете, и я всегда мог находить свободные часы, чтобы учиться. Помимо этого, я постоянно пытался оптимизировать управление своим временем, и это позволяло мне при большой загруженности все равно находить время на учебу. Об этом и хотелось бы поговорить подробнее.

В общей сложности я потратил на обучение примерно два года. И если сравнивать мою историю с историями других людей без опыта, входящих в IT, то это довольно много. Я разговаривал с ребятами, которые становились программистами за шесть, восемь, одиннадцать месяцев. Одна девушка, молодая мама, стала тестировщицей с нуля всего за три месяца. Интервью с ними вы можете посмотреть у меня на YouTube-канале. То есть в плане скорости меня вряд ли можно назвать образцом для подражания.

Но я не могу сказать, что мой путь оказался таким долгим по причине моей личной неорганизованности. Действительно, довольно

много времени ушло на изучение и понимание языка Java и основных принципов работы на платформе Android. То есть сам материал для меня оказался сложным. Но что касается всего процесса обучения, то мне удалось его оптимизировать, и в первую очередь правильно организовать свое время.

Поначалу я часто переключался на какие-то мелкие дела, которые отвлекали меня от учебы. Я говорил себе: «Вот сейчас быстро отвечу на это письмо», «В этом видео про новинки компьютерных игр всего десять минут, я быстро его посмотрю», «Я всего пять минут полистаю соцсети, ведь я же устал. Весь день работал. Надо себе давать время и на отдых». Получается, я постоянно искал себе оправдания в виде других дел, чтобы отложить процесс учебы.

Но самое ужасное, что потом, при отсутствии прогресса, я искал причину во внешних обстоятельствах, которые не мог контролировать. То есть пытался переложить ответственность за отсутствие результата на эти обстоятельства и дела, вместо того чтобы взять вину на себя.

Когда я осознал это, я испытал смешанные чувства. Меня накрыли мысли, что материал слишком сложный и я не могу его осилить, поэтому избегаю учебы, выдумывая себе дела. Да, никто не любит горькую правду, но если не признавать ее, ситуация не изменится. Если бы я не отвлекался на все эти мелочи, я был бы уже, условно говоря, на третьем уроке, но я отвлекался — и поэтому я до сих пор на первом.

И это очень важная мысль, я прошу прочитать ее более вдумчиво. Причины, по которым вы не добиваетесь своих целей, абсолютно не важны. Важен только результат. И если вы не достигаете своих промежуточных целей (не заканчиваете уроки, не доделываете приложения, не дочитываете книги), то и к главной цели вы не

приблизитесь. Осознание этого — главный признак, что пора что-то менять.

И первое, что нужно сделать, — успокоиться и принять тот факт, что это нормально — не понимать сложный материал. Наш мозг устроен таким образом, что если ему что-то не нравится и что-то не получается, он пытается это прекратить. И когда мы слабо контролируем процесс, долго не можем понять сложную тему, то мозгу в этой ситуации становится очень некомфортно. Он будет перегреваться, и обучение не принесет никакого удовольствия.

Какой выход я нашел для себя из этой ситуации? Я решил переосмыслить и поменять основную цель, которую ранее планировал достигать ежедневно. Я попробовал отпустить ситуацию и не зацикливаться на конкретных результатах. Если основной целью станет просто заниматься в течение одного часа, а не добиться какого-то результата, то мозг расслабится, и информация начнет усваиваться гораздо эффективнее. Скажу по собственному опыту, что, выбрав такой подход, я иногда настолько погружался в процесс, что не замечал, как пролетает этот час, и часто делал даже больше. Но я понимал, что должен провести этот час действительно качественно, то есть отложить мобильный телефон, выключить видео и полностью погрузиться в программирование.

Это помогло, но, конечно, этого было недостаточно для того, чтобы оптимизировать весь процесс и научиться правильно распределять время. В какой-то момент я понял, что пора предпринимать более серьезные и системные шаги. Еще до смены своего профессионального пути я читал много литературы по тайм-менеджменту, а также смотрел различные обучающие видео. В каждой системе было что-то, что мне подходило, а что категорически не устраивало. Я пробовал разные приемы и методы; некоторые корректировал и успешно внедрял, а некоторые отметал.

В итоге у меня сформировался собственный подход к управлению временем, который я и по сей день использую в своих делах. Я говорю о нем и на своем YouTube-канале, поэтому если вам более удобен видеоформат, то можете ознакомиться с ним там.

Расскажу, как действовал я, помимо того, что стал более тщательно следить за своей дисциплиной и пресекать влияние всех отвлекающих факторов. Я решил, что для начала было бы неплохо измерить общее количество часов, которое я трачу как на изучение программирования, так и на то, что к нему не относится. Для этого я разделил все свое время на две группы. Первая группа — время, потраченное исключительно на написание кода. Вторая — все остальное. Почему я решил отдельно назвать именно время, потраченное на написание кода? Потому что это реальная практика, то есть именно то, что прокачивает мой навык, который позволит мне устроиться на работу программистом. Где-то на просторах интернета мне однажды встретилась фраза: «Чтобы научиться программировать, нужно много программировать». Вот почему я поставил себе цель тратить на написание кода ровно один час в день. Эту цель легко измерить и понять, достиг ты ее или нет. Благодаря этому я почувствовал, что вернул себе контроль над процессом. Я уже не пытался быстрее двигаться по курсу обучения, но в моих силах было заниматься программированием один час в день, поэтому на ней я и сфокусировался. Сейчас самое главное — понять, что я добился результата и прогресса, потому что начал оценивать количество времени, проведенное за написанием кода, вместо того чтобы оценивать количество пройденных уроков. И в итоге не только смог планомерно прокачивать свой навык, но и получил бонус в виде контроля над ситуацией, который дал мне много сил и энергии продолжать двигаться вперед.

Однако я понял, что просто измерять затраченное время в часах — это не совсем точный метод, и пришла пора его усовершенствовать.

Все мы понимаем, что один час перед работой и один час в конце рабочего дня, когда ты устал, по ощущениям, проходят совершенно по-разному. Поэтому для себя я ввел еще один показатель — уровень продуктивности. Это наше состояние на текущий момент, то есть то, насколько мы способны расходовать наше время с пользой для достижения цели. Я измеряю этот уровень показателями от нуля до десяти: то есть чем меньше уровень продуктивности, тем меньше работоспособность. В итоге планирование времени становится более объективным и контролируемым процессом, потому что при таком подходе один свободный час в два часа дня выглядит более продуктивным, чем тот же один свободный час в два часа ночи, когда мы устали и хотим спать.

Для наглядности я создал табличку, где вы можете видеть ячейки со временем и уровнем продуктивности (см. рис. 26).

	ПОНЕДЕЛЬНИК	ВТОРНИК	СРЕДА	ЧЕТВЕРГ	ПЯТНИЦА	СУББОТА	ВОСКРЕСЕНЬЕ
09:00	3–5	3–5	3–5	3–5	3–5	0	0
10:00	3–5	3–5	3–5	3–5	3–5	0	0
11:00	3–5	3–5	3–5	3–5	3–5	0	0
12:00	7	7	7	7	7	0	0
13:00	3–5	3–5	3–5	3–5	3–5	1–8	1–8
14:00	3–5	3–5	3–5	3–5	3–5	1–8	1–8
15:00	3–5	3–5	3–5	3–5	3–5	1–8	1–8
16:00	3–5	3–5	3–5	3–5	3–5	1–8	1–8
17:00	3–5	3–5	3–5	3–5	3–5	1–8	1–8
18:00	5	5	5	5	5	1–8	1–8
19:00	5	5	5	5	5	1–8	1–8
20:00	5	5	5	5	5	1–8	1–8
21:00	8	8	8	8	8	9	9
22:00	8	8	8	8	8	9	9
23:00	8	8	8	8	8	8	8

Практика, пишем код
Теория, общие IT знания
Отдых

Рис. 26. Зависимость уровня продуктивности от дня недели и времени суток

На рисунке вы можете видеть, что уровень продуктивности в рабочие и выходные дни сильно различается. Если у вас нестандартный график, к примеру, день отдыха, затем два дня работы, то ваша таблица будет выглядеть по-другому.

Также вы можете заметить, что уровень продуктивности в разные часы тоже разный. Это связано с тем, что настрой и способность воспринимать информацию часто зависят от внешних факторов, поэтому я не могу на сто процентов предсказать уровень продуктивности на тот или иной момент, но, исходя из своего графика, я могу примерно спрогнозировать его.

Но нужно учитывать, что уровень моей работоспособности так же ограничен, как и количество часов. Поэтому, как бы я ни старался, я не могу превысить этот показатель, и со временем по мере нарастания усталости работоспособность и продуктивность будут понижаться.

Чтобы вам было проще понять логику моих размышлений, давайте возьмем для примера стандартный рабочий график с 9:00 до 17:00 пять дней в неделю. Представим, что я работаю в офисе и постоянно нахожусь на рабочем месте, имея непрерывный доступ к компьютеру. Начало рабочего дня — это обычно самый интенсивный период. Поэтому я знаю, что в начале каждого рабочего дня буду усиленно работать и разгребать дела. В связи с этим у меня не получится полностью сконцентрироваться на изучении программирования, поэтому больше пяти баллов в графе мы поставить не можем. Если работа очень тяжелая, то уровень продуктивности будет не более трех, а может быть, и меньше.

Примерно так же будет выглядеть каждое мое утро с понедельника по пятницу, поэтому уровень продуктивности на этот период

времени в период будничных дней везде примерно совпадает — от трех до пяти. Естественно, он может быть и равен нулю, если утро сильно загруженное.

Тут хочу отметить, что в нашем абстрактном примере мы допускаем, что на этой офисной работе я имею возможность во время перерывов переключаться на свои дела. То есть не сижу целый час непрерывно на телефоне, а, скажем, в течение сорока пяти минут интенсивно работаю, а потом в перерыве могу посмотреть какие-то видео по программированию, вместо того чтобы выпить кофе или подышать воздухом на улице с коллегами.

Теперь вернемся к табличке. Напомню, что уровень продуктивности в баллах — это то, насколько я готов в данный момент воспринимать материал и размышлять над ним. По причине того, что это утро, обычно я нахожусь в «боевом режиме» и за эти десять-пятнадцать минут перерыва могу посмотреть какой-то материал и понять его.

Когда наступает обеденный перерыв (в табличке это 12:00), то тут я уверен, что у меня не будет встреч, звонков, а значит, уровень продуктивности будет соответствовать примерно семи. Но не выше, потому что к обеду, после интенсивной работы в первой половине дня, мы обычно устаем и чувствуем себя утомленными.

Дальше идет вторая часть рабочего дня с 13:00 до 17:00, и там уровень продуктивности тоже примерно от трех до пяти, а следовательно, и схема действий такая же. Если я нахожусь не в офисе, например, еду на встречу, то в это время могу просто слушать теоретические материалы. Благодаря этому в каждый час рабочего дня навык программирования можно прокачивать. Даже если в общей сумме будет потрачено тридцать минут, это лучше, чем ноль, согласитесь.

Когда рабочий день заканчивается, у меня обычно появляется свободное время, но тут нужно понимать, что, во-первых, я уже довольно сильно устал, а во-вторых, повседневные заботы никто не отменял. Лично я после работы предпочитаю сделать перерыв и заняться спортом, либо побыть с семьей, либо просто вздремнуть. В 21:00 мы ужинаем, и потом, в десятом часу, у меня появляется свободное время, которое я могу потратить на программирование. В этот момент уровень продуктивности равен восьми, потому что я уже чувствую себя отдохнувшим и могу приступить к обучению с новыми силами. Это время я считаю самым продуктивным, и потому в этот промежуток я стараюсь преимущественно писать код и думать над самыми сложными заданиями.

Примерно по такому же принципу я организовываю выходные. Первое, на что нужно обратить внимание, — это нули в графе уровня продуктивности с 9:00 до 11:00. Это связано с тем, что я люблю поспать в выходные, поэтому никакими делами в это время не занимаюсь, только если они совсем уж важные. Далее продуктивность колеблется от одного до восьми, потому что в выходные дни мы с семьей часто выбираемся из дома. Бывает такое, что меня целый день нет дома и я не могу учиться вовсе, но бывает и такое, что на улице плохая погода, и у меня появляется час или два на свои дела. Вечером у меня также бывают свободные часы, плюс я полностью отдохнувший и свежий, так что это наилучшее время для работы над собственными проектами и для обучения. Поэтому уровень продуктивности в эти часы — от девяти до десяти.

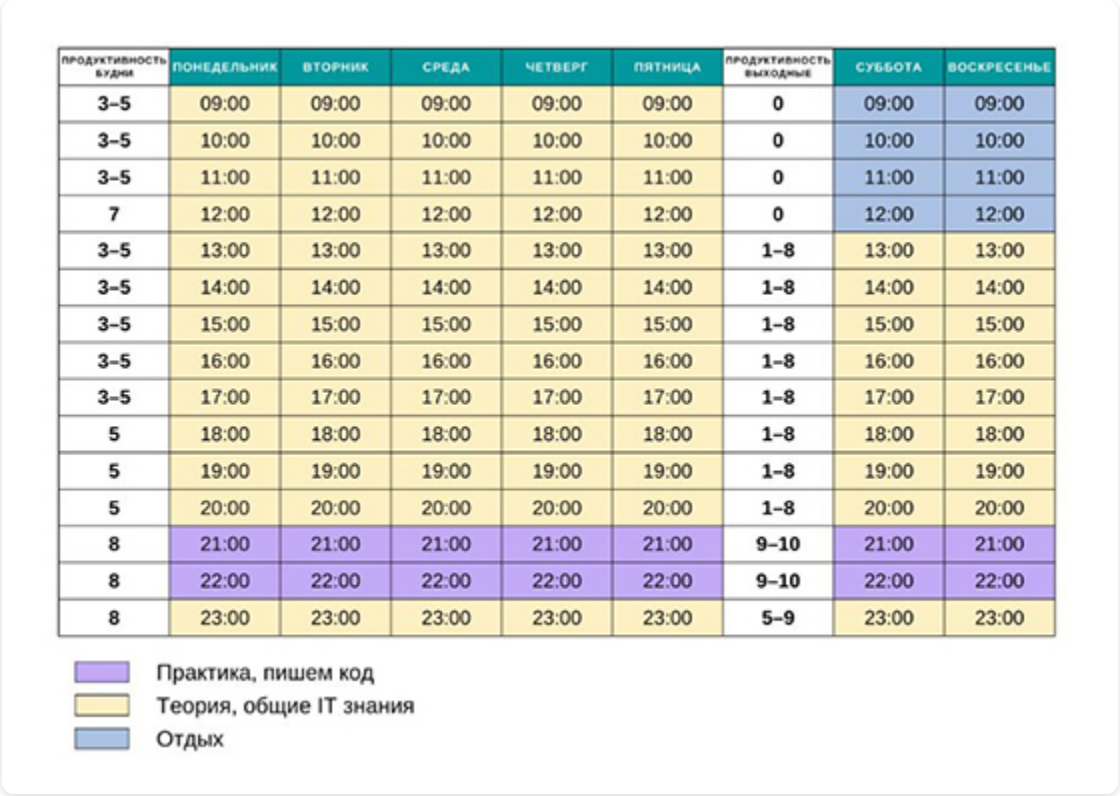
После того как график составлен, я определяю список целей, которых мне нужно достичь. В нашем случае это будет изучение программирования. Тут очень важный момент, на который я прошу обратить внимание. При изучении программирования я разделяю освоение теории и написание кода на два отдельных блока.

ЦЕЛЬ	ТРЕБУЕМЫЙ УРОВЕНЬ ПРОДУКТИВНОСТИ
Теория	1–7
Практика	8–10

Рис. 27. Требуемый уровень продуктивности для теории и практики

При таком подходе намного легче согласовывать время, цели и ресурсы. Посмотрите на рис. 27. С правой стороны вы видите тот уровень продуктивности, который необходим для работы над данной задачей. То есть для изучения теории мне потребуется от одного до семи баллов продуктивности, а при написании кода — от восьми до десяти. Работа над теорией тоже может быть разделена, потому что сложный теоретический материал, когда объясняют язык программирования, — это семерка, но теория, где программист просто делится какими-то секретами, может быть равна одному. Поэтому в теоретическом материале разница большая, но я не стал разделять их в таблице, в ней и так большой разброс.

Дальше нам осталось только соединить наши цели с таблицей времени (см. рис. 28).



Продуктивность будни	Понедельник	Вторник	Среда	Четверг	Пятница	Продуктивность выходные	Суббота	Воскресенье
3–5	09:00	09:00	09:00	09:00	09:00	0	09:00	09:00
3–5	10:00	10:00	10:00	10:00	10:00	0	10:00	10:00
3–5	11:00	11:00	11:00	11:00	11:00	0	11:00	11:00
7	12:00	12:00	12:00	12:00	12:00	0	12:00	12:00
3–5	13:00	13:00	13:00	13:00	13:00	1–8	13:00	13:00
3–5	14:00	14:00	14:00	14:00	14:00	1–8	14:00	14:00
3–5	15:00	15:00	15:00	15:00	15:00	1–8	15:00	15:00
3–5	16:00	16:00	16:00	16:00	16:00	1–8	16:00	16:00
3–5	17:00	17:00	17:00	17:00	17:00	1–8	17:00	17:00
5	18:00	18:00	18:00	18:00	18:00	1–8	18:00	18:00
5	19:00	19:00	19:00	19:00	19:00	1–8	19:00	19:00
5	20:00	20:00	20:00	20:00	20:00	1–8	20:00	20:00
8	21:00	21:00	21:00	21:00	21:00	9–10	21:00	21:00
8	22:00	22:00	22:00	22:00	22:00	9–10	22:00	22:00
8	23:00	23:00	23:00	23:00	23:00	5–9	23:00	23:00

Рис. 28. Зависимость уровня продуктивности от времени и целей

В итоговом виде таблица выглядит следующим образом. Желтым цветом я раскрасил все время, где уровень моей продуктивности составляет менее семи единиц. Это означает, что в часы, выделенные желтым цветом, я могу только изучать теорию, а в то время, которое отмечено фиолетовым цветом, я могу практиковаться в написании кода. Таким образом, я примерно понимаю, в какие часы какими делами я могу заниматься.

Последняя деталь, которую необходимо добавить, — это приоритет. Так как целей у нас всего две, то в данном случае все достаточно просто. Когда я изучал программирование, то самым большим приоритетом для меня была практика. Поэтому, если у меня были время и желание, я всегда занимался написанием кода. Но при изучении теоретических материалов я старался расставлять приоритеты. Ход моих рассуждений при этом выглядел следующим

образом: если я более-менее свежий и у меня есть доступ к компьютеру, то я изучаю материалы по программированию и готовлюсь к самому процессу написания кода. Но если я уставший либо еду за рулем или занимаюсь спортом и не имею возможности погрузиться в материал, то я изучаю материал второстепенной важности. Например, слушаю истории успеха программистов-самоучек или изучаю общие понятия в области программирования. Другими словами, я расширял свой кругозор за счет теории, которая не требовала полного погружения и доступа к компьютеру.

С таким подходом я медленно, но методично продвигался к своей главной цели — получить первую работу. У меня всегда в голове была эта таблица, и я пытался каждый час провести с пользой, в зависимости от того, насколько работоспособен я был в текущий момент.

Итак, еще раз подчеркну, что самое важное в этом процессе — это «фиолетовое» время, которое вы посвящаете написанию кода. Залог успеха кроется в практике. Практика, практика и еще раз практика. Чем больше часов вы потратите на написание кода, на поиск ошибок в собственном коде и на внедрение решений, тем быстрее вы дойдете до цели.

И напоследок хочу рассказать еще один эпизод из своей жизни. Это было в те времена, когда пандемия еще не перевернула нашу жизнь с ног на голову. В то время я работал в офисе, и каждый день мне приходилось преодолевать около трехсот километров. То есть я много времени проводил в дороге и мог спокойно предаваться размышлениям. В эти часы я обычно выключаю музыку и еду в тишине, прислушиваясь к своему внутреннему голосу и думая о своих делах.

В один из таких дней я ехал по дороге ранним утром. Часы показывали семь, солнце только встало из-за горизонта. Глядя на

окружающий меня пейзаж, я погрузился в свои мысли. Я думал о том, как в древние времена, в эпоху расцвета Римской империи, у людей было очень развито магическое мышление. Они всерьез верили, что есть люди, которые умеют вызывать дождь. Принципы строительства и земледелия многие государства старались хранить в секрете, и эти сведения ценились так сильно, что ради них собирались целые военные походы. Люди были готовы отдать жизни за то, чтобы узнать важные секреты.

А что сейчас? Сейчас мы видим, что у нас есть доступ к информации, о которой древний человек и мечтать не мог. Нам круглосуточно доступны любые знания, будь то сложный рецепт или уроки дрессировки собак. Более того, если всего каких-то пятьдесят лет назад, до изобретения интернета, люди тратили деньги на приобретение книг, чтобы узнать что-то новое, то сейчас практически любую книгу можно бесплатно добыть в сети.

Да, качество информации, находящейся в свободном доступе, не всегда бывает высоким. Например, такую сложную дисциплину, как медицина, вы вряд ли сможете освоить с помощью бесплатных ресурсов. Но, по крайней мере, вы можете найти какие-то основополагающие вещи и разобраться в чем-то. Также нельзя отрицать, что с развитием технологий выросло количество форматов, которые мы можем использовать при изучении чего-либо. Можно смотреть видео, слушать лекции, подкасты. Мой ребенок учит математику не по скучному учебнику, как учил ее я, а с помощью уникальных анимированных игр с красивой графикой.

Также хочу отдельно назвать такой момент, как удобство. С развитием мобильных телефонов все перечисленные форматы доступны нам практически из любой точки и в любое удобное нам время. И это открытие просто поразило меня в тот момент: ведь это

действительно похоже на волшебство — иметь доступ к такому огромному количеству информации всего лишь с одного устройства!

Но тут я посмотрел на экран своего телефона и поймал себя на мысли, что большую часть времени я использую телефон, чтобы играть в игры или сидеть в социальных сетях. А моя история просмотров в YouTube содержит преимущественно развлекательный контент. Как же так вышло, что мы, обладая таким мощным инструментом познания и имея возможность изменить свою жизнь в лучшую сторону, используем его совсем для других целей? Смешные развлекательные видеоролики набирают миллионы просмотров, в то время как уроки скорочтения или изучения языков — всего тысячи...

С этого момента я начал проводить время в интернете более осознанно. Теперь, если у меня было свободное время в дороге, я тратил его на расширение своего кругозора. На той работе, о которой я говорил выше, я провел полтора года, и за эти месяцы узнал много нового из мира IT: я слушал об истоках формирования IT-индустрии, о жизни людей, связанных с этой сферой, изучал истории успеха как великих программистов, так и обычных ребят вроде меня. Так, постепенно пересмотрев свой подход к использованию мобильного телефона, я стал еще на шаг ближе к тому, чтобы кардинально изменить свою жизнь.

Вы тоже можете сделать это и начать потреблять контент более осознанно и дозированно, наполняя свой разум полезной информацией и расширяя свой кругозор.

Хитрости и лайфхаки, которые я выработал на первом году обучения

Как только я разобрался со временем, обучение стало даваться мне гораздо легче. Но, несмотря на это, я все еще не знал, сколько месяцев или лет пройдет, прежде чем я получу свою первую работу. Поэтому, как только я понял, что это забег на длинную дистанцию, я продолжил оптимизировать все процессы и пытаться максимально облегчить свою задачу. В этой главе я расскажу, с какими сложностями я сталкивался на этом этапе и как мне удалось их разрешить.

Помню, как однажды вечером я засел за тему изучения абстракций и интерфейсов. Не буду углубляться в техническую сторону, но даже из названий самих терминов понятно, что это что-то страшное и сложное. Так и оказалось.

Я приехал после работы, поужинал и провел вечер с семьей. Уложив ребенка, я отправился к своему ноутбуку. Настроение было хорошее, состояние более-менее свежее. Просмотрев видео основного курса, я понял, что ничего не понял. Поэтому я решил поискать дополнительные материалы. Зайдя на YouTube, я вбил нужный поисковый запрос и начал просматривать все подряд видео по теме интерфейсов. Я часто пересматриваю видео от разных людей, чтобы посмотреть на одну и ту же вещь под разными углами. Но это мне не помогло, и я понял, что перегрелся. Мозг просто кипел от полученных сведений и уже не мог воспринимать никакую входящую техническую информацию. Это те ощущения, которые возникают, когда ты смотришь на яркий экран, учитель объясняет

какую-то тему, напичканную терминами, и вдруг ты осознаешь, что его слова просто идут фоном и сфокусироваться на них не получается. В этот момент могут появляться мысли, совсем не связанные с темой урока, например, что тебе нужно не забыть оплатить счет за электричество перед тем, как идти спать. Это похоже на то чувство, когда ты едешь по туннелю и смотришь вперед, в одну точку, а все, что по бокам, становится размытым и слабо доступным для восприятия.

Вот почему я никогда не могу ответить на вопрос «Сколько часов в день нужно учиться, чтобы стать программистом?». Это очень индивидуально и зависит от обстоятельств. Кто-то впадет в такое туннельное состояние через тридцать минут, потому что, например, уже устал, а кто-то — через два часа, потому что выходной и мозг свежий. Но это и не важно. Суть в том, чтобы научиться отслеживать такое состояние и вовремя принимать меры. Какие именно? Об этом я расскажу подробнее.

1. Если чувствуете, что мозг перегрелся, — делайте паузу.

Обычно я чувствовал, что «мозги кипят», в двух случаях: когда смотрел обучающие материалы повышенной сложности или когда писал код и сталкивался с какой-то проблемой — например, приложение не запускалось или код работал не так, как нужно. Я чувствовал, что ответ где-то близко, но ничего не получалось.

И во всех этих случаях мне помогало одно простое средство: сделать перерыв. Это работало, когда я только изучал программирование, и работает сейчас, когда я уже стал профессионалом. В такие моменты я просто ловлю себя в этом состоянии и вытягиваю себя из него. Если я завяз и устал, я не пытаюсь вылезти из этого болота, потому что оно может засосать еще глубже. Я просто полностью останавливаюсь и делаю перерыв.

Насколько большой должна быть эта пауза? Зависит от многих параметров, в первую очередь от степени вашей усталости и от сложности задачи. У меня бывало такое, что я мучился с какой-нибудь проблемой всю пятницу, проходился по всему коду в поисках ошибки, перекапывал все источники в интернете и не мог найти решение. Но, вернувшись в понедельник с ясной головой, я находил решение за тридцать минут.

2. Переключайтесь с одной задачи на другую, более простую.

Если мозг перегрелся, не обязательно делать большую паузу и идти заниматься другими делами. Можно продолжать заниматься тем же, но немного изменить направление работы. Например, если я чувствую в себе силы еще поработать, то могу сделать небольшой пятнадцатиминутный перерыв, а потом вернуться обратно к программированию. Но ту первую, сложную задачу я уже не буду трогать, а попробую подумать над чем-нибудь другим. К примеру, если я пишу какую-то сложную логику калькулятора, и у меня не получается тот результат, который мне нужен, то я могу переключиться на создание визуальной части калькулятора — допустим, сделать кнопки, внедрить дизайн. И тогда в понедельник, когда я, отдохнувший и посвежевший, вернусь к логике, останется только ее подключить. Поэтому, если что-то не получается, сделайте то, что точно получится, а потом вернетесь обратно к сложной части.

3. Разбивайте большие задачи на мелкие.

Вы, должно быть, заметили в предыдущем пункте, что в примере с калькулятором я разбил большую задачу по его созданию на две части: создание дизайна и написание логики. Но на самом деле каждую из этих частей я разбиваю на еще более мелкие части. В таком подходе есть немало плюсов. По причине того, что все задачи небольшие, я могу спокойно между ними переключаться. К примеру, если я чувствую, что задача потребует больше времени и я начинаю

снижать темп, то я просто переключаюсь на следующий кусок кода. И таким образом я, как мозаику, собираю все кусочки крупной задачи. Это делает процесс более гибким и снижает мой уровень стресса. Этому подходу я учу на своем бесплатном курсе, который можно найти на сайте www.borisproit.expert.

Второй бонус, который я получаю при таком подходе, — это постоянный прогресс. Чувство прогресса очень важно при программировании, особенно когда вы находитесь еще на этапе обучения. Отсутствие ощущения, что ты продвигаешься вперед, влечет за собой риски потерять мотивацию. В результате возникает опасность вообще забросить это дело, ведь ответственности намного меньше, у нас нет никаких обязательств перед работодателем, и наше обучение строится лишь на собственном желании.

Все эти небольшие хитрости и лайфхаки, которые я перечислил, очень помогали мне в процессе обучения и помогают до сих пор. Но за время обучения я много раз оказывался в ситуации, когда, столкнувшись с ошибкой, я полностью останавливался и вообще не мог найти решение проблемы. Что же делать, если ты в полном ступоре и не понимаешь, куда двигаться дальше? Как быть, если не помогают ни Google, ни обучающие видео, ни чтение технической документации?

В этом случае, как я уже писал ранее, последней инстанцией для меня становился репетитор. Но хорошая новость в том, что чем больше прокачивается ваш навык поиска решений, тем меньше вам требуется помощь репетитора. На схеме (см. рис. 29) я визуально изобразил, как снижалась частота моих обращений к репетитору с течением времени.

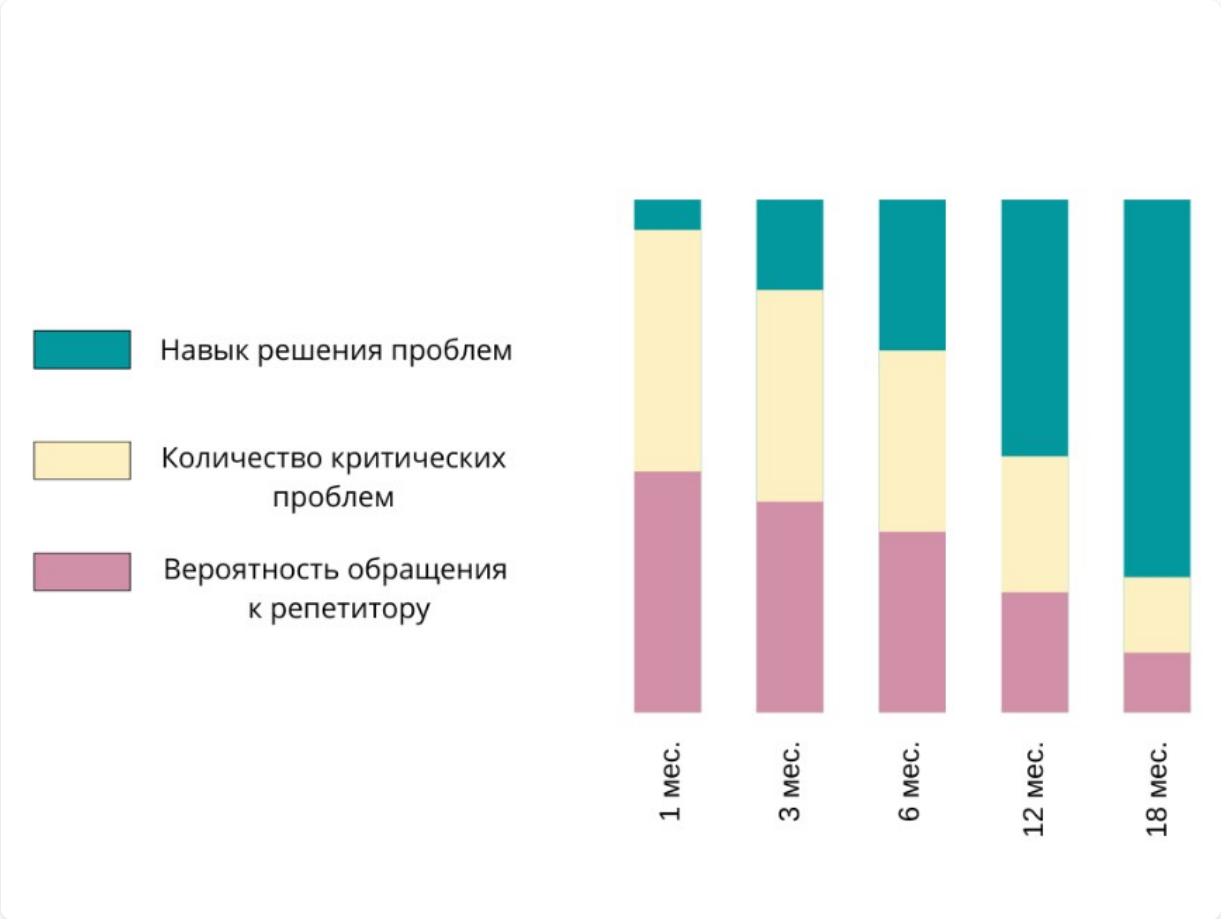


Рис. 29. Зависимость частоты обращений к репетитору от уровня навыка

По мере движения к своей основной цели любой ученик постоянно сталкивается с проблемами и необходимостью их решить. На моей схеме критические проблемы, которые не позволяют студенту двигаться дальше и вводят его в ступор, обозначены желтым цветом. Именно из-за подобных ситуаций многие перестают изучать программирование. Зеленым же цветом я изобразил навык решения проблем. А розовым — вероятность обращения к репетитору.

Как вы можете заметить, на первом месяце пути проблем возникает много, большинство из них — критические, а навык их решения еще не прокачан, поэтому на данном этапе вероятность обращения к репетитору максимально высока. Но с течением времени желтый показатель начнет падать, потому что студенту будет все проще и

проще находить решения. А зеленый показатель, наоборот, со временем прокачивается. И, как следствие, снижается необходимость в привлечении репетитора.

Подводя итог, хотелось бы еще раз донести одну из основных мыслей этой книги, что программирование — это не только написание кода, но еще и решение возникающих проблем. Со временем я привык к постоянному появлению проблем и отлично развил свой навык поиска их решений. Тем не менее бывали проблемы, которые я все равно не мог решить самостоятельно. Они забирали много энергии, вплоть до того, что мне хотелось все бросить. Именно в таких случаях я прибегал к помощи репетиторов. Поэтому я призываю всех не отказываться от помощи репетитора (будь то профессиональный программист или индивидуальный учитель), который сможет в критический момент поддержать вас и помочь найти нужное решение. Я прибегал к помощи программиста ровно три раза за все время обучения, после того как перепробовал абсолютно все. К тому моменту на поиск решений у меня уходило не менее недели, я чувствовал себя ужасно и хотел все бросить. Поэтому я искал профессионала, платил ему деньги, и проблема была решена. Сайтов, где можно найти программиста для индивидуального урока, очень много. Я просто покупал часовое занятие, и мы вместе решали мою проблему. Самое главное на тот момент было выйти из этого болота, в которое я попал, и продолжить путь. Выход есть всегда.

Где брать идеи для первых проектов?

Шел уже второй год обучения. Сейчас я осознаю, что это довольно много. Но на тот момент мне это казалось нормальным. Учеба превратилась в рутину. Я оптимизировал все, что только мог. Не было ни одного дня, чтобы я не узнал чего-то нового или не написал хотя бы строчки кода. Я хотел как можно скорее дойти до финиша, пересечь красную ленту и получить заветный приз. Дома тоже все уже приспособились к моему графику. На выходных меня иногда специально оставляли в одиночестве, чтобы я мог полностью погрузиться в процесс.

В целом я чувствовал себя хорошо, поймал нужный ритм, и оставалось только продолжать двигаться вперед. Еще одним важным фактором было то, что передо мной стояла ясная цель: наличие трех готовых приложений, опубликованных в Play Store, которые будут включать в себя все основные технологии, указанные в интересующих меня вакансиях. Эту цель очень легко измерить, поэтому и идти к ней было намного легче. Я изучал свой курс, где мы как раз делали приложения, которые я мог разместить в Play Store и показать потенциальному работодателю на собеседовании.

Одним словом, на тот момент мой мозг полностью осознавал всю картину происходящего, а потому не было никаких проблем с мотивацией и с энергией для реализации своих целей. Как я писал ранее, изучение программирования — это не просто изучение кода, это еще большое количество процессов, над которыми нужно работать. Поэтому спустя год, когда я знал, что, как и в каком ритме мне нужно делать, я пребывал в хорошем рабочем тонусе.

Но почему именно три приложения? И что это за приложения?

Три — потому что в вакансиях, которые мне были интересны, было написано, чтобы кандидат имел в наличии хотя бы два приложения. Поэтому я решил опубликовать три, чтобы было больше, чем нужно. И теперь хочу рассказать, как я искал идеи для своих первых проектов, которые в итоге помогли мне устроиться на свою первую работу.

В этой книге мы немало говорили о том, как важна практика для обучения. И результатом ваших трудов можно считать первый завершенный проект. В моем случае это было приложение, которое показывало список рецептов для приготовления разных блюд. Если нажать на изображение блюда, пользователь попадал на экран с описанием. Сверху отображался видеоплеер, где можно было увидеть процесс приготовления блюда, а снизу были рецепт в текстовом виде, список ингредиентов и комментарии пользователей (см. рис. 30).

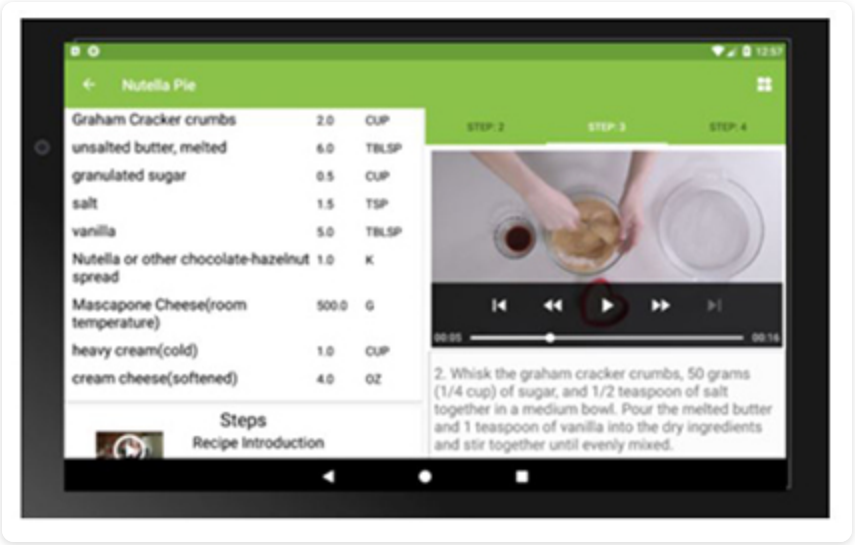
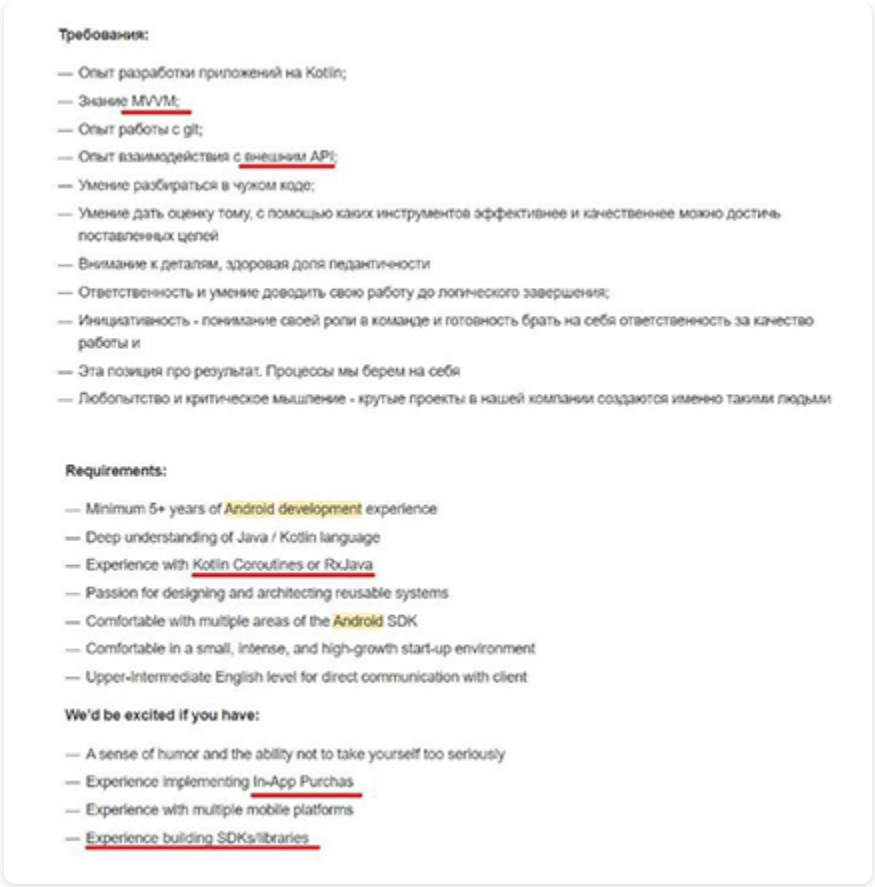


Рис. 30. Скриншот моего первого приложения

Я помню это невероятное ощущение, когда я впервые запустил скачанное с Play Store приложение на своем мобильном телефоне и воочию увидел результат своих трудов. Да, пусть это не суперкрутое приложение, и оно не заинтересует миллионы пользователей, но я создал его своими руками, продираясь через все возникающие проблемы и исправляя все ошибки. И вот я нажимаю все кнопки и понимаю, что приложение работает отлично. Я был особенно рад тому, что проект закончен, и теперь у меня есть что-то, что можно добавить в портфолио. А еще тому, что я закончил очередной модуль обучающего курса, это давало силы и мотивацию для следующего рывка.

Но вернемся к теме главы. Как искать идеи для первых проектов? Многие учащиеся — и я был в их числе — размышляют, какие именно приложения писать для того, чтобы добавить их к себе в резюме.

Лично я считаю, что сама идея не так важна, как ее реализация. Если вы сделаете приложение-чат, то это не будет каким-то новшеством, и вы не удивите самой идеей рекрутера. Но если этот чат будет иметь какие-либо сложные механизмы, например, возможность сохранения истории чатов или обработки сообщений (с их сохранением и последующей отправкой) в условиях авиарежима, когда у пользователя нет доступа к интернету, то именно этот функционал и будет интересен работодателю. Поэтому в качестве своих первых проектов вы можете смело брать распространенные идеи, которые сразу понятны и их не нужно объяснять. И уже в имеющийся стандартный функционал внедрять свои собственные идеи, которые будут основываться на списке требований из вакансий. На рис. 31 я привел пример требований к навыкам, которые хочет видеть работодатель. Эти требования я собрал из нескольких типовых вакансий.



Требования:

— Опыт разработки приложений на Kotlin;
— Знание MVVM;
— Опыт работы с git;
— Опыт взаимодействия с внешним API;
— Умение разбираться в чужом коде;
— Умение дать оценку тому, с помощью каких инструментов эффективнее и качественнее можно достичь поставленных целей
— Внимание к деталям, здоровая доля педантичности
— Ответственность и умение доводить свою работу до логического завершения;
— Инициативность - понимание своей роли в команде и готовность брать на себя ответственность за качество работы и
— Эта позиция про результат. Процессы мы берем на себя
— Любопытство и критическое мышление - крутые проекты в нашей компании создаются именно такими людьми

Requirements:

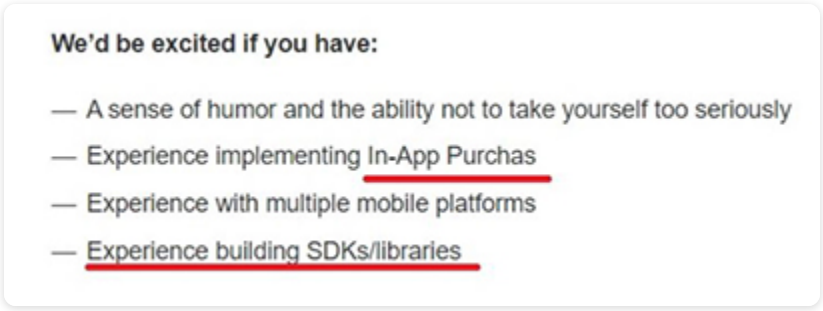
— Minimum 5+ years of Android development experience
— Deep understanding of Java / Kotlin language
— Experience with Kotlin Coroutines or RxJava
— Passion for designing and architecting reusable systems
— Comfortable with multiple areas of the Android SDK
— Comfortable in a small, intense, and high-growth start-up environment
— Upper-Intermediate English level for direct communication with client

We'd be excited if you have:

— A sense of humor and the ability not to take yourself too seriously
— Experience implementing In-App Purchas
— Experience with multiple mobile platforms
— Experience building SDKs/libraries

Рис. 31. Список типичных требований из вакансий

Давайте проанализируем пару требований.



We'd be excited if you have:

— A sense of humor and the ability not to take yourself too seriously
— Experience implementing In-App Purchas
— Experience with multiple mobile platforms
— Experience building SDKs/libraries

Рис. 32. Скриншот с сайта вакансий hh.ru

В переводе с английского требования звучат так:

- наличие чувства юмора и возможность не воспринимать все слишком серьезно,
- наличие опыта в работе с In-App Purchase,
- наличие опыта в работе с несколькими мобильными платформами,
- наличие опыта в построении библиотек.

Красным я выделил опыт внедрения технологии In-App Purchases. Эта технология позволяет принимать платежи от пользователей. Поэтому, если вы хотите добавить этот навык в свое резюме, вы можете внедрить в приложение чат, возможность пользоваться расширенным пакетом шрифтов для переписки. Таким образом, в вашем проекте в разделе «Используемые технологии» появится строчка «In-App Purchases», что будет отвечать одному из требований вакансии. А на самом собеседовании, когда вас спросят про опыт, можно сказать, что вы сделали чат и внедрили в него возможность брать с пользователя деньги за дополнительный функционал. И я вас уверяю, в ответ вы услышите что-то типа: «О да, очень интересно, мы как раз используем In-App Purchases у себя на проекте, расскажите об этом опыте подробнее!» Поэтому берите любое понятное для обычного человека стандартное приложение и внедряйте в него те технологии, которые чаще всего просят в вакансиях. Тем самым вы сильно увеличиваете шанс, что ваше портфолио заинтересует работодателя.

Подводя итог данной теме, я хочу еще раз сфокусировать ваше внимание на том, насколько важны проекты, над которыми вы работаете в процессе обучения. Все они состоят из множества мелких деталей, а в итоге складываются в общую картину. В этом процессе все взаимосвязано, и развитие навыка написания кода — далеко не единственный фактор, которому нужно уделять много внимания. Мыслите глобально! Вы должны стратегически понимать, как все эти фрагменты связаны друг с другом. Поэтому я призываю

вас осознанно писать код и понимать, где он пригодится, а не просто повторять за учителем определенные действия.

И второй момент, на котором я хочу остановиться. Уникальность программирования состоит в том, что человек может набираться опыта самостоятельно, вне зависимости от других людей. Если взять такие профессии, как, например, учитель, юрист или доктор, то там для получения опыта обязательно нужны другие люди, на которых можно практиковаться: ученики, пациенты и т. д. Но если вы программист, то вы можете бесконечно прокачивать свои навыки самостоятельно, без чьей-либо помощи, и рано или поздно этого опыта будет достаточно, чтобы на вас обратил внимание рекрутер. Однако это нужно делать осознанно, понимая всю суть происходящего. Я делал именно так, и это сработало, ведь в один момент сложность моих проектов перевесила факт отсутствия технического образования, и на меня обратили внимание рекрутеры. Так я и нашел свою первую работу. Расскажу об этом периоде подробнее.

Мой опыт поиска работы и финальный рывок

С того памятного момента, когда я посетил дом программиста и задумался о смене профессии, прошло примерно два года. Я хорошо помню этот период. Это было начало осени, я заканчивал свой продвинутый курс на Udacity. К тому моменту я уже мог сказать, что владею навыком программирования пусть и на довольно низком, но уверенном уровне. По причине того, что я учился в одиночестве, у меня не было возможности сравнить себя с кем-то другим и понять, насколько я плох или хорош. Заканчивая курс, я уже разместил несколько проектов на Play Store и потихоньку начинал планировать уже третью на тот момент попытку найти работу.

Первую попытку я осуществил после шести месяцев обучения. И тогда я не получил ни одного звонка или отклика. Вторую попытку предпринял еще через месяцев шесть, и она тоже с треском провалилась. И после этого я решил просто учиться до того момента, пока не буду чувствовать себя более-менее уверенно при написании кода и не создам несколько качественных приложений для портфолио. И в принципе, к моменту третьей попытки у меня все это уже было, но я планировал потратить еще несколько месяцев, чтобы подтянуть теорию, хорошо подготовиться к собеседованиям и отполировать до блеска свое последнее приложение. Однако, как обычно, жизнь внесла свои коррективы, и в один прекрасный момент меня уволили. Это было как раз то увольнение по СМС, о котором я писал выше.

Поэтому в одно прекрасное утро все мои планы резко поменялись, и, отойдя немного от шока, я понял, что стою перед выбором. С

одной стороны, я понимал, что нужно как можно скорее найти любую работу, потому что приток денег остановился, а накоплений надолго не хватит. С другой стороны, мне хотелось начать сразу искать работу программистом, но для этого мне было необходимо закончить приложение и доработать резюме, а на это потребуется какое-то время. В итоге я решил рискнуть и начал сломя голову доделывать дела в программировании и готовиться к собеседованиям.

Спустя неделю подготовки я выбрал время и полночи просидел в интернете, закидывая свое резюме на всевозможные порталы вакансий. Каково же было мое удивление, когда утром, примерно в девять часов, я получил свой самый первый звонок рекрутера. Я был ужасно взволнован, мой голос дрожал, и я просто мычал что-то в трубку. Первым чувством, которое я испытал, когда разговор закончился, было невероятное облегчение от того, что мне по крайней мере звонят.

Но хотя я и начал получать звонки, тем не менее быстро осознал, что моего опыта все равно не хватает. Рекрутеры спрашивали про технологии, и когда они узнавали, что у меня нет коммерческого опыта, они вежливо говорили, что перезвонят, и клали трубку. Я не буду расписывать весь процесс поиска работы в деталях, потому что про это можно рассказывать очень долго. Этой теме я посвятил целый раздел в расширенной версии книги и отдельный курс, который вы можете найти на моем сайте. Здесь же я кратко скажу, что благодаря советам одного международного рекрутера из Индии я очень сильно переработал свое изначальное резюме, и это позволило мне перейти на следующий этап и начать получать приглашения на собеседования.

В итоге спустя примерно две недели я сильно продвинулся в процессе поиска первой работы, и мне регулярно звонили

рекрутеры. Более того, после разговора я получал от них приглашения на собеседования.

Собеседования пошли одно за другим, но все были провальными. Я понял, что очень плохо знаю теорию. Хотя я и мог ответить на вопросы касательно кода, но все время допускал ошибки, когда речь шла о теории. Поэтому после нескольких проваленных собеседований я с головой погрузился в изучение теоретической базы. Оказалось, что в интернете есть огромное количество вопросов по Java и Android, и именно их задают на собеседовании. И вот я сидел, учил ответы, заглядывая в теоретические материалы, которые пропустил в ходе обучения.

И только после этого, спустя примерно четыре недели с момента увольнения, я получил свое первое предложение о работе в роли Android-разработчика. Я помню, что в те дни у меня было по три собеседования в день. Мне постоянно звонили рекрутеры, и я постоянно висел на телефоне, отвечая на одни и те же вопросы касательно моего опыта. А в перерывах продолжал учить теорию по Java и Android. И я так сильно набрал скорость, что когда получил звонок, где мне сказали, что меня взяли, даже не смог вспомнить компанию и команду, собеседование в которой я проходил.

Здесь еще вот какой момент. В первые две недели я откликался только на вакансии города, в котором я жил. Но так как меня не брали, спустя две недели я начал подавать отклики на все возможные вакансии в любых городах, решив в случае чего переехать. Именно поэтому количество звонков сильно увеличилось. В итоге мне предложили работу в другом городе, причем переезд и проживание я должен был оплачивать самостоятельно. Сейчас я понимаю, что условия, на которые я тогда согласился, были не самыми лучшими, но в той ситуации у меня не было совершенно никакого выбора. К тому моменту я не работал уже месяц и за это

время потратил все свои сбережения. Я, конечно же, согласился на предложение, оставил жену и ребенка дома, а сам поехал на машине в другой город. Мне пришлось занять денег, часть из которых я оставил семье, а часть потратил на дорогу, аренду жилья и еду в новом городе. Помню, что первые две недели до зарплаты я покупал только макароны. Так я продержался до первой зарплаты, и потом уже стало легче.

Но все эти временные трудности были совершенно не важны. Потому что я наконец-то добрался до этой заветной красной ленточки, забег до которой я начал двумя годами ранее. Я помню, что просто лежал на полу в гостиной своей арендованной квартиры, смотрел в потолок и пытался осознать весь путь, который я прошел с момента, когда побывал в доме того программиста, и до момента, когда мне нужно самому уже выходить на свою первую работу; с того вечера, когда я сидел в темноте и в голове кружились сплошные вопросы, и до момента, когда мне на почту прислали оффер для подписи. Это был важный путь. В результате я приобрел навык, который не только кормит меня и мою семью и позволяет ложиться спать с уверенностью в завтрашнем дне, но и дает мне веру в свои силы. Отныне я твердо убежден в том, что при правильном подходе, при высоком уровне осознанности и при хорошей дисциплине можно много чего достичь. Поэтому все усилия однозначно стоили того.

Сколько часов у меня ушло на изучение программирования?

Процесс обучения настолько вошел в привычку, что в какой-то момент я перестал считать месяцы. Я просто продолжал учиться, двигаясь вперед. То есть я очень хорошо помню первые шесть месяцев своего пути, остальное время представляется мне чем-то однородным и однотипным. Я писал приложение, параллельно изучая новый материал. Время от времени всплывали новые ошибки и проблемы, я их решал, и это меня очень радовало. Я продолжал свой путь, на котором опять появлялись новые проблемы. Таким образом, все двигалось по кругу, опять и опять.

Как я писал в главе «Контроль времени», одной из первых оптимизаций, которую я сделал, было начало счета потраченных часов. Поэтому вместо того, чтобы пытаться измерить уровень своего навыка программирования, я считал потраченные часы. Навык измерить сложно, часы легче. Так вот, дойдя до самого конца, я еще раз убедился в силе этого лайфхака.

Если вы не измеряете ваше время обучения, то не управляете процессом. В результате доучиться будет сложнее, потому что вы не знаете, сколько времени уже прошло и сколько осталось. Мозг начинает паниковать. Помимо этого, не измеряя прогресс в часах, вы не видите, сколько реально вы тратите времени ежедневно, поэтому велик соблазн начать делать себе поблажки. Именно поэтому, если мы хотим похудеть, первым делом мы начинаем измерять калории, чтобы взять количество съеденного под контроль.

Я приведу лишь ориентировочную цифру, потому что, к сожалению, не сразу начал считать время, которое тратил на обучение. В

среднем у меня ушло примерно 800–900 часов со старта обучения до момента, как я начал работать программистом. Но здесь я бы хотел напомнить, что в финале этого пути я уже не пытался найти работу на начальный уровень Junior. Вместо этого я осознанно пытался доучиться до среднего уровня, Middle. Поэтому у меня ушло так много времени.

Этот принцип тысячи часов, как я его называю, справедлив практически для любого рода занятий, и программирование — не исключение. Потратьте тысячу часов на написание кода, и когда вы преодолеете этот рубеж, вы с большой вероятностью либо уже устроитесь на работу, либо будете буквально в одном шаге от ее получения.

Поэтому цените свое время, и если учитесь, то делайте это грамотно и организованно. А если вы уже взрослый работающий человек, имеющий семью и окруженный повседневными заботами, время становится еще более ценным ресурсом. Поэтому, коль скоро вы решились на переход в IT, делайте это максимально быстро и организованно, иначе вы сами себе усложните жизнь и продлите этот период, когда вы будете вынуждены тратить свое личное свободное время и не получать при этом никакой оплаты, а возможно, и удовольствия.

Стоило ли тратить на этот путь целых два года? Стоило ли просиживать свое свободное время за компьютером, изучая тонны материалов? Ведь за это время можно было организовать собственный бизнес, создать фирму либо освоить большое количество более легких направлений: в нынешнее время люди спокойно зарабатывают на YouTube и в TikTok, ну или просто на стриминге игр. Есть пути намного проще и интереснее. В конце концов, можно было просто получать удовольствие от жизни, отдыхая и наслаждаясь свободным временем.

Но я все равно не жалею и считаю, что это было не только одно из самых важных, но одно из самых правильных решений, которые я когда-либо принимал. У него есть ряд важных преимуществ.

Во-первых, дойдя до конца, я понял, что могу освоить сложный навык! Осознание того, что я осилил что-то большое и сложное, отразилось и на моей личности. У меня появилась уверенность в себе и в своих силах, и это чувство уже никуда не денется. Этот опыт научил меня тому, что если не сдаваться и идти, то все равно дойдешь до результата.

Во-вторых, у меня появился важный и интересный навык, который позволяет создавать что-то осязаемое. То, что могут использовать другие люди. Это не то же самое, что делать бизнес или что-то продавать, когда ты занимаешься организацией процессов, продумываешь схему работы. Это нечто другое, тут идет процесс создания реального продукта. Ты делаешь приложение и видишь, как им кто-то пользуется. Это классное ощущение. Я не говорю, что создание продукта лучше или хуже организации бизнеса. Но в этом лично я вижу больше творчества.

Третье — это востребованность. Навык программирования очень востребован, и я понимаю, что теперь мне открыты двери практически в любые страны. И со временем, по мере развития, эта востребованность будет только расти. Время в данном случае работает на меня. Поэтому чувство того, что мои навыки нужны другим людям, очень вдохновляет и воодушевляет. Я не испытывал такого, когда работал в сфере менеджмента и продаж.

Ну и, наконец, это стабильный заработок. Наличие реального востребованного навыка дает ощущение стабильности и спокойствия. Даже во времена пандемии ковида мне не составляло особого труда найти работу. Если в стране все закрыто, я всегда могу

работать удаленно. И при этом буду получать достаточно неплохие деньги.

Поэтому, пройдя весь этот путь и оборачиваясь назад, я могу твердо сказать, что оно того стоит. Владение реальным навыком дарит мне ощущение радости. А если этот навык еще и востребован и хорошо оплачивается, то приходит чувство стабильности и спокойствия. Я всегда смотрел на врачей и думал: как же круто, что ты умеешь лечить людей. Не все это могут делать, поэтому всем нужны люди с этим навыком. Конечно, программирование вряд ли можно поставить в один ряд с медициной по важности, но тем не менее в наше время, когда мир активно движется в сторону развития технологий, когда ни одна компания не представляет свою работу без web-сайта, все стараются сделать свои приложения, автоматизировать процессы и внедрить управление проектами, важность навыка программирования уже сложно переоценить.

Как мне удалось перестать сравнивать себя с другими в процессе работы

Так получилось, что я прошел свой путь в одиночестве и не имел возможности сравнивать свой прогресс с другими людьми. Минус здесь в том, что у меня не было ориентира. Я не мог оценить, насколько хорошо или плохо я понимаю материал. Иногда мне казалось, что совсем ничего не понимаю. Я задумывался, нормально ли тратить на этот материал столько-то времени или обычно его проходят намного быстрее.

Но одновременно такое мое положение было и плюсом, потому что, не опираясь на пример других людей, я мог не торопиться с изучением материала, ведь мне не нужно было никого догонять.

Мои чувства можно описать так: я как будто находился в каком-то белом пространстве без стен, потолка и пола, и мне не от чего было оттолкнуться и не к чему было стремиться. Просто ощущение невесомости. Наиболее отчетливо я испытал это ощущение ближе к концу обучения.

Впрочем, забегая вперед, могу сказать, что оно никуда не делось даже после того, как я впервые начал работать в команде. Более того, к этому ощущению присоединилось чувство неуверенности в себе — тот самый пресловутый синдром самозванца. В первый же день мне в голову полезли разные неприятные мысли: «Они умнее меня, талантливее меня, продуктивнее и моложе меня». Если вы обучаетесь на курсах в команде, то это чувство вы можете испытать намного раньше, чем я. Эти мысли появляются сразу же, как только

из белого пространства невесомости вы переноситесь в пространство с точками опоры, от которых уже можно отталкиваться. В нашем случае такие точки опоры — показатели людей, с которыми мы учимся или работаем. Оценивая их достижения, я мог понять, хорош я или плох. Но вот в чем проблема: в основном я акцентировал внимание на том, насколько я хуже. Поэтому постоянно возникало неприятное чувство неуверенности, а иногда и стыда. Как в школе, когда хотелось поднять руку и ответить вслух, но боязнь ошибиться и быть поднятым на смех заставляла сидеть тихо и лишний раз не высовываться.

Теперь, когда я уже получил опыт профессиональной разработки и поработал во многих командах, я могу поделиться с вами одним секретом. Он заключается в том, что даже спустя столько лет в моем восприятии ничего не изменилось. Я до сих пор испытываю эти ощущения при общении с коллегами и невольно сравниваю себя с другими программистами. Временами я чувствую неуверенность, задавая вопрос публично. И мне до сих пор кажется, что мой уровень недостаточно высок. Я так и не смог избавиться от этих мыслей, поэтому мне ничего не оставалось, кроме как изменить свое отношение к ним.

Как именно я это сделал?

Во-первых, я начал уважать себя и ценить свои навыки. Я осознал, что я там, где я есть сейчас, не благодаря каким-то связям, удаче или случайному стечению обстоятельств. Я стал профессиональным разработчиком и получил должность Senior Software Engineer в результате своих собственных действий, благодаря упорному труду и приобретенным знаниям. Я этого заслуживаю, ведь я прошел собеседование, успешно преодолел все тесты и не должен испытывать никаких угрызений совести.

Во-вторых, я понял, что это нормально — не иметь ответов на все вопросы. Не существует людей, которые знают все. Каждый имеет право искать ответы и ошибаться. Все мы люди. Благодаря осознанию этих моментов я перестал пытаться делать вид, что я идеален и знаю все на свете. Поэтому страх и неуверенность в себе отступили. Если я чего-то не знаю или не понимаю, я ставлю в известность об этом команду. Также я не стесняюсь сообщить о том, что на ознакомление с этим или иным вопросом мне может потребоваться больше времени. И никаких проблем с таким подходом я ни разу не испытал. О своем опыте коммуникации в команде я более детально расскажу в расширенной версии книги.

Проблема неуверенности в себе и сравнения себя с другими особенно была особенно острой в самом начале профессионального пути, когда я был еще просто программистом-самоучкой без диплома и опыта. Мне казалось, что я случайно попал на проект, в отличие от коллег, у большинства из которых за плечами имелись реальный опыт и техническое образование. Я ужасно волновался, и сейчас с улыбкой вспоминаю эти страхи. Ведь спустя несколько недель работы я понял, что моих навыков вполне достаточно, чтобы хорошо выполнять поставленные задачи. Как я уже рассказывал ранее, я осознанно не пошел на должность разработчика уровня Junior, а сразу целился на Middle, и на собственных проектах я повысил свой уровень программирования настолько, что по качеству работы и скорости никак не уступал остальным программистам. И когда старшие программисты хвалили мою работу, было очень приятно ощущать, что подготовка не прошла даром.

Поэтому я советую не принимать близко к сердцу все негативные мысли и страхи. Не пытайтесь казаться умным роботом, который не совершает ошибок. От ошибок никто не застрахован, все мы учимся, поэтому старайтесь просто быть собой. Потому что в действительности ничего страшного не случится, даже если вы

ошибетесь или попросите о помощи. Тем более если вы устраиваетесь на должность Junior: спрос с начинающих программистов вообще небольшой, и изначально на этом уровне никто не ждет от вас каких-то подвигов. Набирая в команду начинающих специалистов, компания осознанно идет на то, что неизбежно будут ошибки, много вопросов и исправлений. Поэтому не держите эту ответственность на себе, отпустите ситуацию и наслаждайтесь процессом. В крайнем случае обратитесь к профессиональному психологу, чтобы снять тревогу.

Итак, я достаточно подробно рассказал о том, как я шел по пути обучения. Но многие моменты остались за кадром. В соцсетях и на YouTube меня нередко спрашивают о каких-то нюансах обучения, процессе поиска работы за границей или о психологических трудностях. Я выбрал наиболее частые вопросы и в следующих главах книги постараюсь честно и подробно ответить на них.

Как и зачем я получил IT-образование?

Где-то на втором году своей практики я все-таки решил получить официальное техническое образование. Мне было важно не столько приобрести какие-то знания (к тому моменту я был уже достаточно подкован), сколько обзавестись бумагой, подтверждающей наличие этих знаний. Да, реалии таковы, что в нашем мире, где цифровые технологии развиваются настолько стремительно, что для идентификации уже повсеместно используется отпечаток пальца или сетчатка глаза, наличие бумажного диплома все еще имеет значение. Особенно остро его нехватка ощущается при работе с иностранными компаниями. Я закончил казахстанский университет, и у меня диплом уровня бакалавра по специальности «менеджмент». И вопреки мнению, что наши дипломы не котируются в развитых странах, я решил не переучиваться полностью, а использовать уже имеющуюся у меня базу бакалавра, на получение которой я потратил четыре года.

Проанализировав все возможные варианты, я понял, что у меня есть три пути, по которым можно пойти, чтобы сменить специальность:

- поступить на бакалавриат по технической специальности (так как диплом бакалавра по специальности «менеджмент» у меня уже имелся);
- поступить в магистратуру;
- получить сертификат дополнительного образования.

Чтобы не ошибиться с выбором, я проанализировал плюсы и минусы каждого варианта и хочу поделиться с вами ходом своих рассуждений. При анализе меня интересовали несколько

параметров: сроки, формат обучения и условия аккредитации, а также стоимость обучения.

Вопрос стоимости волновал меня меньше всего. Я всегда считал, что инвестиция в себя, свои навыки и образование, — это хорошая инвестиция. Если знания помогут мне в достижении цели, за них стоит заплатить. Поэтому при выборе я руководствовался другими критериями.

Сроки, в которые я мог получить техническое образование, были для меня одним из важнейших факторов, влияющим на принятие решения. Я сопоставил сроки получения образования, и вот что у меня получилось:

- бакалавриат — 4 года,
- магистратура — 2 года,
- сертификат — 1 год.

Как вы можете видеть в списке выше, больше всего времени требовало обучение на бакалавриате. Поэтому такой вариант мне пришлось отмести и делать выбор между обучением в магистратуре и получением сертификата о переквалификации. Я нашел несколько вариантов магистратуры, которые заинтересовали меня своей учебной программой. Темы, которые они рассматривали в обзорной документации, мне показались практичными и востребованными в реальной практике. Но минус заключался в том, что они не имели удаленного формата. Имея работу с фиксированным графиком и большой спектр семейных обязанностей, я не мог себе позволить идти в университет на полный учебный день, так что единственным возможным вариантом для меня было удаленное обучение. Я проанализировал предложения вузов и понял, что учебные заведения стран бывшего СНГ не так активно практикуют дистанционное образование, как иностранные вузы. Поэтому я

выбирал между иностранной магистратурой и получением сертификата в иностранном вузе.

Что касается получения сертификата о дополнительном образовании, тут тоже были свои сложности. Государственные учебные заведения бывшего СНГ в принципе не предполагали возможности получить сертификат по ускоренной программе. Оставались лишь обычные частные курсы, не имеющие никакой аккредитации и не предполагающие выдачи диплома государственного образца.

Поэтому единственным вариантом, который мне подходил, было американское учебное заведение, которое предоставляло возможность учиться удаленно. Оно имело аккредитацию, его можно было найти во всех официальных списках учебных заведений. Но самое главное, что, помимо магистратуры, это учебное заведение предоставляло возможность получить сертификат о переквалификации всего за один год. Это меня более чем устраивало. Несколько подобных заведений я обнаружил и на территории Канады. В итоге я составил список из трех учебных заведений, которые подходили под следующие критерии:

- наличие программы дополнительного образования;
- наличие официальной аккредитации, которая принимается во всем мире;
- гибкий график обучения.

Я сравнил цену обучения во всех трех заведениях и в итоге остановился на американском вузе, связался с координатором и начал оформление документов.

И в этой части книги я хочу более подробно рассказать о ценообразовании в зарубежных вузах, так как эта система показалась мне интересной и гибкой.

Выбирая вуз, вы также выбираете программу, которая включает в себя определенное количество кредитов. Я не помню точно, сколько кредитов было в выбранной мной программе, но, допустим, тридцать.

В рамках программы можно было выбрать несколько направлений, каждое из которых оценивается в определенное количество кредитов. Например:

- Java Core (Основы Java) — 3 кредита,
- Java Core Advanced (Продвинутый уровень Java) — 5 кредитов,
- Web-Development (Разработка web-сайтов) — 5 кредитов,
- Mobile-Development (Мобильная разработка) — 4 кредита,
- Software Development Lifecycle (Цикл разработки программного обеспечения) — 5 кредитов и т. д.

Поскольку у меня уже имелся опыт программирования, я упомянул об этом в разговоре с координатором. После этого со мной связался куратор IT-направления и начал задавать мне вопросы о том, какими технологиями я владею. После устной беседы он попросил меня выслать резюме и какие-либо документальные подтверждения, что я имею опыт работы. К счастью, я имел на руках все, что нужно — обычно я прошу своих работодателей дать мне рекомендации с перечнем моих компетенций.

Когда в вузе рассмотрели высланные мною бумаги, мне сообщили, что благодаря моему опыту в Android-разработке они могут зачесть мне Java Core, Mobile-Development и Software Development Lifecycle.

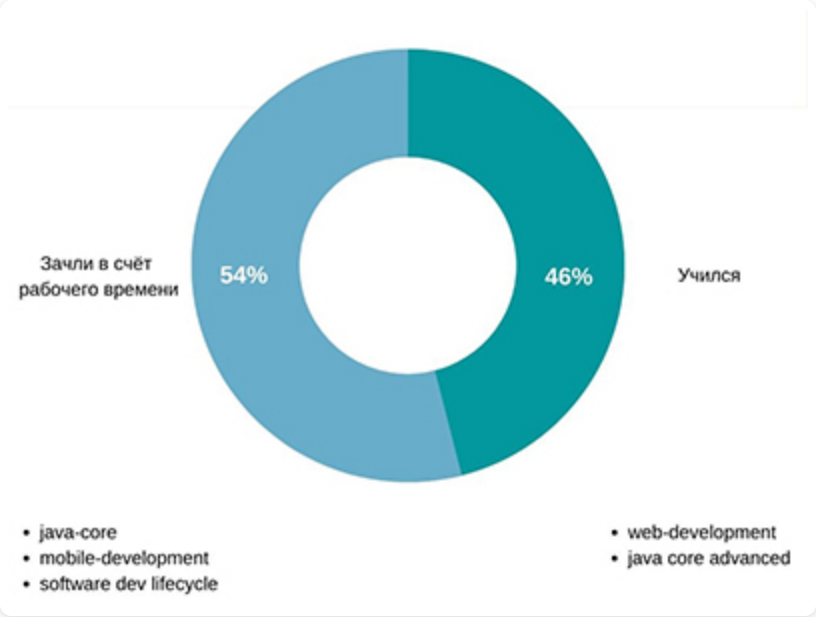


Рис. 33. Как распределились кредиты в результате зачета ряда дисциплин

Как мы видим, мне удалось сэкономить примерно половину суммы на обучение.

Также сократилось общее время на освоение программы: начав обучение в сентябре, я закончил не в мае, а в декабре, потратив всего три с половиной месяца. Теперь я был не программист-самоучка, а профессиональный специалист с техническим образованием. И хотя формально, с точки зрения навыков, для меня ничего не изменилось, но официальный статус открывал мне новые возможности: теперь я мог претендовать на работу в любой стране и в любой организации.

Для ясности приведу пример расчетов. Допустим, что один кредит стоит 50 долларов, а за все время обучения нам нужно покрыть 30 кредитов. То есть мы берем стоимость одного кредита, который равен 50 долларам, и умножаем его на 30. Таким образом, получаем полную стоимость программы. В нашем случае это 1500 долларов.

$50 (один кредит) * 30 (кол-во кредитов) = $1500 (общая стоимость)

Итак, если подвести краткие итоги, то можно выделить несколько основных моментов.

В странах бывшего СНГ дистанционное образование только начинает набирать обороты, поэтому выбора намного меньше, во всяком случае, так было тогда, когда я задался целью получить техническое образование. Однако сама программа и направления, если верить рекламным проспектам, которые мне попадались, весьма интересны и перспективны. Поэтому, если позволяют жизненные обстоятельства, я советовал бы обратить внимание на магистратуру в местных вузах с возможностью вечернего посещения.

Однако в целом на Западе рынок обучения развит очень хорошо. Западные вузы имеют отличную техническую базу и предоставляют больше гибкости при выборе формата обучения. У каждого, даже самого маленького колледжа есть свой собственный интернет-портал. Есть возможность связи с преподавателем или с сокурсниками через мобильные приложения и другие технические решения. Это очень сильно упрощает жизнь. Однако и стоимость обучения в таких заведениях, конечно, выше, особенно учитывая тот факт, что оплату придется производить в валюте.

Поэтому в данном вопросе у меня нет однозначного ответа, куда идти. Выбирайте исходя из ваших целей и возможностей. А ниже (см. рис. 34) я привел итоговую таблицу при сравнении типов обучения по срокам и по количеству затрачиваемых денег в иностранных вузах.

НАПРАВЛЕНИЕ	ПРОДОЛЖИТЕЛЬНОСТЬ	СТОИМОСТЬ
Бакалавриат	4 года	не узнавал
Магистратура	2 года	10–30 тыс. долларов
Сертификат	1 год	3–10 тыс. долларов

Рис. 34. Сравнительная таблица стоимости обучения

Много ли зарабатывает программист и стоит ли идти в IT ради денег?

Широко распространено мнение, что программисты всегда и везде хорошо зарабатывают. Так ли это на самом деле и может ли быть финансовая составляющая главным мотиватором, который способен подтолкнуть нас к смене профессии?

Разберемся сначала со вторым тезисом. Тут все далеко не так однозначно, как кажется на первый взгляд. В интернете часто пишут о том, что деньги не могут быть главной причиной, по которой нужно вставать на путь программирования. Читая это, я всегда чувствовал себя некомфортно, понимая, что внутренне не согласен с этим тезисом. И действительно, исходя из собственного опыта и мировоззрения я могу сказать, что деньги для меня — очень важный фактор. В первую очередь, это стабильность для меня и моей семьи, гарантия, что я смогу обеспечить ей достойный уровень жизни. Это возможность делать выбор и жить так, как я хочу: выбирать, что носить, что есть на ужин. Деньги позволяют помогать близким и спать спокойно, не переживая за завтрашний день.

Так почему благосостояние моей семьи и наличие выбора не могут быть для меня первостепенной мотивацией? Это ведь нормально — хотеть лучшей жизни, если текущая работа не дает такой возможности.

И еще один важный момент — это понятие хорошей и плохой мотивации. Кто вообще может точно сказать, какая мотивация плохая, а какая хорошая? Она всегда индивидуальна. Какие-то

факторы могут быть важными для одного и не иметь абсолютно никакого значения для другого. Почему же мы так зависим от мнения незнакомых людей? Когда-то, читая статьи в блогах и сообщения на форумах программистов о том, что деньги — плохой мотиватор, я расстраивался. Из-за чьего-то чужого мнения я испытывал чувство вины и опасения, что не подхожу под чьи-то стандарты. Была и неуверенность в своих силах, ощущение, что я могу не дойти до конца. В голове постоянно крутился вопрос: «А надо ли начинать такой длинный и сложный путь, если есть риск, что тебе не понравится сам процесс?» Вероятность разочарования огромна, если ты идешь в профессию только ради денег.

Но сейчас, все-таки пройдя весь этот путь и дойдя до своей цели, я могу с уверенностью сказать: хорошая мотивация — это та, которая заставляет вас двигаться вперед и дает силы не опускать руки при возникновении сложностей. Если именно деньги служат для вас основным мотивационным фактором, то пусть так и будет.

Теперь, когда я состоялся как программист и зарабатываю этим на жизнь, я могу констатировать, что отношусь к этой профессии довольно прагматично. Я не склонен романтизировать процесс написания кода и в целом не могу сказать, что люблю писать код настолько, что готов заниматься этим в два часа ночи в субботу. Я отношусь к этому навыку как к работе. Но я очень рад, что овладел им. Особенно это чувствуется, когда я выхожу из отпуска и начинаются будни. Задайте себе вопрос: с каким настроением вы выходите из отпуска на работу? Если вы испытываете негативные ощущения, не хотите опять погружаться в рабочий процесс и думаете о том, когда наступит следующий отпуск, то я могу вас понять. Я испытывал похожие ощущения до того, как стал программистом. Сейчас же при выходе из отпуска я чувствую, как медленно набираю скорость.. Никакого стресса, я просто постепенно

возвращаюсь к работе и не спеша набираю нужный темп, плавный и достаточно размеренный.

Но вернемся к разговору о том, сколько все-таки зарабатывает программист. Я не буду называть конкретных цифр, но дам примерный расклад, чтобы вы могли ориентироваться.

Во-первых, хотелось бы сказать, что много или мало — это понятие субъективное. Кто-то умеет комфортно жить на тысячу долларов в месяц, а кому-то и десяти тысяч будет недостаточно.

В целом конкретная цифра заработка зависит от очень многих факторов. Из них можно выделить основные. Это:

- направление, в котором вы работаете,
- регион, где расположена компания работодателя,
- регион вашего проживания. Я выделяю эти два пункта отдельно, потому что сейчас очень много удаленной работы, и жить можно в Омске, а работать на компанию в Лондоне. К тому же многие компании специально нанимают на работу программистов из маленьких городов, потому что им можно платить меньше, ведь жизнь в таких городах в целом дешевле,
- ваш опыт работы и конкретные навыки.

Давайте попробуем на примере рассмотреть, на что может рассчитывать программист исходя из своих навыков. Эти рассуждения основываются исключительно на личном опыте, поэтому пример в первую очередь показателен для Android-разработчиков. Но в целом его, с небольшой погрешностью, можно экстраполировать на все направления, кроме специальностей, которые не предполагают наличия навыка программирования, таких как web-дизайнер, IT-рекрутер, менеджер IT-проектов и другие. Там картина может очень сильно отличаться.

Первое ощущение, которое появилось, когда я начал работать программистом, — это то, что нижний порог зарплаты довольно-таки высокий. То есть даже будучи джуниором без опыта, можно зарабатывать более-менее нормальные деньги. Я не знаю, из какой сферы придете именно вы, но если мы сравниваем уровень заработка начинающего программиста с уровнем заработка в другой сфере, то зарплата начинающего программиста в большинстве случаев будет выглядеть довольно неплохо. Что касается меня, то, где бы я ни работал до этого, во всех сферах на старте я всегда получал меньше, нежели когда я начал работать программистом.

Такое же ощущение складывается по поводу верхней планки. Но тут важно учитывать тот факт, что у программиста высокого уровня всегда есть возможность работать удаленно на иностранные компании либо переехать из бывшего СНГ в страны первого мира. И в этом плане зарплата программиста тоже может выиграть у множества других профессий, где такой переход либо очень сложен ввиду локальных особенностей, либо вообще невозможен. Например, Java-программист высокого уровня при переходе на работу в иностранную компанию практически не потратит своего времени на адаптацию и сможет сразу приступить к работе. В то же время условный юрист для подобного перехода должен будет после переезда потратить немало ресурсов не переучивание и изучение местных законов, включая Конституцию. Поэтому высшая планка заработной платы у программистов может расти довольно долго и при этом без особых затрат и каких-либо ресурсов. В других же направлениях присутствуют свои ограничения и условия в виде обязательных лицензий и дипломов.

Повторюсь, это все очень субъективно и основано на личных ощущениях, потому что каждый живет по-своему, и существует огромное количество факторов, влияющих на итоговый результат. У меня семья из трех человек, и в ней я выступаю основным

источником дохода, в вашем же случае все может быть по-другому. Но в целом все выглядит примерно так.

Итак, можно сказать, что программист получает неплохие деньги даже в самом начале. И здесь особенно важно понимать, что время на вашей стороне. То есть чем дольше вы работаете программистом, чем больше усваиваете технологий и получаете опыта, тем больше денег можете зарабатывать. Кроме того, не забывайте о таком плюсе, который интересует многих, как полное сопровождение при переезде в чужую страну, где уровень зарплаты может очень сильно поменяться в лучшую сторону.

В качестве небольшого итога я хотел бы еще раз констатировать, что в самом начале карьеры разработчик не будет получать очень много. Поэтому если вы до смены профессии зарабатывали большие деньги, то вы почувствуете разницу. И если у вас есть какие-либо кредиты, то этот фактор нужно брать во внимание, прежде чем менять профессию.

Тем не менее стартовые зарплаты разработчиков зачастую выше стартовых зарплат в других областях. Плюс, поработав на реальном проекте год или два, вы можете рассчитывать на то, что ваша ценность как специалиста вырастет в разы, и рост в зарплате не заставит себя ждать.

Если брать в долгосрочной перспективе, то тут время на вашей стороне. Ваша зарплата будет расти постоянно, и в конечном счете наступит момент, когда она с легкостью обгонит показатели зарплаты в других областях. То же самое и с показателями стабильности. Через три-четыре года в профессиональной разработке перед вами откроются все те возможности, о которых я писал в начале этой главы. Поэтому помните: чем раньше вы начнете обучение, тем быстрее вы получите все те бонусы, которые предлагает данная профессия.

Обязательно ли техническое мышление?

Как я говорил ранее, я имею образование по специальности «менеджмент», которая не имеет ничего общего с программированием. Я не изучал никаких технических предметов и, соответственно, не мог указать в своем резюме, что разбираюсь в них. Поэтому в данном разделе я поделюсь тем, как я все-таки решил этот вопрос при минимальных затратах сил и времени и стоило ли это того.

Нужно ли с детства быть технарем, чтобы стать программистом? Нужно ли иметь математический склад ума? Или же без этих качеств можно обойтись? Чтобы ответить на этот вопрос, я приведу пример из собственной жизни и расскажу, какими были мои отношения с математическими дисциплинами. Для этого я приглашаю вас вернуться со мной в школьные времена. Начиная с седьмого класса я был в числе тех, у кого средней оценкой за год была тройка. Единственные хорошие оценки в школьные годы у меня были по физкультуре, английскому языку и истории. Эти три предмета мне нравились, и я с удовольствием на них ходил. Что касается математики, то сказать, что отношения были плохими, — это ничего не сказать. Оценки по алгебре и геометрии, особенно в старших классах, у меня были между двойкой и тройкой. И я очень радовался, когда за контрольную получал три.

Когда я окончил школу и пришла пора поступить в университет, единственным однозначным критерием, который я ясно держал в голове, было отсутствие математики и технических наук в экзаменационной программе. В итоге я поступил на специальность

менеджера. В университете нам преподавали в основном гуманитарные предметы, и лишь малая их часть касалась математики или программирования. Закончив университет и открыв свой диплом, я не смог найти ни одной четверки напротив предмета, связанного хоть как-то с математикой. Напротив предмета «программирование» тоже стояла тройка.

Одним словом, в свои юные годы я ни за что не поверил бы в то, что когда-нибудь стану программистом, а тем более буду писать на эту тему книгу. Единственный плюс, который у меня был к моменту начала изучения, — это то, что компьютер всегда присутствовал в моей жизни. С самого детства я любил играть в компьютерные игры. Я знал основы интернета и разбирался в том, как это все работает на уровне пользователя. Я был из тех детей, кто хоть раз переустанавливал Windows или подключался к интернету через кричащий модем, занимая телефонную линию. Помимо этого, благодаря тому что я в основном играл в ночное время, когда свет в комнате был выключен, я был вынужден запомнить расположение клавиш на клавиатуре. Поэтому сейчас я могу спокойно печатать на русском и на английском языках вслепую, не глядя на клавиши. Этот навык действительно ускоряет работу и облегчает жизнь, когда часто работаешь на компьютере. Но в остальном никакого опыта в плане написания кода, никаких заслуг в области математики и других точных наук у меня не было. Мало того, я вообще не любил это направление и никогда не планировал связывать с ним свою жизнь. Поэтому могу с уверенностью сказать, что стать программистом без технического образования и знаний в области математики возможно.

Да, существуют направления, где без математики не обойтись. Но если мы говорим о мобильной разработке или разработке сайтов, тут вполне достаточно базовых знаний. Поэтому перед тем как заняться изучением программирования, узнайте, в каких

направлениях можно обойтись без знания математики. Программирование — это больше про логическое мышление, внимательность и умение пользоваться инструментами. И все эти навыки приходят с опытом изучения и написания кода. Причем их можно начинать прокачивать, не имея никакого опыта, то есть полностью с нуля.

Все навыки, которые необходимы программисту, можно приобрести без специальной подготовки. Тут как с игрой на гитаре: изначально никто не умеет этого делать, но каждый может научиться, если потратит время. Представьте, что вы начинающий гитарист, который медленно перебирает пальцами струны, постоянно подсматривая в тетрадку с аккордами. Но через десять лет опыта вы уже сможете виртуозно владеть инструментом, воспроизводя любую популярную мелодию, и пальцы будут свободно перемещаться по грифу. Я веду к тому, что навыки в области программирования, которые вам нужны, так же подлежат изучению и прокачиванию, как и навыки в других сферах. И никакой особенный склад ума здесь не нужен. Нужно просто набираться опыта, обучаться и писать много кода.

Но в изучении программирования очень важен правильный подход. Так что если вы не знаете, с чего начать, то можно использовать техники, приведенные мной в этой книге, адаптировав их под себя.

Как найти работу, если нет технического образования?

Для ответа на этот вопрос давайте рассмотрим рисунок, который я привел ниже.

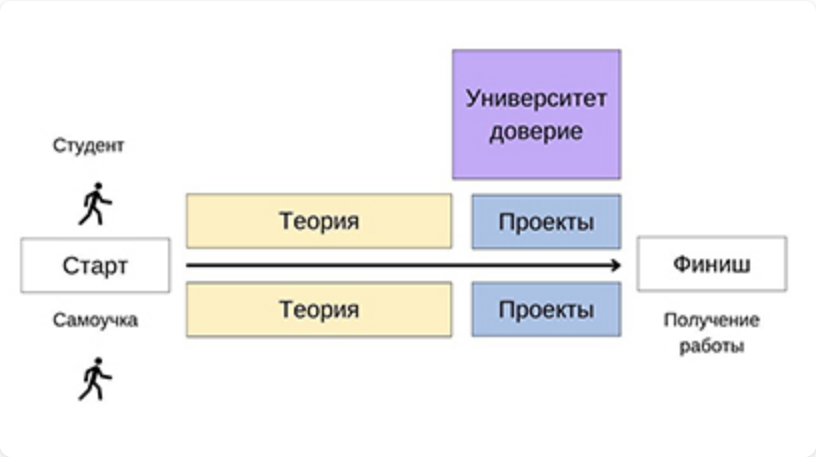


Рис. 35. Сравнение процесса обучения в вузе и самостоятельного обучения

Я попытался изобразить путь человека, который учится в университете, и человека, который изучает программирование самостоятельно. У них одна и та же цель: выучиться и получить работу программиста. Как вообще выглядит этот путь? Сначала человек получает теоретические знания, которых должно быть достаточно, чтобы начать создавать свои собственные маленькие проекты. Эти проекты показываются работодателю как подтверждение, что навыки программирования освоены в достаточной степени. Как вы можете заметить, сама структура пути в обоих случаях абсолютно одинаковая. Поэтому на моем графике и студент, и самоучка на своей стороне имеют желтый блок с теорией и синий блок с практическими проектами. Разница в том, как они строят данные блоки.

Обратите внимание, что на стороне студента мы видим блок фиолетового цвета, который означает университет. А университет в глазах работодателя в первую очередь ассоциируется с доверием. Да, вузы бывают разные, и качество образования везде разное, но в принципе в каждом из них человек на протяжении нескольких лет получает структурированные базовые знания. Однако это не гарантия того, что студент получает качественные навыки, которые будут полезными на реальном проекте, и об этом работодатель тоже знает. И этот параметр во многом зависит от конкретных учителей и методик, которые используются в том или ином учебном заведении. Поэтому при оценке престижности и рейтинга университета работодатель может сделать выводы о том, насколько высок уровень желтого и синего блоков у человека, который претендует на вакансию, — другими словами, насколько хорошо ученик прокачал теорию и практику. Так что репутация университета играет важную роль. Он выступает гарантом того, что человек действительно потратил определенное количество времени на техническое образование, и тому есть подтверждение в виде диплома.

В случае с человеком, который изучил весь материал самостоятельно, таких гарантий нет, потому что весь процесс построения блока с теоретическими материалами и реализованными проектами организовывает сам учащийся. Такой подход изначально вызывает недоверие работодателя, ведь каждый человек индивидуален, и процесс изучения складывается у каждого по-своему.

Я понимаю, что сейчас существует большое количество различных сертифицированных курсов, но их так много, что подлинность и качество этих курсов тоже вызывают много вопросов.

Как мы видим, в случае самостоятельного изучения материала человек может иметь меньше доверия со стороны потенциального

работодателя. Но это не значит, что люди без диплома не смогут найти работу. Я — один из этих людей, и знаю много примеров, когда мои подписчики, в том числе благодаря моему блогу, смогли без труда найти работу.

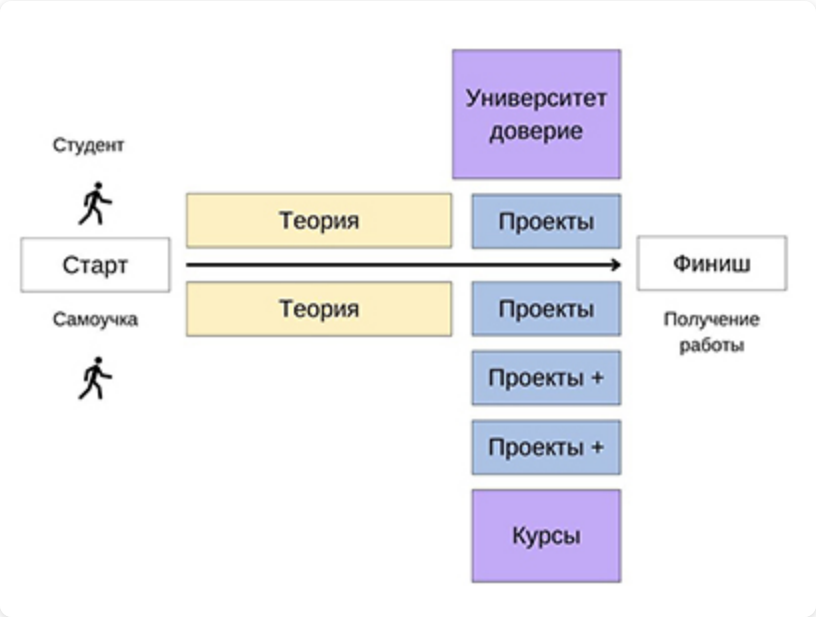


Рис. 36. Оптимизация процесса самостоятельного обучения

На примере рис. 36 я хочу показать, какие блоки еще можно добавить, чтобы повысить шансы на успех.

Первое, что вы должны понимать: основным критерием найма на работу является уровень прокачанности синего блока, то есть уровень сложности проектов, над которыми вы работали. Это тот блок, на который нужно ставить акцент. И ваша основная задача состоит в том, чтобы прокачать этот синий блок настолько сильно, чтобы он перевешивал подобные блоки студентов из университетов. Тут у меня было преимущество: в отличие от студентов, я мог сам составлять себе программу и внедрять те технологии, которые мне были нужны. Студенты же должны строго следовать программе курса, и если курс устарел, то у человека нет выбора, кроме как

продолжить учебу. Я же всегда изучал только то, что действительно нужно и востребовано, не тратя время на устаревшие технологии.

Второе, на что нужно обратить внимание, — это наличие сертификатов с курсов. Ранее я упомянул о различных курсах, которые в короткие сроки обучают вас специальности. Да, они не имеют большого веса. Но все же наличие таких сертификатов лучше, чем их отсутствие. Я всегда предпочитал курсы на базе крупных учебных заведений, имя которых на слуху. Сертификат в резюме определенно придаст ему веса.

Но все же даже крутая школа или курсы не дадут гарантий получения качественных знаний. В любом учебном заведении может попасться плохой преподаватель, а также некачественно составленный материал. Либо сама программа может быть плохо структурирована. Мне встречались курсы, где в группе было по тридцать человек, и они просто писали лекцию под диктовку учителя. Поэтому в плане выбора курсов у меня нет однозначных советов, нужно включать голову и анализировать информацию, которую вам предоставляют.

Что касается меня, то я проходил и обучение в крупной онлайн-школе, и индивидуальные курсы. Плюсы и минусы были и там и там, но в результате к своему первому резюме я прикрепил порядка шести сертификатов по программированию, которые накопил за время обучения.

Но повторюсь еще раз, что самый важный пункт — это проекты. Я делал их на основе материалов курсов, на которых учился. Помимо этого, я усложнял их с технической точки зрения. До того как начать внедрять что-то новое, я шел на сайт с вакансиями и смотрел списки технологий вакансий по Android-разработке. В основном технологии, указанные в разных вакансиях, совпадали. Поэтому я составил список технологий, которые должен был внедрить в свои

проекты. И благодаря этому списку и такому подходу проекты в моем резюме имели все технологии, которые требовала вакансия.

Моей целью было внедрить и попробовать все технологии, которые повторялись в вакансиях. И с таким подходом я знал, что рано или поздно сложности моих проектов будет достаточно, чтобы на мое резюме обратили внимание.

Так и произошло. Мое резюме стало чаще выскакивать в поиске у рекрутеров, потому что его содержание было полностью оптимизировано под поисковые запросы и требования. А на этапе просмотра резюме рекрутер понимал, что требования к навыкам по большинству пунктов подходят, поэтому я получал от рекрутера звонок. И уже во время собеседования я мог не просто сказать, что знаю ту или иную технологию, а уточнить, в каких конкретно проектах я ее использовал, был готов предоставить ссылку на опубликованное приложение и на код, который показывает, как я осуществлял реализацию.

Итак, уровень прокачки моего синего блока с проектами стал выше синего блока любого студента без опыта работы, поэтому я смог попасть в свою первую проектную команду. Таким образом, ответ на вопрос «Что делать, если нет технического образования?» звучит так: прокачивать синие блоки, то есть снова и снова делать проекты, набираясь практического опыта и параллельно наполняя свое портфолио.

Возможно ли на постоянной основе работать удаленно?

Я начал работать программистом, когда мир еще не знал, что такое COVID-19. Я застал время, когда приходилось ездить в офис, ходить на совещания и планерки. Но сейчас, после того как пандемия, а потом и всеобщий мировой кризис, вызванный известными событиями, внесли существенные изменения в наш образ жизни, все стало совсем по-другому. Мир уже не будет таким, каким он был до 2020 года. И одна из глобальных перемен — это переход многих компаний на удаленную систему работы. Удаленка существовала и раньше, но теперь, после пика пандемии и принудительной перестройки многих производственных процессов, эта схема стала практически повсеместным стандартом и воспринимается как норма. И программирование — это одна из тех специализаций, которая не сильно пострадала от этой ситуации, а то и вовсе выиграла.

Я работаю удаленно уже более двух лет. За это время я ни разу не видел ни одного участника своей команды вживую, но тем не менее не чувствую острой нехватки реального общения с коллегами. Поделюсь своими ощущениями от входа в новую команду на удаленной основе.

Процесс знакомства с командой при стандартном сценарии происходит таким образом, что ты встречаешь людей, жмешь им руку, перебрасываешься парой фраз, и уже к обеду примерно понимаешь, с кем тебе предстоит работать. В первые недели ты анализируешь их личности и уже знаешь, к кому обращаться, а к кому нет. Приблизительно понимаешь энергетику человека,

визуально запоминаешь его, и у тебя складывается какая-то общая картина.

Но при входе в новую команду на удаленной основе возможностей составить личное впечатление не так много. Обычно ты подключаешься к рабочему звонку и видишь список участников. Вместо реальных людей наблюдаешь аватарки их профиля в чате, а у некоторых картинка может и вовсе отсутствовать. Быстрое представление участников — и сразу переход к текущим делам. Удаленно познать свою команду намного сложнее, потому что через чат не чувствуешь энергию и характер человека. То есть изначально у меня нет никакого понимания о том, с какими людьми я буду работать. Так что если вы новичок, мне кажется, вы тоже поначалу будете испытывать определенные сложности при удаленной работе. Плюс ко всему в начале работы над проектом обычно возникает большое количество организационных вопросов, а у новичка — еще и технических. Может появиться ощущение, что ты не часть команды, а просто человек, открывающий ноутбук, пишущий код и закрывающий ноутбук до следующего дня. Я привык к этому чувству, и меня оно устраивает, но если для вас это критично, то поначалу вам точно будет некомфортно. Также не забываем о том, что многие не любят находиться дома все время. Им необходимо периодически выходить в офис, чтобы пообщаться, посмотреть на других, обменяться новостями. Наверное, таким людям я не советовал бы рассматривать удаленные варианты, тем более что и офисной работы в мире все еще достаточно.

Ну а для тех, кого не пугает чувство одиночества и отстраненности от команды, перечислю плюсы, которые лично для себя я нахожу в удаленной работе.

1. *Наличие большего количества свободного времени, нежели при работе из офиса.* По сути, речь идет о свободном графике. Благодаря

тому что у меня есть конкретные задачи, которые я должен выполнить, я сам могу выбирать время, удобное мне для работы. Например, я могу спокойно в середине рабочего дня выделить пару часов на личные дела, а потом просто закончить рабочий день на час-два позже. Благодаря этому я могу чаще общаться со своей семьей. У меня появилась возможность возить сына на спортивные занятия или забиратьпосле школы и принимать больше участия в его развитии. А также появилась возможность помогать супруге по дому, если такая помощь требуется. Кроме того, я скинул лишний вес, потому что свободный график позволяет мне ходить в зал и на прогулки, да и вообще больше заниматься своим здоровьем.

2. *Возможность питаться домашней едой*. Этот пункт, быть может, актуален не для каждого. Но благодаря тому что я смог регулярно питаться дома, а моя супруга всегда готовит вкусную и здоровую пищу, мое здоровье и самочувствие улучшились. Ну и, конечно, ушли лишние траты на покупку еды в закусочных.

3. *Вместе с улучшением здоровья выросла и способность переносить болезни*. Раньше, когда я заболевал простудой, мне все равно постоянно приходилось ездить в офис. Моя болезнь развивалась по нарастающей, и на выздоровление требовалось гораздо больше времени. Сейчас же, благодаря тому что я дома, при появлении каких-либо признаков простуды я устраиваю себе максимально щадящий режим. В обеденный перерыв могу поспать, выпив лекарства. Поэтому болезнь не развивается до критической отметки, и я начинаю чувствовать себя хорошо уже через пару дней. Да и в целом болеть я стал намного реже.

4. *Помимо экономии большого количества времени, я наблюдаю довольно ощутимую экономию денег*. Во-первых, меньше затраты на содержание автомобиля. Раньше у нас в семье было два автомобиля: одним пользовался я, другим — моя жена. Но с переходом на

удаленную работу необходимость иметь два автомобиля отпала, а это существенная экономия на страховке и обслуживании.

Вот такие плюсы я выделил лично для себя. Хочу уточнить, что все они прилагаются к удаленной работе не сами по себе. Благоприятные перемены произошли со мной не благодаря тому, что я сижу дома, а потому что я использую те возможности, которые дает удаленная работа.

В целом работа из дома мне очень нравится, для меня плюсов тут явно больше. Но все же я до сих пор не смог до конца привыкнуть к ощущению, что общаюсь и работаю с людьми, не видя их в реальной жизни.

Как найти работу за границей и переехать?

Один из плюсов профессии программиста, особенно для людей на постсоветском пространстве, — возможность переехать в другую страну. Многие специально идут в IT, чтобы найти работодателя за границей. И естественно, у новичков часто возникает множество связанных с этим вопросов. Я сам прошел этот путь и уехал, поэтому с удовольствием поделюсь информацией.

Самая распространенная ошибка начинающих программистов — думать, что они легко заинтересуют иностранного работодателя, едва закончив обучение. На основе собственного опыта и разговоров с рекрутерами я все-таки пришел к выводу, что вероятность получить работу за границей без какого-либо стажа очень мала. Сначала нужно хоть какое-то время проработать в своей стране. Я проанализировал несколько вакансий, где сотруднику обещают оказать помощь в эмиграции, и требуемый опыт кандидата в этих вакансиях — от трех до пяти лет.

Итак, опыт — это основное, на что смотрят потенциальные работодатели. Но речь идет не просто о каком-то определенном количестве лет, в течение которых вы работаете программистом. Работодатели также обращают внимание и на то, в качестве кого вы работали, какие проекты реализовывали, было ли у вас дополнительное образование и прочее.

Исходя из этого опыт можно разделить на три группы.

1. Опыт, полученный при обучении. Это могут быть какие-то бесплатные проекты, которые реализуются в ходе изучения той или

иной технологии. Опыт можно получить как на курсах, так и самостоятельно. Я бы добавил сюда и разработку собственных проектов, но только тех, которые не принесли денег и не выросли до таких масштабов, чтобы их можно было считать стартапами. Такие проекты еще называют pet-projects (в переводе с английского — «проекты-питомцы»). Плюс этой категории заключается в том, что вы можете реализовывать эти проекты самостоятельно. Но они не дают вам опыта взаимодействия в команде с другими разработчиками, а также с заказчиком, который будет тестировать конечный результат. Когда я учился, я делал упор на этот тип проектов, то есть самостоятельно писал приложения и усложнял их настолько, насколько мог. Я делал это для того, чтобы создать себе максимально сильное портфолио, которое доказывало бы мой уровень навыка работы с технологией и кодом.

2. Опыт, полученный на проектах, реализованных по схеме фриланса. От стартовых проектов их отличает то, что тут уже есть заказчик, который выдвигает определенные требования к результату, а вы получаете опыт не только реализации и использования технологии, но и доставки результата разработки, а также исправления ошибок. Большой минус здесь в том, что фриланс-проекты имеют более низкий рейтинг доверия, чем работа по найму, так как качество таких проектов сложнее проверить. Более того, на фрилансе заказчик зачастую уделяет крайне мало внимания качеству кода. Зато фриланс-проекты начинающему разработчику получить легче, нежели обрести постоянную работу в компании. Первыми заказчиками могут стать ваши друзья или знакомые.

3. Опыт, полученный в команде других разработчиков. Он ценится больше всего, потому что это значит, что кто-то уже проверял ваш код, кто-то уже работал и общался с вами в команде. Компания несет минимальные риски при найме такого сотрудника. Поэтому

первый коммерческий опыт способен открыть перед вами очень много дверей.

Итак, подводя промежуточный итог, хочу подчеркнуть, что, получив определенное количество опыта в коммерческой разработке, в один момент вы можете стать профессионалом, востребованным во всем мире. О каком именно временном промежутке идет речь? Я бы сказал, что в среднем речь идет о двух годах профессиональной разработки. Поэтому если одним из важнейших факторов мотивации для вас является переезд в другую страну, то чем раньше вы начнете обучаться, тем раньше вы там окажетесь. Допустим, если с этого момента вы потратите два года на обучение программированию и затем два года на работу, то уже через четыре года вы сможете задуматься о выборе страны для переезда…

И тут самое время поговорить о том, какие страны считаются наиболее привлекательными в плане эмиграции. Мне не удалось найти детальной статистики по данному вопросу, но я смог собрать некоторое количество информации, которая может помочь в создании хоть какой-то общей картины.

Естественно, в качестве основных направлений для переезда люди выбирают такие, как Европа, США, Канада, Австралия. Давайте рассмотрим их более детально. Сразу хочу оговориться, что я не специалист в области иммиграции и все данные брал из открытых источников, так что прошу воспринимать эту информацию не как руководство к действию, а лишь как общие сведения, от которых можно оттолкнуться, чтобы в дальнейшем изучить вопрос самостоятельно либо с помощью специалистов.

- *США* — лидер по количеству нанимаемых иностранных программистов. IT-рынок в этой стране просто огромен, и на нем представлены все лидирующие IT-компании мира. Я не буду углубляться в процесс переезда и иммиграции, об этом

нужно писать отдельную книгу. Но скажу, что основной способ переезда в США для программистов — это виза H1B, которая оформляется, если вы выиграли в лотерею. Лотерея проводится среди всех желающих уехать IT-специалистов каждую весну, и, естественно, их количество превышает квоту. Поэтому та часть, которая не прошла, может попробовать испытать удачу в следующем году. Если же кандидат выиграл в лотерее, то начинается процесс оформления бумаг, который может занять несколько месяцев. В итоге, если лотерея прошла в апреле, то человек может начать работу осенью. Это я дал очень краткий обзор, просто для того, чтобы вы понимали, как это работает.

- *Европа*. Здесь сориентироваться немного сложнее, потому что у каждой страны свои программы и свои законы. Условия в этих программах часто меняются, поэтому нужно узнавать все детали индивидуально, исходя из того, какие страны вам интересны. Одно я знаю точно: в северных странах Европы работы для IT-специалистов намного больше, чем в южной части. Лидером по части возможностей для программиста переехать и получить право на работу считается Германия. Если бы я выбирал, куда можно поехать, я бы присмотрелся к таким странам, как Великобритания, Швеция и Нидерланды. Одним из главных преимуществ этих стран стало достаточно широкое распространение английского языка. То есть, зная английский язык, можно спокойно жить и работать в этих странах. Также существуют общеобразовательные школы, где уроки ведутся на английском.
- *Канада*. Чем мне нравится эта страна, так это тем, что у них есть четко прописанная программа иммиграции, основанная на балльной системе. То есть чем больше баллов вы набрали, тем легче получить вид на жительство. И в этой программе есть один пункт, который позволяет получить большое количество баллов при условии работы на территории Канады в течение нескольких лет. То есть за каждый год вам начисляются баллы, и

через определенное количество лет, в зависимости от того, сколько баллов у вас было изначально, вы сможете получить вид на жительство. Вакансий в этой стране довольно много, особенно если мы сравниваем с южной частью Европы. Также работодатели изначально готовы искать сотрудников за рубежом.

- *Австралия*. Довольно специфический регион. В сравнении с другими странами вакансий здесь намного меньше. Несколько смущает и большая удаленность от крупных материков. Но в стране так же, как и в Канаде, действует балльная система для иммигрантов. То есть, проработав определенное количество лет на территории Австралии, вы получаете возможность претендовать на ВНЖ.

Отдельно хочу обратить ваше внимание на то, что переехать смогут далеко не все IT-специалисты. Есть такие направления, со знанием которых переехать будет сложно или попросту невозможно по причине большого количества предложений в той или иной области. Так что старайтесь более тщательно выбирать направление.

Что еще требуется для переезда, помимо собственно опыта и навыков? Разумеется, знание английского языка. Если вы собираетесь работать в иностранных компаниях удаленно или с переездом, вам так или иначе придется учить английский. Однажды я читал новость про одного сильного специалиста из Китая, которого взяли на работу без знания языка: его технический уровень был настолько высоким, что компания наняла ему переводчика. Но это скорее исключение из правил.

Даже если переезд и работа в иностранной компании не входят в ваши планы, нужно понимать другой очень важный момент: большинство документации, различные руководства и обучающие материалы созданы на английском языке. Поэтому рано или поздно

настанет такой момент, когда вы просто не сможете найти ответ на ваш вопрос в интернете на русском языке, и тогда вам придется искать на английском. Конечно, всегда можно воспользоваться Google-переводчиком, но процесс поиска решений будет идти гораздо дольше, чем если бы вы могли свободно читать и понимать по-английски.

Одним словом, владеть этим языком сейчас крайне выгодно и полезно. Не зная его, вы сами лишаете себя многих возможностей, в том числе и заработка в иностранной валюте в качестве программиста или специалиста IT-сферы.

Я получал много вопросов от читателей моего блога по поводу изучения английского языка, и всегда отвечаю, что этот навык нужен только вам. Мы не в школе, здесь не ставят оценки, и никто не ругает за пропуски уроков, никто не будет уговаривать или просить. Все очень просто. Хочешь — учишь, не хочешь — не учишь. Как бы то ни было, моя рекомендация такова: параллельно с программированием лучше начать учить английский, чтобы спустя полгода интенсивной работы вы могли бы по меньшей мере ориентироваться в выдаче поисковика без переводчика.

Есть ли место женщинам в программировании?

Это еще один вопрос, который мне часто задают читатели моего блога. Широко распространено мнение, что программирование — это сугубо мужское направление, и девушкам здесь не место. Но давайте обратимся к данным опроса с портала zippia.com[4].

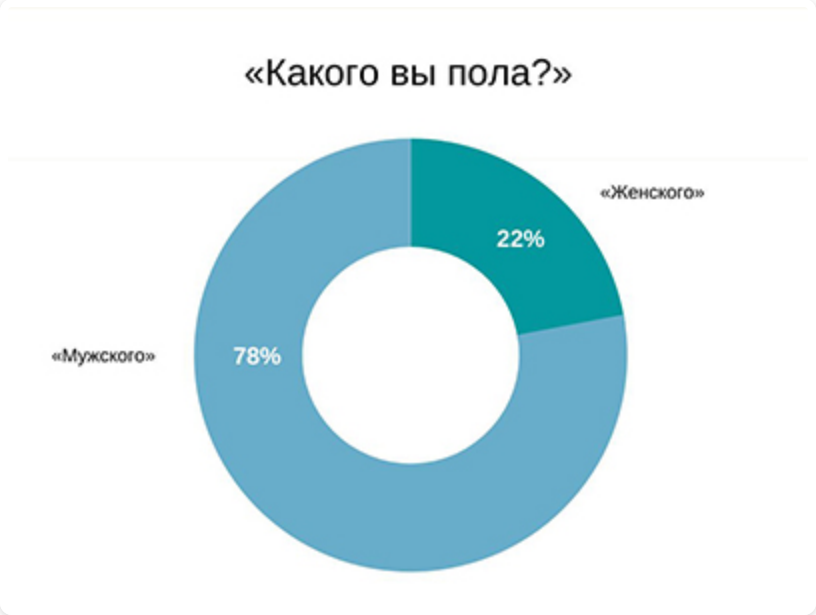


Рис. 37. Данные опроса о гендерных показателях

Судя по диаграмме, количество мужчин составляет 78%, что кажется мне довольно реалистичным показателем. Но тем не менее представителей женского пола тоже достаточно много, целых 22%. То есть, по сути, каждый четвертый разработчик в команде — женщина.

Если говорить о моем личном опыте, то мне в большинстве случаев доводилось работать в коллективах, которые состояли

преимущественно из мужчин. Но однажды я работал в команде по направлению web-разработки, где главным разработчиком и его заместителем были женщины. Места руководителя проекта, а также дизайнеров и маркетологов, тоже занимали представительницы прекрасного пола. По большей части им всем было уже за сорок, то есть это были опытные профессионалы, проработавшие в своей сфере более десяти лет. Получается, что они начинали свой путь, когда девушек среди программистов было еще меньше, но это не помешало им остаться в профессии и развиваться дальше. Поэтому я могу твердо утверждать, что команды разработчиков состоят не только из представителей мужского пола, и наличие женщин в таком коллективе уже никого не удивляет.

Многие девушки, решившие встать на путь программирования, опасаются, что рекрутеры будут принимать решение о сотрудничестве только по половому признаку. Я спешу их разубедить: главный фактор найма на работу программистом — наличие нужных навыков и портфолио. Первое, на что обращают внимание рекрутеры при открытии профиля кандидата, — это ваши навыки.

Также важно не забывать, что на данный момент огромное количество вакансий подразумевает работу из дома, то есть необходимость общения с коллективом сводится почти к нулю. Лично мне это позволяет чувствовать себя более комфортно в любом коллективе и быть более расслабленным, ведь при удаленной работе весь фокус смещается на то, как качественно я выполняю свои обязанности и насколько хорошо я поддерживаю коммуникацию. И никому нет дела до того, как я выгляжу, во что одет и какое у меня настроение. У меня бывает такое, что я за весь день ни разу не связываюсь с командой. При таких условиях работы гендерная принадлежность программиста вообще не имеет значения.

И последний важный момент, на который хотелось бы обратить внимание. В наше время, напротив, происходит популяризация найма представителей меньшинства в определенных областях. Эта тенденция особенно чувствуется в западных странах, таких как США и страны Европы. Сейчас рекрутеры специально стараются разбавлять коллективы и в мужские команды разработчиков ищут представительниц женского пола. Я как-то проводил совместный эфир с программисткой по имени Юлия, живущей в Испании, и она рассказала, что у них наличие девушки в команде является обязательным условием. Поэтому рекрутеры специально ищут девушек на должность разработчиков. Кстати, узнать больше о том, как Юля выучилась на программиста, и о ее опыте в поиске работы в Испании вы можете на моем YouTube-канале.

Итак, мы можем прийти к выводу, что женщинам совершенно нечего бояться. Наличие в команде девушки-программиста уже не считается чем-то необычным и воспринимается нормально. Мало того, если вы планируете работать за границей, у вас, возможно, будет даже преимущество, ведь в некоторых странах это даже всячески приветствуется. Ну и основные факторы — хорошее портфолио и владение соответствующими навыками.

А можно ли стать программистом, если тебе уже за сорок?

Мне всегда казалось, что IT — это сфера для молодых ребят. Когда я слышал словосочетание «команда разработчиков», в моем воображении возникала группа активных молодых людей с ноутбуками, полных энергии и готовых покорить этот мир. Но на самом деле дела обстоят совсем не так. В сфере IT найдется место людям любого возраста. Работая на реальных проектах, я общаюсь как с теми, кому только что исполнилось двадцать пять, так и с теми, кому уже идет шестой десяток.

Давайте приведу пример из статистики, которая была создана на основе анализа тридцати миллионов профилей компании Zippia в США. В этом аналитическом отчете содержится много интересной информации, поэтому я советую ознакомиться с ним, но сейчас нас интересует именно возраст[5].

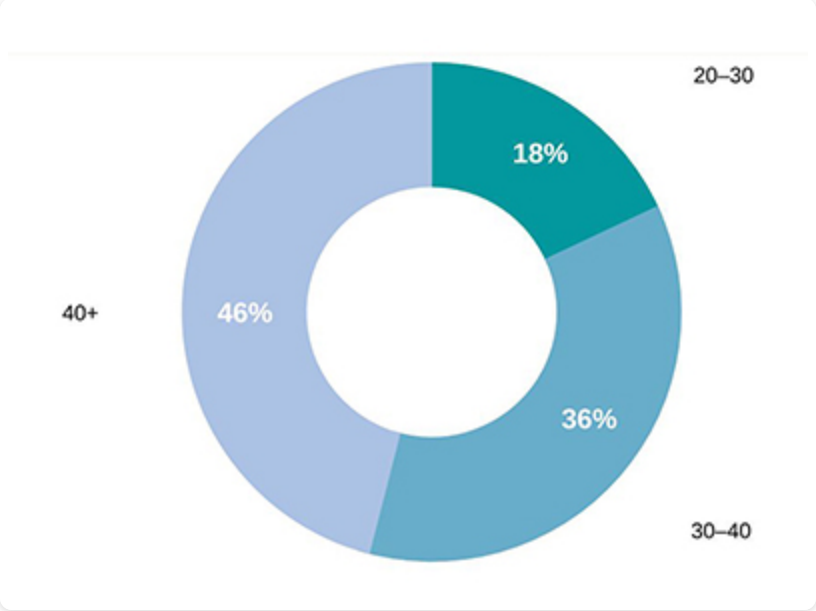


Рис. 38. Возраст программистов в США

Как вы можете заметить из графика выше:

- 46% разработчиков — это программисты старше сорока;
- 36% — это люди в возрасте между тридцатью и сорока годами;
- и только 18% — это молодежь до тридцати лет.

То есть средний возраст программиста — 39 лет. Удивлены? Я да!

Сейчас, проработав в данной сфере уже какое-то время, пройдя огромное количество собеседований, я бы хотел поделиться несколькими выводами. Ключевым фактором найма на работу всегда был и остается уровень навыка, а вовсе не возраст. Уровень навыка программирования очень легко проверить с помощью различных заданий. Также кандидату задают конкретные теоретические вопросы, ответы на которые он либо знает, либо нет. Поэтому результат технического собеседования полностью зависит от подготовки, а не от возраста. Второй немаловажный фактор — софт-скиллы, то есть то, как вы умеете работать в команде, справляться с проблемами, управлять своим временем. И человек с опытом, который уже трудился в какой-то другой сфере, всегда будет иметь здесь преимущество по сравнению с молодым специалистом, потому что он уже работал в команде, умеет справляться со стрессовыми ситуациями и находить кратчайшие пути выхода из кризиса. Смена профессии — это уже сам по себе показатель силы характера, которого не нужно стыдиться, наоборот, это следует уверенно указывать в своем резюме.

Теперь вы видите сами, что утверждение о том, что в этой сфере работают только молодые ребята, — миф. Приведенные выше данные и выводы указывают на то, что возраст не является основным критерием для принятия на работу. Основными критериями все-таки являются портфолио и человеческие качества.

Есть ли у программиста перспективы открытия собственного дела?

Когда-то в прошлом я имел опыт создания и раскрутки собственного бизнеса. Помимо этого, я по образованию менеджер и в целом могу сказать, что мне нравится это делать. У меня есть опыт и понимание того, как строится и работает малый и средний бизнес. И начиная изучать программирование, я был настроен со временем создать собственный IT-стартап.

Глобально все виды бизнесов можно разделить на несколько типов. Первый — это бизнес, основанный на продаже товаров, а второй — на продаже услуг. Ну а третий, то самое направление, которое всегда было мне интересно, — это построение бизнеса на основе продажи собственного технологичного продукта.

Такой бизнес имеет ряд уникальных преимуществ.

- *Возможность масштабирования*. Вы и ваша команда можете находиться в любой точке земного шара, там, где вам нужно, а вашим продуктом при этом будут пользоваться во всем мире. Подобное невозможно реализовать с такой же легкостью в классической схеме продаж товаров и услуг.
- *Минимальные требования для запуска бизнеса*. Если ты программист, то ты можешь самостоятельно создать прототип и проверить работоспособность модели заработка без каких-либо вложений, кроме собственного времени. В классической схеме открытия бизнеса основное условие для старта — закупка товара, а для этого нужен бюджет. Также нужно думать о

> логистике, приеме платежей, обслуживании клиентов, продажах и так далее. Если речь идет о продаже услуг, вся сложность заключается в построении команды, которая нужна для их предоставления. Да, есть варианты перепродажи чьих-то товаров либо продажи услуг с минимальными затратами, когда владелец самостоятельно делает всю необходимую работу. Но в первом случае выручка минимальна, а во втором вы рано или поздно упираетесь в ограничения по количеству времени. Что касается IT-продукта, то тут достаточно сделать сам продукт, который будет работать за вас и обеспечивать приток пользователей, то есть всю работу будет делать программа.

Сейчас у меня есть бизнес-приложение, которое уже имеет клиентов в разных концах земли — США, Южной Америке, Европе, Индии и СНГ. В большинстве этих стран я ни разу не был, но с помощью оптимизации страницы приложения я смог показать свой продукт и продвинуть его во всех этих регионах. А теперь представьте, если бы речь шла о международном бизнесе в области товаров и услуг. Вы никогда не сможете сделать это так легко, и тем более в одиночку. На это уйдет огромное количество бюджета, который еще нужно найти. Не буду вдаваться в подробности, но очевидно, что разница в простоте запуска сайта/приложения и полноценного бизнеса огромна. На момент написания этой книги мой прототип уже продвинут в поисковиках и на маркетах, я уже вижу аналитику и отзывы реальных пользователей, что дает мне ясную картину того, что нужно доделать и в каком направлении двигаться. Благодаря этому у меня уже имеются договоренности с партнерами, которые хотят внедрить мою технологию в свою инфраструктуру, потому что их клиенты нуждаются в этом функционале. Также у меня уже есть реальные клиенты, с которыми получилось протестировать приложение, получить от них обратную связь и подтверждение того, что они готовы платить за приложение. Это все можно сделать самостоятельно, при наличии навыка программирования. Впрочем, я

решил ускорить процесс и нанять команду, которая помогает мне в написании кода. Все участники команды находятся в разных точках земли, все работают удаленно, со свободным графиком, и мне не надо волноваться по поводу аренды офиса и поиска сотрудников в конкретном городе.

Таким образом, благодаря навыку программирования я могу спокойно строить бизнес, основанный на технологии, не рискуя ничем. Даже если в итоге ничего не получится, я все равно приобрету крутой опыт в программировании и могу добавить проект себе в портфолио, а также рассказать о нем на собеседовании. Поэтому если кто-то из вас мечтает создать бизнес, любит руководить людьми и проектами, то вот вам еще одно преимущество в копилку программирования.

Заключение

Перед тем как подвести итог и сказать слова напутствия, хотелось бы поблагодарить вас за время, уделенное моей книге. Если она оказалась для вас хоть немного полезной, можно рассказать о ней в своих социальных сетях, посоветовать друзьям и знакомым.

В заключение хочу сделать несколько выводов. Самое главное, что вы должны понимать: программирование — такой же обычный навык, как и все остальные. Ему можно научиться точно так же, как когда-то вы учились водить автомобиль или плавать. Просто сам процесс изучения этого навыка очень специфичен и потребует от вас значительных усилий и серьезной дисциплины.

Второй вывод заключается в том, что программирование — это не только написание кода, и что программисты на работе занимаются многими вещами, помимо написания кода.

И последний, самый важный вывод, вытекающий из предыдущего: программирование — это в основном решение возникающих проблем. Это поиск нужной информации и ликвидация регулярно случающихся поломок. Будучи профессиональным программистом, я занимаюсь этим изо дня в день, и за это я получаю деньги. Поэтому, столкнувшись с какой-либо проблемой в процессе обучения, вы можете поздравить себя: вы уже занимаетесь тем, что делает профессиональный программист. Просто нужно потерпеть и решить большое количество подобных проблем. В определенный момент ваш уровень навыка вырастет настолько, что работодатель будет готов платить вам за это.

И в качестве финального напутствия — несколько ценных мыслей, которые помогут вам на этом пути.

- Первое, что нужно понять: вы не должны идти за ментором, который будет вас постоянно направлять и от которого все зависит. Вы должны идти по своему пути, а ментор будет помогать при необходимости.
- Чтобы начать программировать — надо программировать и писать код.
- Если идти, то рано или поздно дойдешь.
- Не считая и не оптимизируя затрачиваемое время, вы не контролируете процесс и повышаете риск не дойти до конца.
- Не существует «Я еще думаю». Есть либо «Да», либо «Нет». Пока вы думаете, вы выбираете «Нет».
- Навык программирования стоит всех жертв. Востребованность у работодателей по всему миру и свобода — это те призы, что ждут вас в конце.
- «Нет времени» — отговорка! «Уже слишком старый» — отговорка! «Нет денег на обучение» — отговорка! Если ищете отговорки, лучше просто откажитесь от идеи и живите в свое удовольствие.
- Процесс программирования похож на сборку авто. Надо знать, зачем нужна каждая деталь, как ее подключить и настроить.

Если этой книги вам оказалось мало, то на моем сайте вы сможете найти большое количество полезных бесплатных материалов, которые позволят вам стать программистом и воплотить в реальность все свои планы. Также вы можете обратиться ко мне в социальных сетях, и я обязательно помогу.

С пожеланиями удачи и вдохновения,

Борис ProIt

Контакты

Сайт: https://borisproit.expert/

Instagram: https://www.instagram.com/borisproit/

YouTube: https://www.youtube.com/@borisproit

Telegram: https://t.me/borisproit

Примечания

1. Подробнее ознакомиться с результатом исследований можно по ссылке: https://vc.ru/hr/102737-issledovanie-bolshe-poloviny-specialistov-prishli-v-it-iz-drugih-sfer.

2. В моем понимании мотивированное самообразование — это когда вы получаете образование самостоятельно, выбрав какую-либо конкретную цель, которой можно достичь, получив определенные знания. — *Прим. авт.* Вернуться

3. Это утверждение справедливо, по крайней мере, для front-end-разработчиков, многие профессиональные разработчики наверняка со мной поспорят. — *Прим. авт.* Вернуться

4. https://www.zippia.com/software-engineer-jobs/demographics/. Вернуться

5. https://www.zippia.com/software-engineer-jobs/demographics/#age-statistics. Вернуться