

Профессия «Программист»

С нуля до трудоустройства



Профессия "Программист"

С нуля до трудоустройства

Автор: Кирилл Мокевнин

Аннотация

Этот учебник — ваш путеводитель в профессию программиста. Он создан для тех, кто хочет начать с нуля и шаг за шагом дойти до трудоустройства в одной из самых востребованных и перспективных областей.

В книге подробно рассказывается о выборе направления в программировании, создании плана обучения, формировании портфолио, поиске вакансий и прохождению собеседований.

Книга поможет вам подготовиться к собеседованиям, разобраться в требованиях работодателей и пройти первые шаги на пути к карьерному росту. Здесь вы найдёте советы по составлению резюме, поиску работы, участию в сообществах и построению долгосрочного развития в профессии.

Сейчас вы читаете первую редакцию этой книги, которая будет значительно дорабатываться в процессе обратной связи. Написать ее можно на support@hexlet.io. Поддержка и обсуждение книги в нашей телеграм-группе <https://t.me/hexletcommunity>

Автор учебника – [Кирилл Мокевнин](#)

Содержание

<u>Аннотация</u>	2
<u>Содержание</u>	3
<u>Введение</u>	5
<u>Почему программирование?</u>	5
<u>Программирование в современном мире</u>	6
<u>Кто может стать программистом?</u>	7
<u>Мифы о программировании</u>	7
<u>Чтобы стать программистом нужно закончить вуз по этой специальности</u>	8
<u>Программирование слишком сложно для обычного человека</u>	9
<u>Чтобы стать программистом, нужно знать математику</u>	9
<u>Стать программистом после 40 нельзя</u>	10
<u>Программирование это профессия для мальчиков</u>	11
<u>Все программисты работают только за компьютером весь день</u>	11
<u>Выбор направления в программировании</u>	12
<u>Веб-разработка (Frontend, Backend, Fullstack)</u>	12
<u>Мобильная разработка (iOS, Android)</u>	14
<u>Разработка игр</u>	15
<u>Встроенные системы и IoT</u>	15
<u>Автоматизация и тестирование</u>	16
<u>1С</u>	16
<u>Не могу определиться</u>	17
<u>Как стать программистом</u>	18
<u>Программист с точки зрения компании</u>	18
<u>Главный критерий успеха</u>	20
<u>Готов ли я к трудуоустройству?</u>	21
<u>С чего начать?</u>	22
<u>Выбираем направление</u>	22
<u>Формируем план обучения</u>	23
<u>Формируем расписание</u>	24
<u>Находим единомышленников</u>	25
<u>Английский язык</u>	26
<u>Подготовка к практике</u>	27
<u>Редактор</u>	28
<u>Отладка (Дебагинг)</u>	30
<u>Изучение программирования</u>	32
<u>Основы программирования</u>	32
<u>Продвинутые техники</u>	35
<u>Окружение</u>	36
<u>Прикладные инструменты</u>	36

<u>Computer Science</u>	37
Искусственный интеллект (ChatGPT)	38
Опыт прикладной разработки	40
Учебные проекты	41
Пет-проекты	42
Открытые проекты	43
Волонтерские проекты	44
Где искать волонтерские проекты	44
Составление резюме	45
Как выделиться среди других кандидатов	46
Структура идеального резюме программиста	47
Рекомендации по оформлению резюме	48
Автофильтры	49
Поиск работы и прохождение собеседований	51
Где искать вакансии: платформы и сообщества	51
Как откликаться на вакансии?	52
Что проверяют на собеседованиях программистов?	53
Тестовые задания	54
Как вести переговоры о зарплате?	57
Точечное трудоустройство	57
Нетворкинг	59
Линкедин	59
Сообщества (группы, Помощь новичкам Q&A)	60
Хакатоны и воркшопы	61
Конференции	62
Как учиться эффективно?	64
Выбор курсов и материалов	64
Самообразование или наставник?	64
Избегаем выгорания	65
Развитие карьеры программиста	67
От джуниора до синьора: ключевые навыки	67
Выбор между продуктовой и аутсорсинговой компанией	68
Фриланс или работа в офисе?	69
Заключение	70
Программирование как дверь в будущее	70
Советы начинающим	70
Приложения	71
Рекомендованные книги, курсы и ресурсы.	71
Список полезных инструментов	71

Введение

Почему программирование?

Программирование открывает двери к свободе и возможностям, которые трудно найти в других профессиях. Удалённая работа, высокие зарплаты и возможность работать в любой точке мира — всё это привлекает тысячи людей к этой профессии. Программист не привязан к месту или условиям, его знания и навыки универсальны и востребованы в любой стране. В отличие от многих других специальностей, программирование дает глобальные перспективы, независимо от того, где вы находитесь.



И всё же главное, что объединяет программистов, — это страсть к созданию. Процесс проектирования и написания кода для многих из них становится настоящим удовольствием, которое не заканчивается в офисе. В свободное время они изучают новые языки программирования, экспериментируют с искусственным интеллектом или работают над личными проектами. Эта особенность часто повергает в шок других людей, которые после работы делают все что угодно, лишь бы это было как меньше связано с их работой.

Но программирование это не только высокооплачиваемая работа, но и возможность запустить свой проект. Программистам это сделать гораздо легче, чем остальным, так как, обычно, это самая дорогая и сложная часть в онлайн проектах. Не зря в среде разработчиков так много разговоров, событий и сообществ вокруг стартапов. А немалая их часть была создана бывшими технарями: google, yandex, vk, telegram и так далее.

Программирование в современном мире

Сегодня программирование стало основой всех сфер нашей жизни. От смартфонов и умных часов до космических спутников и технологий искусственного интеллекта — практически всё, что нас окружает, так или иначе связано с кодом.



Компании во всех отраслях, будь то медицина, образование или сельское хозяйство, активно внедряют технологии, чтобы стать более эффективными. Программисты создают инструменты, которые помогают врачам ставить диагнозы, учителям находить подход к ученикам, а фермерам повышать урожайность. Без их работы невозможно представить современную экономику: от интернет-магазинов до банковских систем.

Благодаря коду создаются виртуальные миры, интерактивные приложения, увлекательные игры и невероятные визуальные эффекты в кино. Люди из самых разных областей используют программирование, чтобы выразить себя и воплотить свои идеи в жизнь.

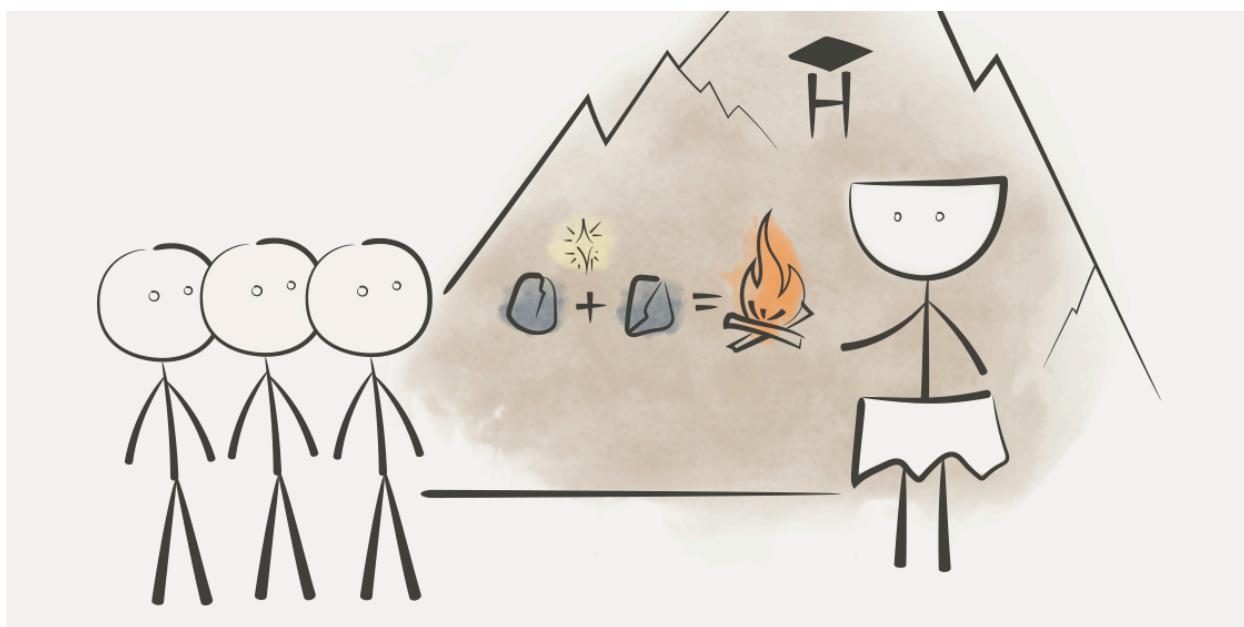
[По данным Минцифры РФ](#), нехватка IT-кадров оценивается в 500–700 тысяч человек, и прогнозируется, что этот дефицит может достигнуть 1 миллиона к 2027 году. Спрос на программистов продолжает расти. В период с января по август 2024 года в России было открыто более 120 тысяч вакансий для программистов и разработчиков, что на 6% больше по сравнению с аналогичным периодом прошлого года.

Речь конечно же идет в первую очередь про профессиональных разработчиков. Это порождает парадоксальную ситуацию, всем нужны опытные программисты, но их недостаточно, с другой стороны, на начальные позиции людей больше чем вакансий, потому что программирование обрело сильную популярность в последние годы. Поэтому несмотря на гигантский дефицит кадров, стать программистом сложнее чем может

показаться на первый взгляд. Станут лишь те, кто готов в это серьезно вкладываться. А то, что надо делать и как, вы узнаете дальше в следующих главах.

Кто может стать программистом?

Программистом может стать [практически каждый](#), кто готов учиться, работать над собой и не боится новых вызовов. Эта профессия не ограничена возрастом, образованием или профессиональным опытом.



Многие начинают программировать в школе или вузе, а другие приходят в эту профессию уже в зрелом возрасте, меняя карьеру в 30, 40 или даже 50 лет. Большинство работодателей ценят не возраст, а навыки, которые вы можете предложить.

Мифы о программировании

Программирование, как и любая другая популярная профессия, окружено мифами. Многие из них создают ложные барьеры для тех, кто только задумывается о смене карьеры или начале обучения. Давайте разберемся с самыми распространёнными.



Чтобы стать программистом нужно закончить вуз по этой специальности

[Согласно анализу платформы Stack Overflow](#), каждый год тысячи программистов начинают свою карьеру без профильного образования. На вопрос об образовательной подготовке около половины респондентов отмечают, что являются самоучками или проходили краткосрочные курсы. Работодателей чаще интересует не диплом, а ваши практические навыки: кодирование, умение решать задачи и адаптироваться к новым технологиям. В современных реалиях огромную роль играет участие в реальных проектах, а не формальное образование.



Программирование слишком сложно для обычного человека

Многие думают, что программирование — это что-то сверхсложное, доступное только избранным гениям. На самом деле, программирование — это навык, который развивается постепенно. Современные языки и инструменты делают этот процесс интуитивным, а огромное количество доступных материалов помогает освоить даже сложные концепции. При условии наличия желания, конечно же.



Чтобы стать программистом, нужно знать математику

Этот миф особенно популярен среди тех, кто невзлюбил математику со школы. Но правда в том, что для большинства направлений в программировании глубокие математические

знания не требуются. Например, веб-разработчикам или мобильным разработчикам достаточно базовых навыков логического мышления и работы с числами. Математика становится важной лишь в специфических областях, таких как машинное обучение, криптография или игровые движки. И даже здесь многие программисты осваивают нужную теорию уже в процессе работы.

Стать программистом после 40 нельзя



Этот миф возникает из-за стереотипов о возрасте в профессиях, связанных с технологиями. Но реальность показывает обратное. Многие люди осваивают программирование в зрелом возрасте и успешно находят работу. Тот же Stack Overflow отмечает, что возраст программистов варьируется от 20 до 60 лет и старше. Более того, зрелые специалисты часто обладают дополнительными навыками — управлением, аналитическими и коммуникативными, — которые делают их ценными сотрудниками. В программировании важен не возраст, а готовность учиться и адаптироваться.

Однако, фильтр по возрасту существует и некоторые компании, а скорее даже конкретные люди в этих компаниях стараются выбирать более молодых кандидатов. Эта особенность присуща не только программированию, но чем дальше тем лучше, потому что профессия становится старше и люди вместе с ней. Так постепенно мы приучаемся видеть разных людей вокруг себя. А когда-то мне было сложно представить себе 40-ка летних программистов и теперь я один из них :)

Программирование это профессия для мальчиков



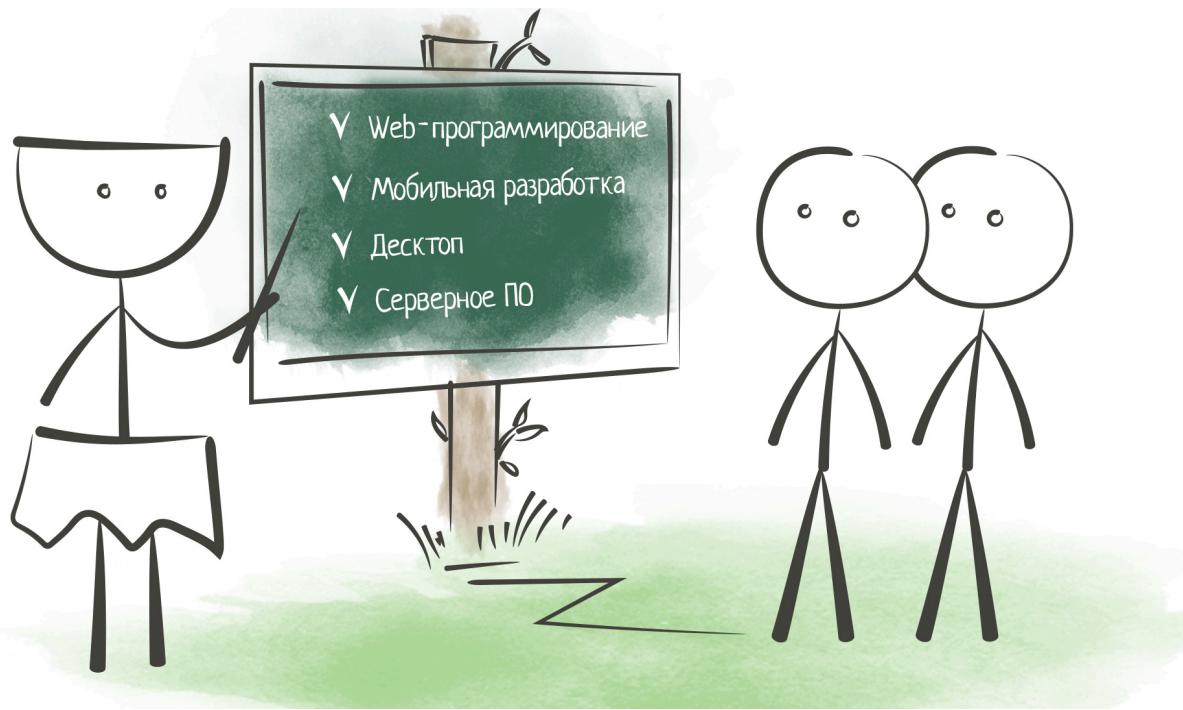
В программировании действительно работает больше мужчин, но это только по той причине, что девушки сами реже выбирают идти в разработку. Но если девушке это интересно и она включается в процесс, то у нее это получается ничуть не хуже.

Все программисты работают только за компьютером весь день

Программирование действительно связано с компьютером, но это далеко не единственная часть работы. Программисты часто участвуют в обсуждениях, встречах, проектировании систем, планировании, поэтому они сидят не только перед экранами мониторов, но и в переговорках. Это командная профессия, где общение играет важную роль.

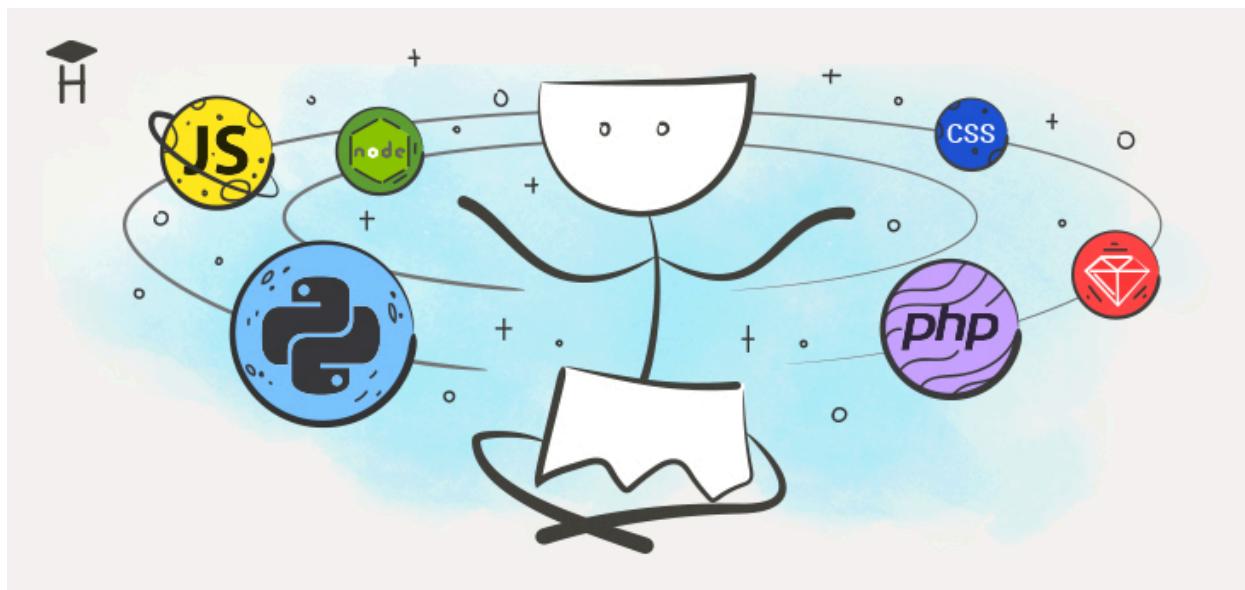
Выбор направления в программировании

Программирование — это многогранная сфера, где каждый может найти что-то по душе. Выбор направления во многом зависит от наличия вакансий, ваших интересов, долгосрочных целей и стиля работы, который вам близок. Рассмотрим ключевые направления, чтобы помочь вам определиться.



Веб-разработка (Frontend, Backend, Fullstack)

Веб-разработка — это создание сайтов и веб-приложений, а также обслуживающих их сервисов. Это одно из самых массовых направлений программирования, и его масштабы впечатляют. На 2025 год в интернете насчитывается более 2 миллиардов сайтов, а каждый день создаётся около 250 тысяч новых. Эти сайты могут быть как простыми визитками компаний, так и сложными платформами, такими как Amazon, Google или Яндекс, над которыми работают тысячи специалистов.



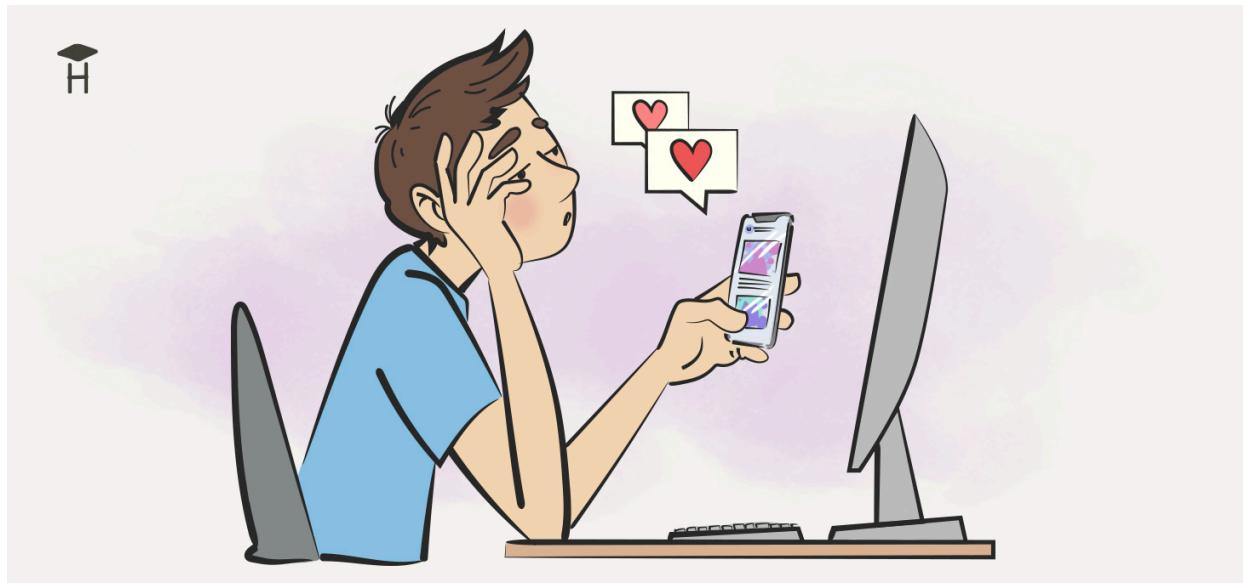
Современные сайты далеко ушли от простых текстовых страниц с гиперссылками, которые были на заре интернета. Сегодня это сложные веб-приложения, которые работают прямо в браузере и могут заменить настольные программы. Например, Photoshop теперь доступен в веб-версии, а инструменты вроде Google Docs становятся альтернативой привычным десктопным приложениям. Веб-разработка становится всё более сложной, но остаётся доступной для новичков, делая её популярным направлением для входа в профессию.

Веб-разработка делится на три основные специализации:

- **Frontend-разработка** занимается созданием визуальной части сайтов и приложений, то есть всего, что пользователь видит и с чем взаимодействует. Это работа с HTML и CSS для структуры и оформления страниц, а также с JavaScript (или его строгой версией TypeScript) для добавления интерактивности.
- **Backend-разработка** — это внутренняя часть сайтов и приложений. Она отвечает за хранение данных, выполнение бизнес-логики и взаимодействие с внешними системами. Backend-разработчики работают с языками программирования, такими как Python, Java, Node.js, PHP, Go или Ruby.
- **Fullstack-разработка** объединяет обе специализации, позволяя разработчику работать как с фронтендом, так и с бекеном. Fullstack-разработчики универсальны и могут создавать проекты целиком, от пользовательского интерфейса до серверной части. Многие начинают с одной из специализаций и постепенно расширяют свои знания, переходя во вторую область. Со временем опытные веб-разработчики часто становятся фулстеками.

Мобильная разработка (iOS, Android)

Мобильная разработка занимается созданием приложений для смартфонов и планшетов, которые работают на популярных платформах, таких как iOS и Android. Это направление продолжает активно расти, ведь в 2025 году в мире насчитывается более 6 миллиардов активных пользователей смартфонов. Люди используют мобильные устройства для всего: от общения и развлечений до покупок и работы.



Мобильные приложения бывают как простыми утилитами, так и сложными системами с миллионами активных пользователей. Например, приложения для банков или социальных сетей требуют огромных усилий не только в разработке, но и в поддержке. При этом многие компании стараются создавать мобильные версии своих веб-приложений, чтобы их сервисы были доступны на всех устройствах.

- **iOS-разработка** сосредоточена на создании приложений для устройств Apple. Здесь используются языки Swift и Objective-C (устарел), а сама работа строится в экосистеме Apple.
- **Android-разработка** предполагает работу с устройствами на Android. Основной язык для написания кода – Kotlin.
- Существуют также **кросс-платформенные решения**, такие как Flutter или React Native, которые позволяют создавать приложения сразу для обеих платформ. В них используются языки Dart и JavaScript.

Разработка игр



Разработка игр охватывает множество платформ, включая ПК, мобильные устройства, Xbox и PlayStation. Однако благодаря универсальным игровым движкам, таким как Unity и Unreal Engine, создавать игры можно с минимальными изменениями для разных устройств. Именно вокруг них и формируются направления:

- **Unity-разработка** использует язык программирования C# и подходит для создания как 2D, так и 3D игр. Unity популярен благодаря своей простоте, широким возможностям и удобным инструментам для разработки мобильных и инди-игр.
- **Unreal Engine-разработка** основана на языке C++ и используется для создания сложных, графически насыщенных 3D-проектов, включая AAA-игры. Unreal Engine предоставляет мощные инструменты для работы с визуализацией, физикой и эффектами.

Встроенные системы и IoT

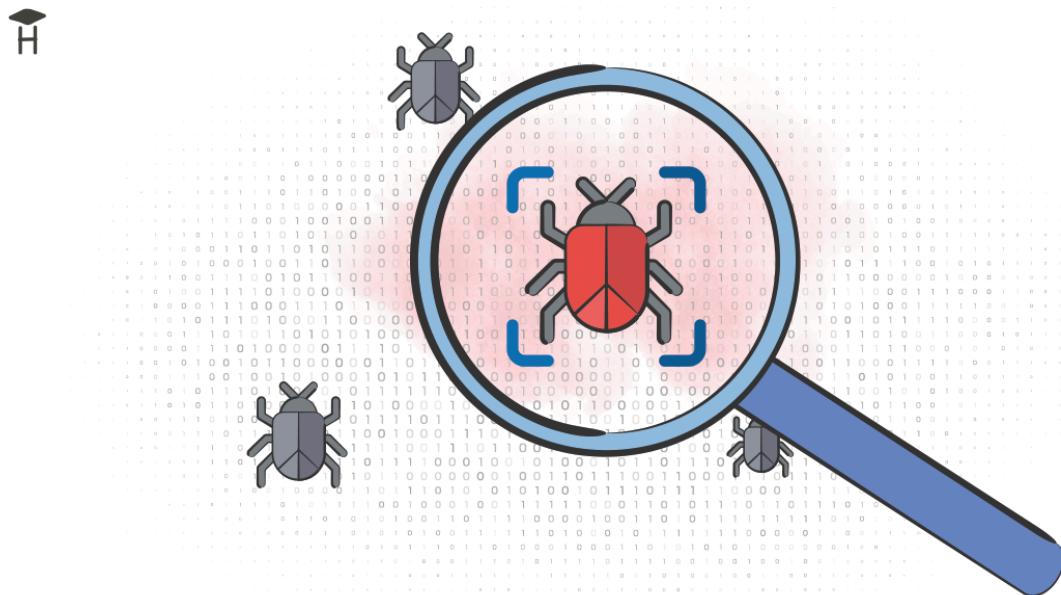
Разработка встроенных систем и IoT (Интернета вещей) связана с программированием "умных" устройств: от бытовой техники до промышленных датчиков. Эти устройства взаимодействуют с окружающим миром, собирают данные и обмениваются ими через сеть.

Работа в этой области включает написание низкоуровневого кода на С или С++, настройку оборудования и создание алгоритмов для обработки данных. Например, это может быть разработка прошивки для "умного дома" или системы мониторинга в автомобиле.

Встроенные системы и IoT подходят тем, кто любит работать на стыке программирования и электроники, создавая решения, которые становятся частью реального мира.

Автоматизация и тестирование

Тестирование и автоматизация — это направление, связанное с обеспечением качества программного обеспечения. Хотя тестирование и разработка отличаются по своей сути, специалисты по автоматизации тестирования являются полноценными программистами. Их работа заключается в создании тестовых сценариев, написании автоматизированных тестов и анализе работы системы для выявления ошибок и повышения надёжности приложений.



Для автоматизации тестирования используются специализированные инструменты, такие как Selenium и Playwright. Эти инструменты позволяют автоматизировать повторяющиеся проверки, что делает процесс тестирования быстрее и эффективнее

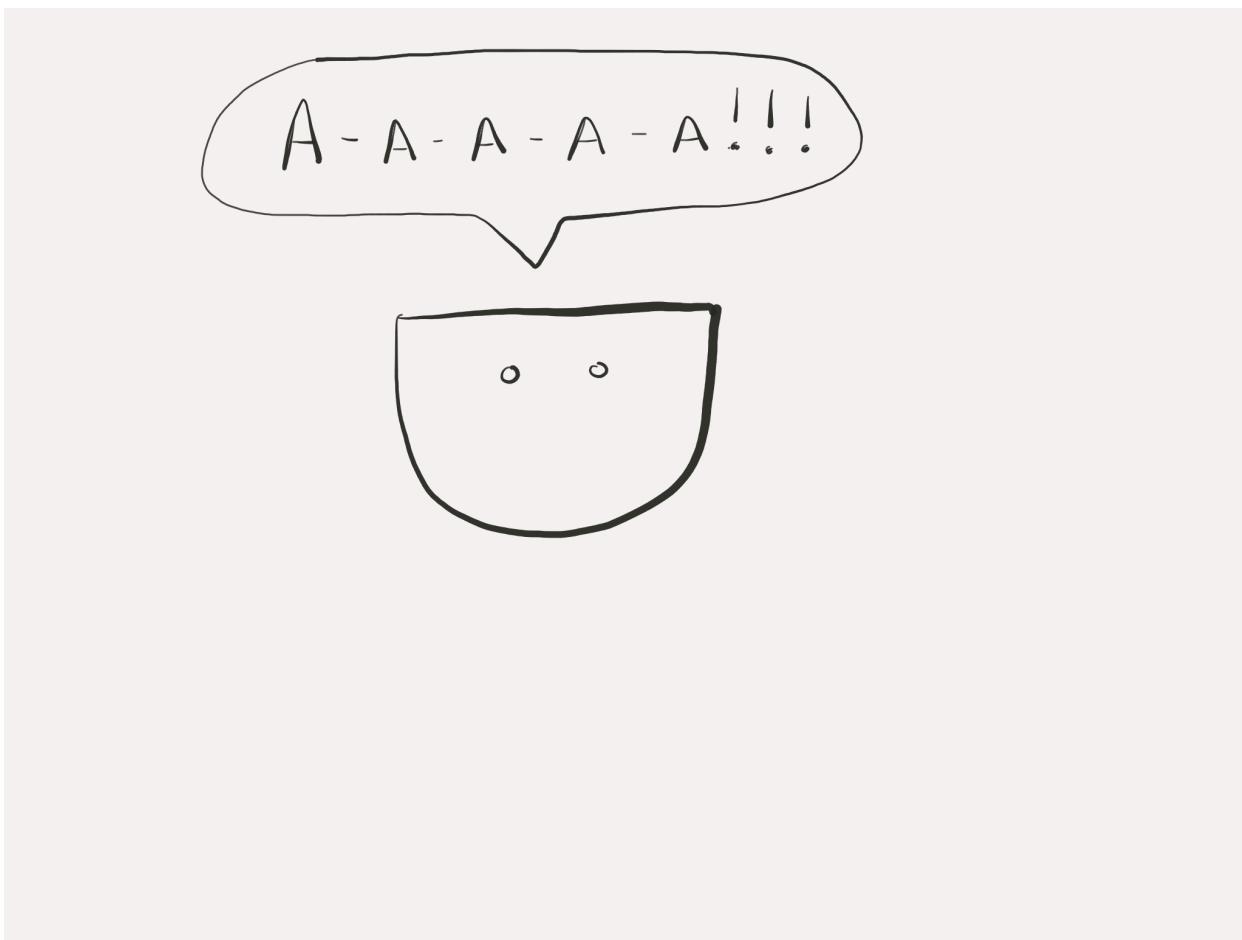
1С

1С-разработка — это направление, связанное с созданием и адаптацией программного обеспечения на базе платформы 1С:Предприятие. Эта система широко используется в России и странах СНГ для автоматизации бизнес-процессов: бухгалтерского учета, управления складом, продажами, кадрами и другими областями.

Работа 1С-разработчика включает разработку новых модулей, доработку существующих решений под нужды конкретного бизнеса, интеграцию с другими системами и поддержку пользователей.

1С-разработка требует знания языка программирования 1C:Enterprise, умения работать с базами данных (например, SQL), а также понимания структуры и логики работы платформы.

Не могу определиться



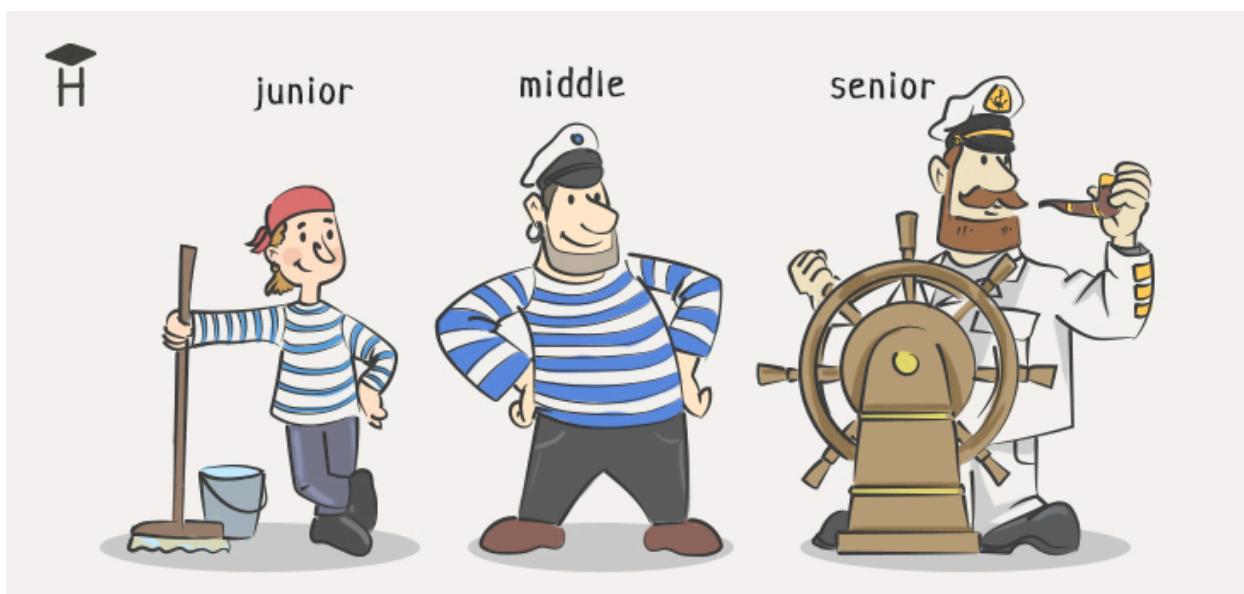
Если вы не знаете с чего начать, то следуйте этому плану:

- Проверьте как вам вообще заходит программирование через [интерактивные бесплатные курсы](#). Здесь вы можете попробовать множество разных языков без необходимости что-то ставить на компьютер и разбираться со сложной настройкой.
- Проверьте количество вакансий по каждому направлению в вашем городе если вы не планируете переезжать в Москву или Санкт-Петербург. Новичков стараются брать в офис, а не удаленно. Важно чтобы ваши желания совпадали с возможностями.
- Ну и в последнюю очередь вы можете последовать нашему совету: Frontend, Backend, Mobile, Gamedev. По языкам это JavaScript (TypeScript), PHP, Python, Java, C#. Остальные языки больше подходят для тех кто меняет направления, а не входит в разработку с нуля.

Как стать программистом

Программист с точки зрения компании

С точки зрения работодателя программист — это специалист, который умеет решать задачи бизнеса с помощью кода. Задачи могут быть сформулированы очень по-разному, где-то их хорошо прорабатывают и дают техническое задание, где-то задача построена на общих формулировках в духе “реализовать систему сбора обратной связи”. Отличный программист может работать в обоих ситуациях.



Основной критерий, по которому оценивают программиста, — это его способность понимать задачи, предлагать эффективные решения и воплощать их в жизнь. Причем результат работы легко измерить: исправлен ли баг, работает ли новая функциональность, отвечает ли система требованиям пользователей.

Высококлассными специалистами становятся не сразу. На это нужны годы качественного опыта, а пока они идут, программист проходит разные стадии, по которым разработчиков делят на три основных уровня (или грейда):

- **Junior** — начинающий разработчик. Его задача — выполнять простые поручения под руководством более опытных коллег. Компании нанимают таких специалистов, чтобы обучить их внутри команды и постепенно вводить в рабочий процесс. Джуниор разработчики, это обычно разработчики с опытом до 2-х лет.
- **Middle** — разработчик, способный работать самостоятельно. Он справляется с задачами средней сложности, может оценить время их выполнения и предложить

подход к решению. Считается что middle это разработчик, у которого за спиной 3-4 года опыта.

- **Senior** — высококвалифицированный специалист. Он не только решает сложные задачи, но и проектирует архитектуру систем, принимает стратегические решения и делится опытом с коллегами. Это разработчики с опытом больше 5 лет.

Для компании уровень программиста — это не только про технические навыки, но и про умение взаимодействовать с командой, понимать бизнес-контекст задач и предлагать решения, которые приносят реальную пользу. Чем быстрее разработчик может погрузиться в процессы и начать приносить результат, тем ценнее он для компании.

Джуниор: "Задача! Круто, сейчас сделаю!"

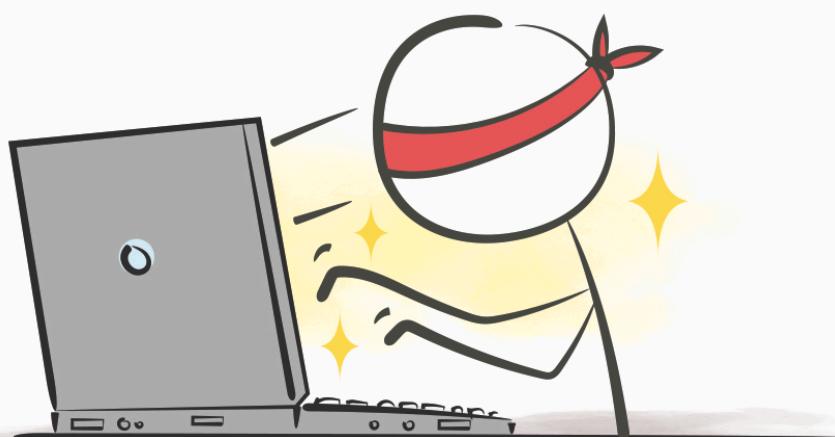
Миддл: "Задача... Хм, надо подумать, как сделать её правильно."

Сеньор: "Эта задача? Лучше обсудим, почему она вообще появилась."

На этом можно было бы закончить про уровни, если бы не пара важных "но".

Количество лет опыта – относительный параметр. Кто-то растет быстрее, кто-то медленнее, зависит и от самого человека, его бэкграунда, от проектов на которых он работает и от команды с которой он работает. Поэтому кто-то становится мидлом через год работы, а кто-то задерживается на этом уровне на очень долго. В общем случае, когда мы говорим про годы опыта, мы говорим не про время, а про реальные ситуации, через которые прошел человек за это время. Хороший мидл, полон историй как что-то ломалось, работало не как задумывалось и чинилось вспыхах. Кстати, именно поэтому, невозможно стать мидл-разработчиком через обучение, без соприкосновения с реальными проектами.

Н



Уровни зависят от компании. Разные компании решают задачи разной сложности. Программист, который пять лет делал корпоративный сайт строительной компании вряд ли без дополнительной подготовки попадет на работу, например, в Google. И даже если пройдет, то на более низкий грейд. Поэтому это разделение достаточно условное.

Главный критерий успеха

Что значит “стать программистом”? Программистом, чисто технически, является любой человек, который может создавать полезные в применении приложения, которые он использует сам или которыми пользуются другие люди. Вроде все правильно, но это ответ, который вас вряд ли устроит. Мы собрались здесь, потому что хотим не только писать код, но и получать за это деньги, что приводит к нас к следующему определению.



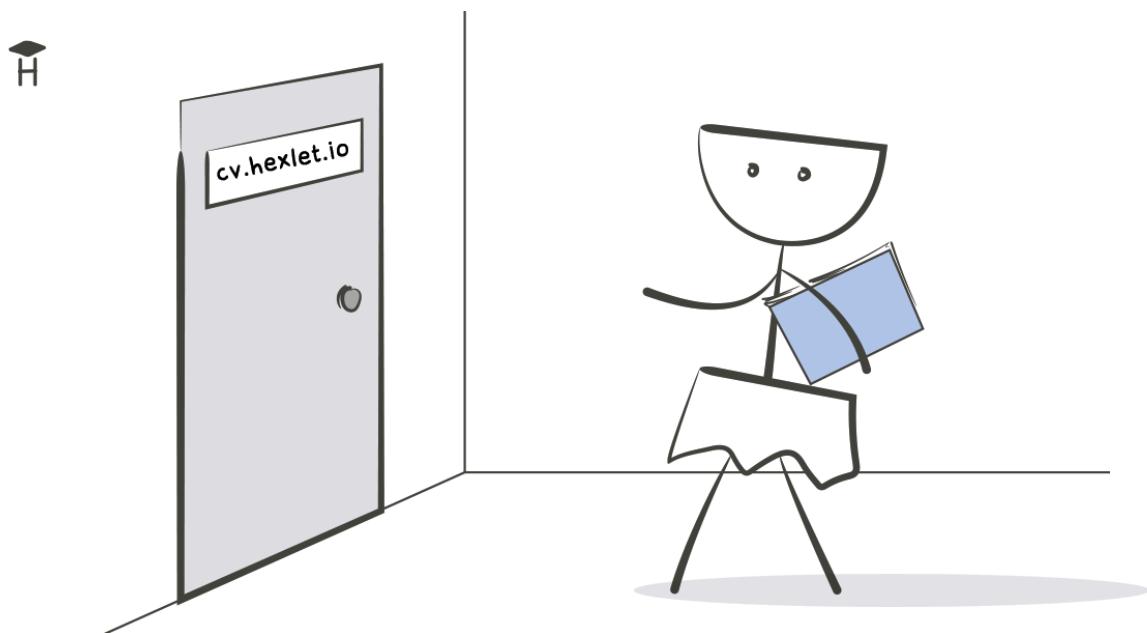
Стать программистом — значит пройти собеседование и доказать работодателю на испытательном сроке, что вы способны решать задачи, которые перед вами поставят. Такая формулировка отрезвляет и задает правильное целеполагание:

1. Учусь программированию
2. Готовлю резюме
3. Ищу работу
4. Прохожу собеседование <=
5. Я программист

Мы подробно разберем каждый из этих пунктов. А пока ответим на важный вопрос. Как понять что я готов к поиску работы?

Готов ли я к трудуоустройству?

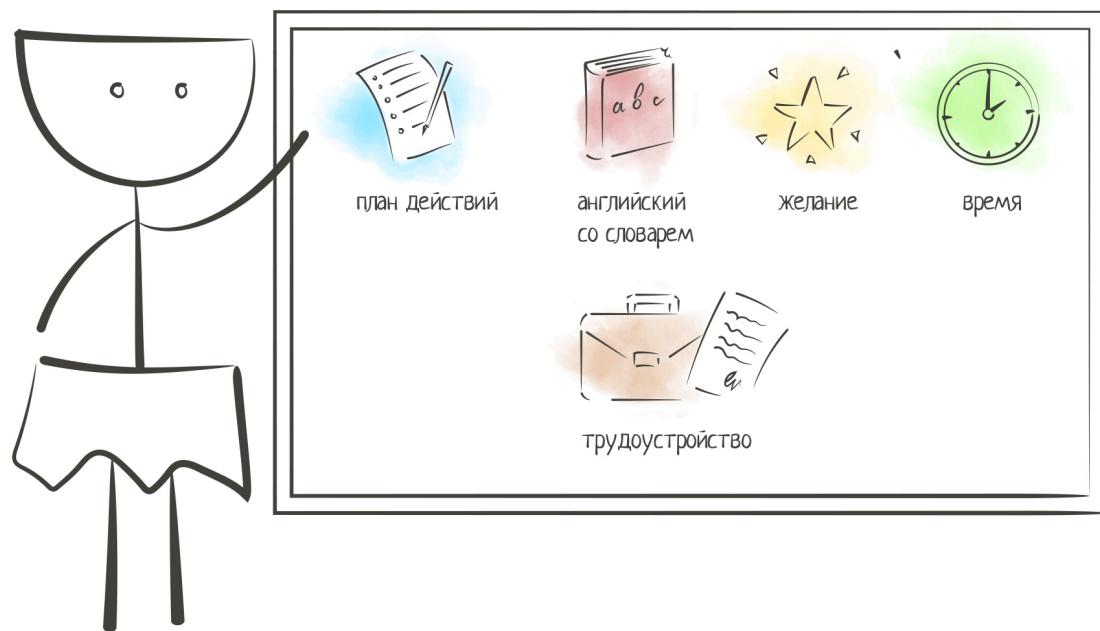
Этот вопрос важно задавать себе заранее, так как легко не заметить, как обучение затягивается. Разработчики, мало того, что должны знать много, так им еще приходится постоянно учиться, чтобы оставаться в форме. Из-за этого процесс входа в программирование может растянуться до бесконечности. Вас будет преследовать постоянное ощущение, что нужно еще чуть-чуть. Происходит попадание в классическую ловушку “чем больше я знаю – тем больше я не знаю”. Каждая новая тема открывает новый мир, со своими особенностями. Внезапно оказывается, что на обучение потрачены годы, а результат непонятен. Готов ли я или нет?



Критерии готовности всегда субъективны, но есть один практический подход: способность выполнить тестовые задания, которые дают кандидатам различные компании. Если вы справляетесь с такими задачами и понимаете, как их решать, это хороший сигнал о вашей готовности. В этот момент точно имеет смысл начинать искать работу. Проверить себя можно взяв любое подходящие задание из [нашой постоянно пополняемой коллекции](#).

Подводя итог, можно сказать, что в первую очередь имеет значение ваша способность собрать работающий проект с нуля, а не иметь множество отрывочных знаний, которые вместе не складываются в общую систему. Последнее часто встречается в классическом образовании, где дают много независимых дисциплин, но без проектной деятельности, все это превращается в набор бессвязных знаний и неспособность сделать что-то работающее на выходе.

С чего начать?



Путь в программирование начинается с выбора направления, составления четкого плана и создания условий для регулярного обучения. Давайте разберем основные шаги, которые помогут вам не только начать, но и успешно двигаться к своей цели.

Выбираем направление



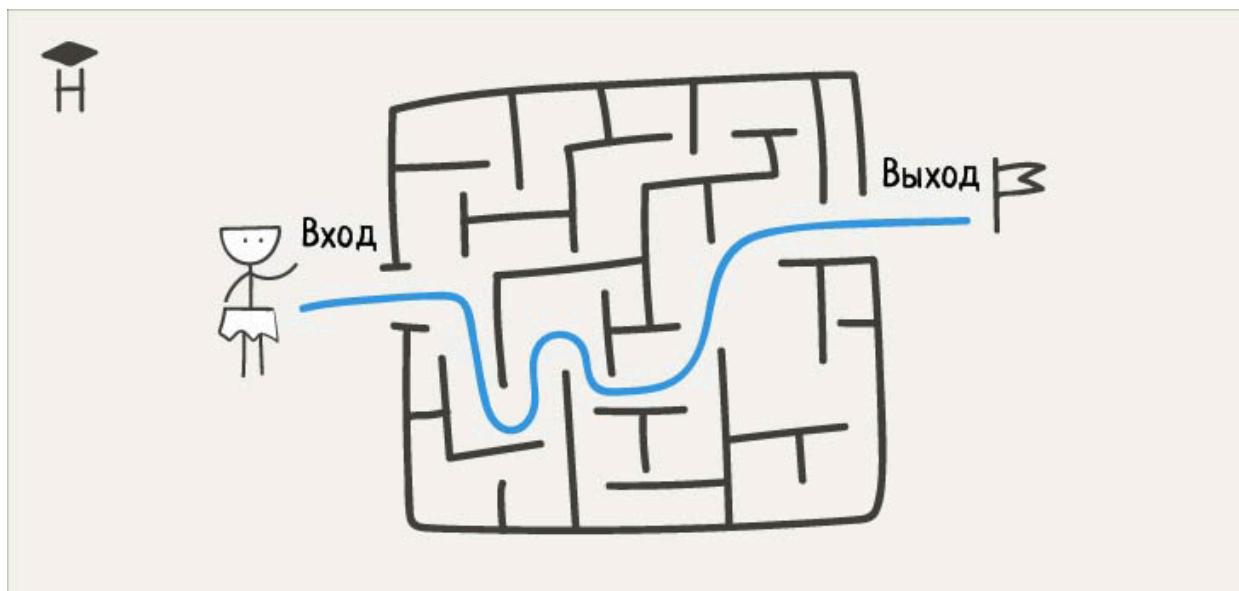
Первый шаг — определиться, в каком направлении вы хотите развиваться. Звучит просто, но на практике не так очевидно. Если я люблю игры, значит мне нужно идти в gameDev? А на тех же мобильниках бывают абсолютно любые приложения от игр до таблиц. И как можно любить сайты это ведь просто инструмент? Все это правда. Если у вас нет четкой уверенности что делать, то ориентируйтесь на эти критерии:

Желание: выберите то, что вызывает у вас интерес и энтузиазм. Это поможет сохранить мотивацию на протяжении всего пути. Этот же совет можно перевернуть. Исключите из списка то, что вам точно не нравится, а остальные направления рассматривайте через следующие критерии

Рынок труда: если вы не планируете переезд, проверьте количество вакансий в вашем городе для выбранного направления. Например, в некоторых регионах может быть больше спроса на веб-разработчиков, чем на специалистов по мобильной разработке. Тоже игровые студии есть далеко не в каждом городе, а там где они есть, спрос на таких разработчиков высокий.

Простота входа: некоторые направления, такие как веб-разработка, имеют более низкий порог для новичков, чем, например, разработка встроенных систем или Data Science. Но даже внутри веб-разработки, есть разные языки и направления, которые могут как сложнее для входа так и проще.

Формируем план обучения



Составить план обучения с нуля — задача непростая, особенно для новичка. Часто новичок не знает, что именно ему нужно изучить, и какова последовательность действий. Поэтому разумным решением будет опереться на готовые программы курсов. Многие образовательные платформы предлагают структурированные планы, которые включают

все ключевые темы и помогают не упустить ничего важного. Отличным дополнением к этому будут книги.

Почему не программы вузов? Эти программы слишком сложны и разделены на разные предметы, которые не имеет смысла так глубоко учить независимо. Прикладное обучение это единый поток, в котором темы добавляются тогда когда нужно и только для того, чтобы решить очередную задачу. Ваша конечная цель – быть способными написать законченное приложение, а не стать специалистами в каждой из областей.

Далее в книге мы расскажем о базовых концепциях и темах, которые стоит освоить, но готовый учебный план станет вашим надёжным помощником в начале пути. Это как карта, которая помогает не сбиться с маршрута.

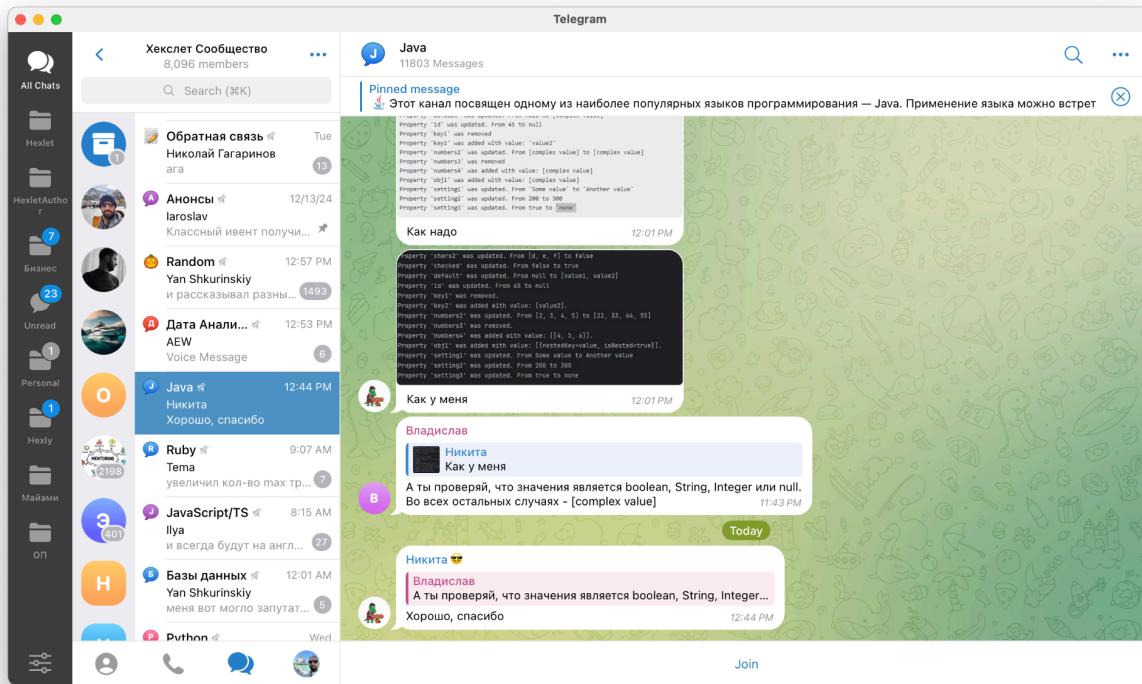
Формируем расписание



Регулярность — ключ к успеху в обучении программированию. Постарайтесь выделять время на занятия каждый день, даже если это будет всего 1-2 часа. Важно выработать привычку учиться регулярно. Идеально если вы пользуетесь каким-нибудь календарем, например в вашем яндекс или гугл аккаунте. Зарезервируйте через них слоты, чтобы телефон вам напоминал о занятиях.

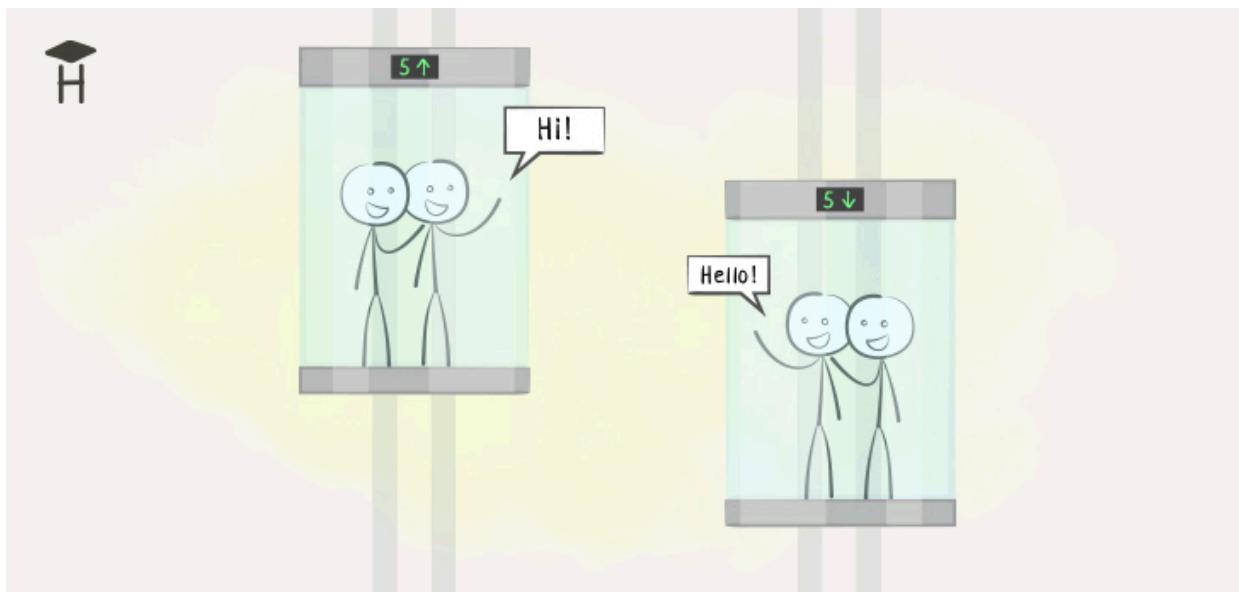
Помимо этого, выделите один выходной день, когда сможете заниматься более продолжительное время, например, 5 часов подряд. Такой "глубокий фокус" помогает погружаться в сложные темы и добирать навыки через практику. В идеале старайтесь закладывать около 15-20 часов в неделю на обучение — это оптимальный баланс между прогрессом и возможностью совмещать учёбу с другими делами.

Находим единомышленников



Учиться в одиночку бывает сложно, поэтому найдите тех, кто идёт по тому же пути. Это могут быть сообщества в интернете, форумы, [Telegram-чаты](#) или [локальные группы](#) начинающих программистов. Общение с единомышленниками не только поддерживает мотивацию, но и помогает находить ответы на сложные вопросы, делиться опытом и участвовать в совместных проектах.

Английский язык



Независимо от того, какое направление вы выберете, первый язык, который нужно начать изучать, — это английский. Вам не обязательно уметь свободно говорить на нём, но минимальный уровень владения — умение читать со словарём — необходим. Почему?

Программирование — это глобальная сфера, и английский язык является её стандартом. Большая часть документации, обучающих материалов, форумов и технических статей написаны на английском. Даже если существуют переводы, они часто отстают от оригиналов по актуальности и полноте.

Кроме того, большинство языков программирования, инструментов и библиотек используют английский синтаксис. Такие базовые слова, как `if`, `else`, `function`, `return`, встречаются в любом коде, и без понимания их значения будет сложно даже начать.

Помимо этого, английский играет ключевую роль в процессе отладки ошибок. Любая программа может содержать ошибки, и сообщения об этих ошибках, которые выводят инструменты разработки, почти всегда на английском языке. Умение понимать такие сообщения позволяет быстрее находить и устранять проблемы. Например, сообщение `NullPointerException` или `SyntaxError` говорит о конкретной причине сбоя. Без понимания смысла этих сообщений отладка превращается в сложный и долгий процесс.

Начните с малого: читайте документацию, смотрите обучающие видео и изучайте термины. Переведите все интерфейсы на вашем компьютере и телефоне на английский язык, это сильно поможет продвинуться вперед без специальных усилий.

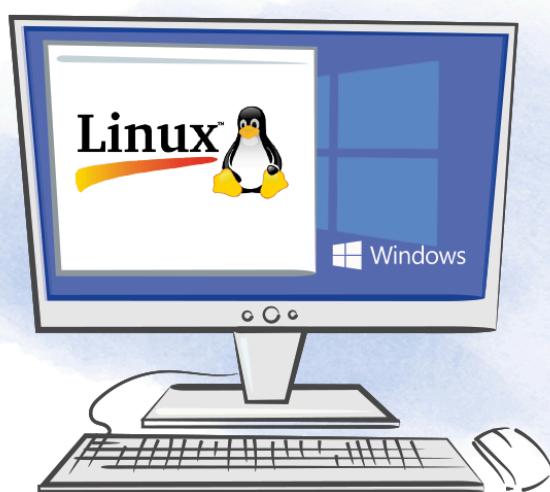
Подготовка к практике

После того как вы выбрали направление, составили план обучения и организовали своё расписание, самое время перейти к практике. Теория важна, но программирование — это навык, который развивается только через регулярное выполнение задач и написание кода. Для этого необходимо правильно настроить рабочее окружение и освоить базовые инструменты, которые будут вашими основными помощниками.

Какой нужен для этого компьютер? Работа с кодом, достаточно простая нагрузка, которая не идет ни в какое сравнение с, например, запуском игр или обработкой видео. Поэтому для обучения разработке почти наверняка подойдет тот компьютер, который у вас уже есть. Конкретные требования, скорее, зависят от ваших задач. Если вы хотите стать разработчиком софта под Iphone, то у вас должен быть Macbook.

В программировании код — это всего лишь текст. Но чтобы этот текст заработал, вам понадобятся специальные программы, которые смогут его "прочитать" и выполнить. Эти программы называются инструментами разработки, и для их установки нужно настроить ваше рабочее окружение.

Н



Настройку можно выполнить на базе разных материалов:

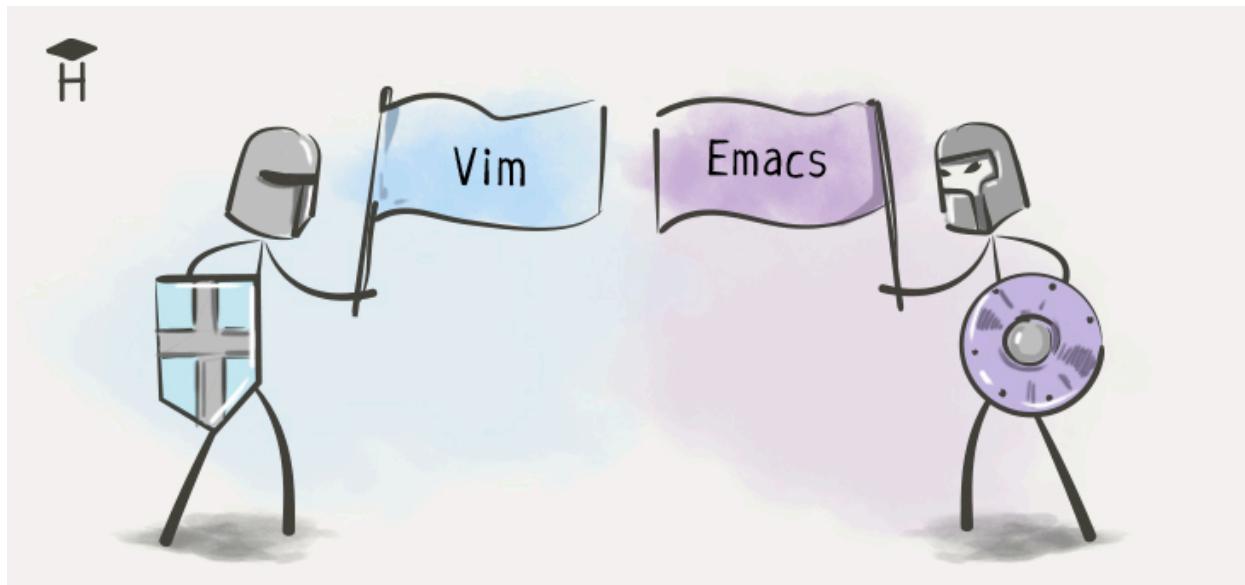
- Найти в интернете статьи на тему того "как установить язык X на операционную систему Y". Лучше на английском языке. Это относительно сложный путь, потому что в этих статьях редко объясняют детали, они больше подходят тем кто уже немного шарит в программировании
- По описанию в книгах, посвященным обучению языку. Может сработать, но они быстро устаревают.

- На базе курсов, если вы выбрали такой путь. В этом случае, скорее всего, вам будет к кому обратиться, чтобы разобраться с возникающими проблемами в процессе настройки.
- Современные редакторы иногда самостоятельно правильно определяют с каким языком вы работаете и устанавливают все автоматически. Тут как повезет.

Настройка окружения хоть и не связана с программированием напрямую, является неотъемлемой частью разработки. Установка нужных программ, настройка операционной системы, запуск баз данных, сетевых сервисов и тому подобное, сопровождает программиста непрерывно на протяжении всей карьеры. Процесс это довольно болезненный, особенно для новичков. Слишком много всего, слишком непонятные ошибки, слишком похоже на магию. И потраченные дни и недели, на то, чтобы оно заработало.

Многих это пугает, они считают что зря потратили время пытаясь завести новый проект у себя на компьютере потратив на это три дня. В реальности же, настройка окружения, это один из факторов роста и обязательный элемент процесса обучения. Главное, чтобы он выполнялся не по принципу “загуглил команду и вставил”, а через попытку понять что за действия вы делаете и зачем. В любом случае готовьтесь к тому, что по инструкции не заработает. Это нормально, слишком у всех разные ситуации и комбинации программ и железа.

Редактор



Редактор кода — это программа, в которой вы будете писать и редактировать свой код. Для начинающих отлично подойдет **Visual Studio Code (VS Code)**. Это бесплатный, мощный и интуитивно понятный редактор, который широко используется в индустрии.

```
JS app.js x
JS app.js > [e] default > fp() callback > [e] api
1 import path from 'path'
2 import AutoLoad from '@fastify/autoload'
3 import fp from 'fastify-plugin'
4 import { TypeBoxValidatorCompiler } from '@fastify/type-provider-typebox'
5 import { errors } from '@vinejs/vine'
6
7 // Pass --options via CLI arguments in command to enable these options.
8 export const options = {}
9
10 /**
11  * @param {import('fastify').FastifyInstance} fastify
12 */
13 export default fp(async function (fastify, opts) {
14 // Place here your custom code!
15
16 const api = fastify
17 .setValidatorCompiler(TypeBoxValidatorCompiler)
18 .withTypeProvider()
19
20 fastify.setErrorHandler(function (error, _request, reply) {
21 if (error instanceof errors.E_VALIDATION_ERROR) {
22 const errorDetail = {
23 status: 422,
24 title: 'Validation Error',
25 detail: 'Errors related to business logic such as uniqueness',
26 errors: error.messages,
27 }
28 reply.code(422).send(errorDetail)
29 } else {
30 reply.send(error)
31 }
32 }
33 )
34
35 // Do not touch the following lines
36
37 // This loads all plugins defined in plugins
38 // those should be support plugins that are reused
```

Ln 18, Col 20 Spaces: 2 UTF-8 LF () JavaScript

Установить VS Code можно через [сайт](#). После установки настройте редактор под свои нужды: выберите удобную тему и установите поддержку вашего языка программирования. Либо дождитесь момента, когда во время обучения вам понадобится написать код. Редактор сам подскажет, что установить основываясь на том, какое расширение у созданных файлов.

В разработке принято разделять два понятия: редакторы и IDE (интегрированная среда разработки, говорят “идэе”). Редактор, по задумке, обеспечивает только базовые функции, например подсветку кода и навигацию по файловому дереву, в то время как IDE позволяет запускать код. В реальности эти понятия почти слились, редакторов в чистом виде практически не осталось. Например VSCode хоть и называется редактором, фактически является полноценной IDE.

Отладка (Дебагинг)



По большей части, изучение программирование ничем принципиально не отличается от всего остального. Советы в этой книге вполне универсальны и почти без изменений могут быть применены для изучения data-аналитики, дизайна или маркетинга. Но есть одна вещь, которая отличает программирования от всего остального настолько сильно, что все обучение и успех строится вокруг этого аспекта. Я говорю об ошибках и работе с ними.

Программирование очень строгая штука с точки зрения кодинга. В отличие от обычных языков, код подчиняется правилам, которые нельзя нарушать. С одной стороны правила упрощают процесс изучения, с другой, на первых порах постоянно возникают ошибки, когда синтаксис программы не соответствует требованиям. Опытные разработчики почти не допускают таких ошибок, но новички могут забыть поставить точку и потом два дня искать что же не так. Еще более сложные ошибки это ошибки логики, когда программа запускается и, даже, работает, но выдает неверный результат. Поиск и устранение таких ошибок занимает у программистов львиную часть времени даже когда они становятся опытными.

Из-за этого, например, очень сложно просчитать время, необходимое на обучение. Человек который не умеет быстро находить и исправлять такие ошибки, будет учиться во много раз дольше, тратя впустую время на поиск больше времени, чем нужно. Все это усугубляется, если не у кого спросить помощи. Сейчас, конечно, проще благодаря ChatGPT, но даже он не всесилен.

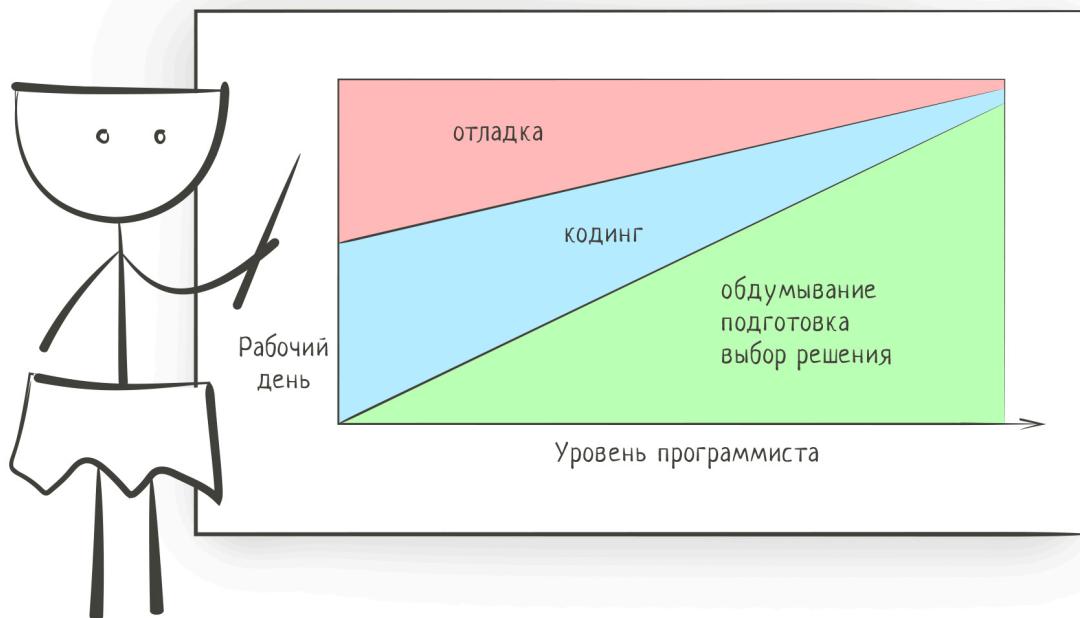
```

nvim
2 . Application.java x GlobalExceptionHa... x EncodersConfig... x JacksonConfig.java x SecurityConfig.. x
  Local:
    auth AuthenticationConfiguration$D
    this SecurityConfig$$SpringCGLIB$#
    SecurityConfig.java:
      62 var provider = new DaoAuthenticat
      63
      64 @Bean
      65     public AuthenticationManager authenticationManag
      66         return http.getSharedObject(sharedType.Authe
      67             .build();
      68
      69 @Bean
      70     public AuthenticationProvider daoAuthProvider(Au
      71         var provider = new DaoAuthenticationProvider()
      72             provider.setUserDetailsService(userDetailsSe
      73             provider.setPasswordEncoder(passwordEncoder)
      74             return provider;
      75
      76
      77
      78
      79
      80
      81
      82
      83
      84
      85
      86
      87
      88
      89
      90
      91
      92
      93
      94
      95
      96
      97
      98
      99
      100
      101
      102
      103
      104
      105
      106
      107
      108
      109
      110
      111
      112
      113
      114
      115
      116
      117
      118
      119
      120
      121
      122
      123
      124
      125
      126
      127
      128
      129
      130
      131
      132
      133
      134
      135
      136
      137
      138
      139
      140
      141
      142
      143
      144
      145
      146
      147
      148
      149
      150
      151
      152
      153
      154
      155
      156
      157
      158
      159
      160
      161
      162
      163
      164
      165
      166
      167
      168
      169
      170
      171
      172
      173
      174
      175
      176
      177
      178
      179
      180
      181
      182
      183
      184
      185
      186
      187
      188
      189
      190
      191
      192
      193
      194
      195
      196
      197
      198
      199
      200
      201
      202
      203
      204
      205
      206
      207
      208
      209
      210
      211
      212
      213
      214
      215
      216
      217
      218
      219
      220
      221
      222
      223
      224
      225
      226
      227
      228
      229
      230
      231
      232
      233
      234
      235
      236
      237
      238
      239
      240
      241
      242
      243
      244
      245
      246
      247
      248
      249
      250
      251
      252
      253
      254
      255
      256
      257
      258
      259
      260
      261
      262
      263
      264
      265
      266
      267
      268
      269
      270
      271
      272
      273
      274
      275
      276
      277
      278
      279
      280
      281
      282
      283
      284
      285
      286
      287
      288
      289
      290
      291
      292
      293
      294
      295
      296
      297
      298
      299
      300
      301
      302
      303
      304
      305
      306
      307
      308
      309
      310
      311
      312
      313
      314
      315
      316
      317
      318
      319
      320
      321
      322
      323
      324
      325
      326
      327
      328
      329
      330
      331
      332
      333
      334
      335
      336
      337
      338
      339
      340
      341
      342
      343
      344
      345
      346
      347
      348
      349
      350
      351
      352
      353
      354
      355
      356
      357
      358
      359
      360
      361
      362
      363
      364
      365
      366
      367
      368
      369
      370
      371
      372
      373
      374
      375
      376
      377
      378
      379
      380
      381
      382
      383
      384
      385
      386
      387
      388
      389
      390
      391
      392
      393
      394
      395
      396
      397
      398
      399
      400
      401
      402
      403
      404
      405
      406
      407
      408
      409
      410
      411
      412
      413
      414
      415
      416
      417
      418
      419
      420
      421
      422
      423
      424
      425
      426
      427
      428
      429
      430
      431
      432
      433
      434
      435
      436
      437
      438
      439
      440
      441
      442
      443
      444
      445
      446
      447
      448
      449
      450
      451
      452
      453
      454
      455
      456
      457
      458
      459
      460
      461
      462
      463
      464
      465
      466
      467
      468
      469
      470
      471
      472
      473
      474
      475
      476
      477
      478
      479
      480
      481
      482
      483
      484
      485
      486
      487
      488
      489
      490
      491
      492
      493
      494
      495
      496
      497
      498
      499
      500
      501
      502
      503
      504
      505
      506
      507
      508
      509
      510
      511
      512
      513
      514
      515
      516
      517
      518
      519
      520
      521
      522
      523
      524
      525
      526
      527
      528
      529
      530
      531
      532
      533
      534
      535
      536
      537
      538
      539
      540
      541
      542
      543
      544
      545
      546
      547
      548
      549
      550
      551
      552
      553
      554
      555
      556
      557
      558
      559
      560
      561
      562
      563
      564
      565
      566
      567
      568
      569
      570
      571
      572
      573
      574
      575
      576
      577
      578
      579
      580
      581
      582
      583
      584
      585
      586
      587
      588
      589
      590
      591
      592
      593
      594
      595
      596
      597
      598
      599
      600
      601
      602
      603
      604
      605
      606
      607
      608
      609
      610
      611
      612
      613
      614
      615
      616
      617
      618
      619
      620
      621
      622
      623
      624
      625
      626
      627
      628
      629
      630
      631
      632
      633
      634
      635
      636
      637
      638
      639
      640
      641
      642
      643
      644
      645
      646
      647
      648
      649
      650
      651
      652
      653
      654
      655
      656
      657
      658
      659
      660
      661
      662
      663
      664
      665
      666
      667
      668
      669
      670
      671
      672
      673
      674
      675
      676
      677
      678
      679
      680
      681
      682
      683
      684
      685
      686
      687
      688
      689
      690
      691
      692
      693
      694
      695
      696
      697
      698
      699
      700
      701
      702
      703
      704
      705
      706
      707
      708
      709
      710
      711
      712
      713
      714
      715
      716
      717
      718
      719
      720
      721
      722
      723
      724
      725
      726
      727
      728
      729
      730
      731
      732
      733
      734
      735
      736
      737
      738
      739
      740
      741
      742
      743
      744
      745
      746
      747
      748
      749
      750
      751
      752
      753
      754
      755
      756
      757
      758
      759
      760
      761
      762
      763
      764
      765
      766
      767
      768
      769
      770
      771
      772
      773
      774
      775
      776
      777
      778
      779
      780
      781
      782
      783
      784
      785
      786
      787
      788
      789
      790
      791
      792
      793
      794
      795
      796
      797
      798
      799
      800
      801
      802
      803
      804
      805
      806
      807
      808
      809
      810
      811
      812
      813
      814
      815
      816
      817
      818
      819
      820
      821
      822
      823
      824
      825
      826
      827
      828
      829
      830
      831
      832
      833
      834
      835
      836
      837
      838
      839
      840
      841
      842
      843
      844
      845
      846
      847
      848
      849
      850
      851
      852
      853
      854
      855
      856
      857
      858
      859
      860
      861
      862
      863
      864
      865
      866
      867
      868
      869
      870
      871
      872
      873
      874
      875
      876
      877
      878
      879
      880
      881
      882
      883
      884
      885
      886
      887
      888
      889
      890
      891
      892
      893
      894
      895
      896
      897
      898
      899
      900
      901
      902
      903
      904
      905
      906
      907
      908
      909
      910
      911
      912
      913
      914
      915
      916
      917
      918
      919
      920
      921
      922
      923
      924
      925
      926
      927
      928
      929
      930
      931
      932
      933
      934
      935
      936
      937
      938
      939
      940
      941
      942
      943
      944
      945
      946
      947
      948
      949
      950
      951
      952
      953
      954
      955
      956
      957
      958
      959
      960
      961
      962
      963
      964
      965
      966
      967
      968
      969
      970
      971
      972
      973
      974
      975
      976
      977
      978
      979
      980
      981
      982
      983
      984
      985
      986
      987
      988
      989
      990
      991
      992
      993
      994
      995
      996
      997
      998
      999
      1000
      1001
      1002
      1003
      1004
      1005
      1006
      1007
      1008
      1009
      1010
      1011
      1012
      1013
      1014
      1015
      1016
      1017
      1018
      1019
      1020
      1021
      1022
      1023
      1024
      1025
      1026
      1027
      1028
      1029
      1030
      1031
      1032
      1033
      1034
      1035
      1036
      1037
      1038
      1039
      1040
      1041
      1042
      1043
      1044
      1045
      1046
      1047
      1048
      1049
      1050
      1051
      1052
      1053
      1054
      1055
      1056
      1057
      1058
      1059
      1060
      1061
      1062
      1063
      1064
      1065
      1066
      1067
      1068
      1069
      1070
      1071
      1072
      1073
      1074
      1075
      1076
      1077
      1078
      1079
      1080
      1081
      1082
      1083
      1084
      1085
      1086
      1087
      1088
      1089
      1090
      1091
      1092
      1093
      1094
      1095
      1096
      1097
      1098
      1099
      1100
      1101
      1102
      1103
      1104
      1105
      1106
      1107
      1108
      1109
      1110
      1111
      1112
      1113
      1114
      1115
      1116
      1117
      1118
      1119
      1120
      1121
      1122
      1123
      1124
      1125
      1126
      1127
      1128
      1129
      1130
      1131
      1132
      1133
      1134
      1135
      1136
      1137
      1138
      1139
      1140
      1141
      1142
      1143
      1144
      1145
      1146
      1147
      1148
      1149
      1150
      1151
      1152
      1153
      1154
      1155
      1156
      1157
      1158
      1159
      1160
      1161
      1162
      1163
      1164
      1165
      1166
      1167
      1168
      1169
      1170
      1171
      1172
      1173
      1174
      1175
      1176
      1177
      1178
      1179
      1180
      1181
      1182
      1183
      1184
      1185
      1186
      1187
      1188
      1189
      1190
      1191
      1192
      1193
      1194
      1195
      1196
      1197
      1198
      1199
      1200
      1201
      1202
      1203
      1204
      1205
      1206
      1207
      1208
      1209
      1210
      1211
      1212
      1213
      1214
      1215
      1216
      1217
      1218
      1219
      1220
      1221
      1222
      1223
      1224
      1225
      1226
      1227
      1228
      1229
      1230
      1231
      1232
      1233
      1234
      1235
      1236
      1237
      1238
      1239
      12310
      12311
      12312
      12313
      12314
      12315
      12316
      12317
      12318
      12319
      12320
      12321
      12322
      12323
      12324
      12325
      12326
      12327
      12328
      12329
      12330
      12331
      12332
      12333
      12334
      12335
      12336
      12337
      12338
      12339
      12340
      12341
      12342
      12343
      12344
      12345
      12346
      12347
      12348
      12349
      12350
      12351
      12352
      12353
      12354
      12355
      12356
      12357
      12358
      12359
      12360
      12361
      12362
      12363
      12364
      12365
      12366
      12367
      12368
      12369
      12370
      12371
      12372
      12373
      12374
      12375
      12376
      12377
      12378
      12379
      12380
      12381
      12382
      12383
      12384
      12385
      12386
      12387
      12388
      12389
      12390
      12391
      12392
      12393
      12394
      12395
      12396
      12397
      12398
      12399
      123100
      123101
      123102
      123103
      123104
      123105
      123106
      123107
      123108
      123109
      123110
      123111
      123112
      123113
      123114
      123115
      123116
      123117
      123118
      123119
      123120
      123121
      123122
      123123
      123124
      123125
      123126
      123127
      123128
      123129
      123130
      123131
      123132
      123133
      123134
      123135
      123136
      123137
      123138
      123139
      123140
      123141
      123142
      123143
      123144
      123145
      123146
      123147
      123148
      123149
      123150
      123151
      123152
      123153
      123154
      123155
      123156
      123157
      123158
      123159
      123160
      123161
      123162
      123163
      123164
      123165
      123166
      123167
      123168
      123169
      123170
      123171
      123172
      123173
      123174
      123175
      123176
      123177
      123178
      123179
      123180
      123181
      123182
      123183
      123184
      123185
      123186
      123187
      123188
      123189
      123190
      123191
      123192
      123193
      123194
      123195
      123196
      123197
      123198
      123199
      123200
      123201
      123202
      123203
      123204
      123205
      123206
      123207
      123208
      123209
      123210
      123211
      123212
      123213
      123214
      123215
      123216
      123217
      123218
      123219
      123220
      123221
      123222
      123223
      123224
      123225
      123226
      123227
      123228
      123229
      123230
      123231
      123232
      123233
      123234
      123235
      123236
      123237
      123238
      123239
      123240
      123241
      123242
      123243
      123244
      123245
      123246
      123247
      123248
      123249
      123250
      123251
      123252
      123253
      123254
      123255
      123256
      123257
      123258
      123259
      123260
      123261
      123262
      123263
      123264
      123265
      123266
      123267
      123268
      123269
      123270
      123271
      123272
      123273
      123274
      123275
      123276
      123277
      123278
      123279
      123280
      123281
      123282
      123283
      123284
      123285
      123286
      123287
      123288
      123289
      123290
      123291
      123292
      123293
      123294
      123295
      123296
      123297
      123298
      123299
      123300
      123301
      123302
      123303
      123304
      123305
      123306
      123307
      123308
      123309
      123310
      123311
      123312
      123313
      123314
      123315
      123316
      123317
      123318
      123319
      123320
      123321
      123322
      123323
      123324
      123325
      123326
      123327
      123328
      123329
      123330
      123331
      123332
      123333
      123334
      123335
      123336
      123337
      123338
      123339
      123340
      123341
      123342
      123343
      123344
      123345
      123346
      123347
      123348
      123349
      123350
      123351
      123352
      123353
      123354
      123355
      123356
      123357
      123358
      123359
      123360
      123361
      123362
      123363
      123364
      123365
      123366
      123367
      123368
      123369
      123370
      123371
      123372
      123373
      123374
      123375
      123376
      123377
      123378
      123379
      123380
      123381
      123382
      123383
      123384
      123385
      123386
      123387
      123388
      123389
      123390
      123391
      123392
      123393
      123394
      123395
      123396
      123397
      123398
      123399
      123400
      123401
      123402
      123403
      123404
      123405
      123406
      123407
      123408
      123409
      123410
      123411
      123412
      123413
      123414
      123415
      123416
      123417
      123418
      123419
      123420
      123421
      123422
      123423
      123424
      123425
      123426
      123427
      123428
      123429
      123430
      123431
      123432
      123433
      123434
      123435
      123436
      123437
      123438
      123439
      123440
      123441
      123442
      123443
      123444
      123445
      123446
      123447
      123448
      123449
      123450
      123451

```

Изучение программирования

Программирование — это не только набор языков или технологий. Это способ мышления, который помогает анализировать задачи и находить решения. Изучение программирования состоит из нескольких ключевых этапов: от освоения основ до глубокого погружения в прикладные инструменты.



В интернете немало гайдов по изучению программирования, например <https://roadmap.sh/>, но откровенно говоря, на их основе сложно понять как и что нужно учить на самом деле, потому что они пытаются охватить вообще все аспекты каждого направления размещая вперемешку как общие знания так и прикладные инструменты, но для ознакомления они вполне подходят.

Независимо от того, какой аспект программирования вы изучаете, помните, что конечная цель – уметь создавать работающие приложения. Поэтому старайтесь не распыляться и не изучать много всего параллельно.

Основы программирования

Основа любого языка программирования — это базовые конструкции, которые лежат в основе написания любого кода и умение из этих конструкций составлять минимально работающие программы, выполняющие в основном примитивные задачи в духе калькулятора. На этом этапе вы изучаете:

- **Переменные:** как хранить данные и работать с ними.
- **Условные операторы:** как принимать решения в коде.

- **Циклы:** как выполнять действия многократно.
- **Функции:** как организовывать код для многократного использования.
- **Модули:** как создавать многофайловые программы



Эти элементы составляют фундамент, на котором строятся все программы. Основы программирования помогают понять, как компьютеры "думают" и выполняют команды. Важно не просто заучивать синтаксис, а пытаться понять, как эти конструкции решают реальные задачи. Например, написать алгоритм для подсчета чисел или фильтрации данных — это первые шаги к решению более сложных задач.

```

const reverse = (str) => {
  let arr = [];
  for (let i = str.length - 1, j = 0; i >= 0;
    i--, j++)
    arr[j] = str[i];
  return arr.join('');
}

reverse('hello, world!')

```

НАВСЕРОТ?

ДЛINA? РАЗМЕР?

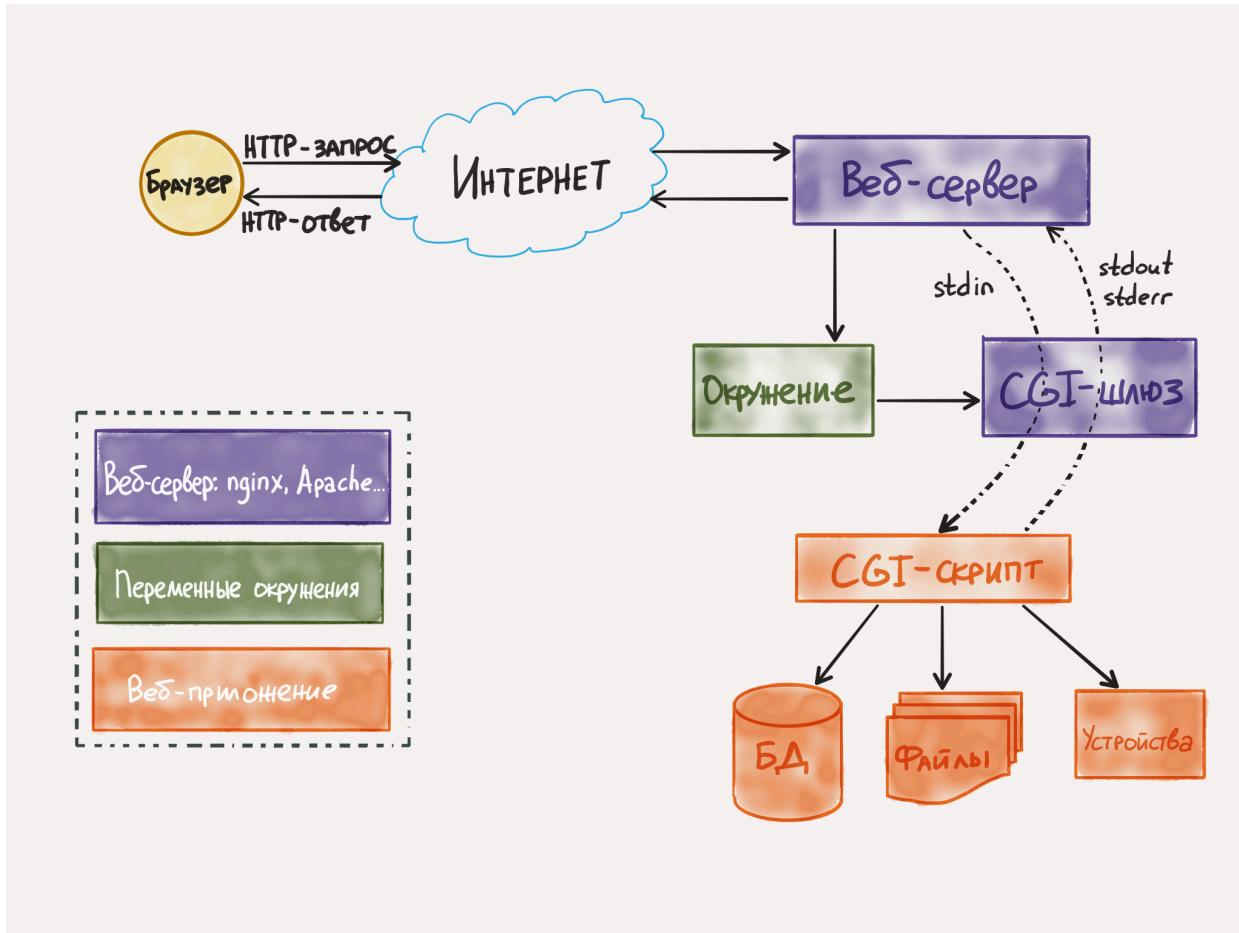
Больше или равно?

Соединить?

Самое приятное в этой части, что она почти идентична для большинства языков. Отличия в основном в используемых знаках, а визуально код будет выглядеть одинаково. Поэтому изучения базы можно делать почти на любом языке. Вы в любой момент сможете переключиться на другой без особых проблем.

Бесплатно и без смс изучить их можно на проекте <https://code-basics.com/ru>

Продвинутые техники



После освоения базовых концепций вы можете переходить к более сложным инструментам:

- **Объектно-ориентированное программирование (ООП):** создание объектов, управление их состояниями и взаимодействием. Например, создание классов для работы с пользователями в приложении.
- **Асинхронное программирование:** выполнение задач, не блокируя программу, например, работа с сетевыми запросами или загрузка данных в фоне.
- **Работа с API:** взаимодействие с внешними системами, такими как базы данных, веб-сервисы или сторонние приложения, через запросы.

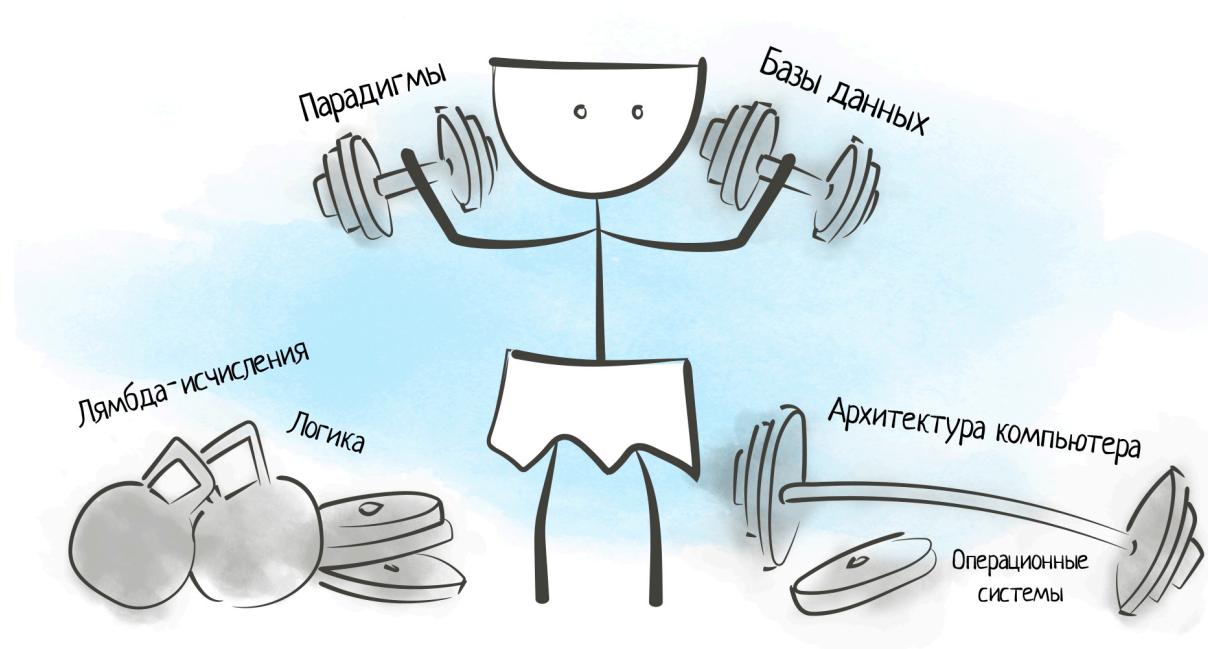
На этом этапе изучаются концепции, которые используются в промышленном программировании для создания по настоящему полезных программ. Здесь начинают проявляться различия между языками. Даже если концепции похожие, то реализация в разных языках может быть очень разная.

Количество тем на этом уровне огромно. Даже опытные разработчики знают не все из них, а те что знают, знают не всегда глубоко. Поэтому тут важно иметь план что учить и в каком объеме, иначе можно закопаться.

Окружение

Работа программиста — это не только написание кода, но и взаимодействие с окружающей средой, такой как операционные системы, файлы, сеть и базы данных.

- **Работа с файлами:** чтение, запись, удаление и обработка данных, например, загрузка файлов или создание логов.
- **Сеть:** понимание, как работают протоколы (HTTP, HTTPS) и как взаимодействовать с серверами через запросы.
- **Базы данных:** хранение и извлечение информации с использованием SQL или других инструментов.

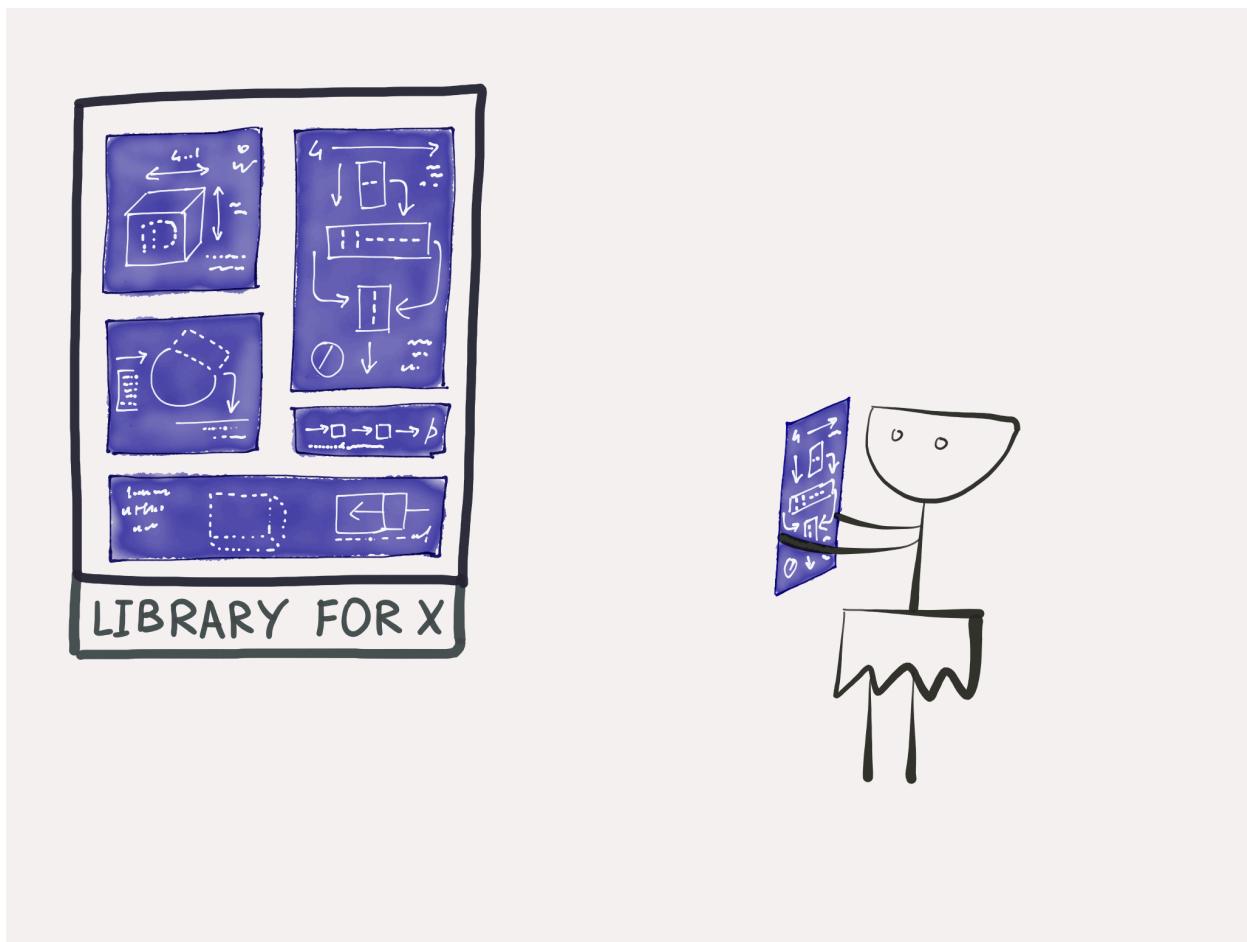


Освоение взаимодействия с окружением поможет вам строить полноценные приложения, которые не только работают локально, но и взаимодействуют с внешним миром.

Прикладные инструменты

Фреймворки и библиотеки — это мощные инструменты, которые ускоряют разработку, упрощая выполнение стандартных задач.

- **Фреймворки:** например, React для фронтенд-разработки или Django для работы с серверной частью. Они предоставляют готовые структуры и шаблоны, на которых можно быстро строить приложения.
- **Библиотеки:** это наборы готовых функций, которые упрощают выполнение задач, например, обработку данных (Pandas) или построение графиков (Matplotlib).



Использование прикладных инструментов позволяет сосредоточиться на решении бизнес-задач. Именно здесь мы учимся писать полноценные приложения от начала и до конца.

Особенность этого уровня в том, что здесь не так много алгоритмической сложности как на первых этапах, но очень много инструментов, которые надо знать и применять на практике. Размер программ на этом уровне сильно вырастает, они переходят уровень в 1000 строк.

Computer Science

Даже если ваша цель — практическое программирование, полезно иметь общее представление о фундаментальных концепциях компьютерных наук:

- **Системы счисления:** двоичная и шестнадцатеричная системы, которые лежат в основе работы компьютеров.
- **Алгоритмы:** понимание того, как эффективно решать задачи, будь то поиск, сортировка или работа с большими данными.
- **Структуры данных:** списки, стеки, очереди, деревья и графы, которые помогают организовывать информацию и работать с ней.



Эти знания делают вас более уверенным программистом, который понимает, как работают технологии "под капотом", и способен оптимизировать код или справляться с нетривиальными задачами.

В хороших программах обучения, эти темы присутствуют и изучаются. Если вы учитесь самостоятельно, то будет полезным прочитать какую-нибудь книгу по информатике. А для алгоритмов взять [Грокаем Алгоритмы](#).

Искусственный интеллект (ChatGPT)

Чтобы вы не учили и с какой задачей не сталкивались, ИИ будет вашим самым эффективным помощником. Поэтому буквально с самых первых дней обучения,

настройте к нему доступ и начните практиковаться в его использовании. С чем он вам поможет?

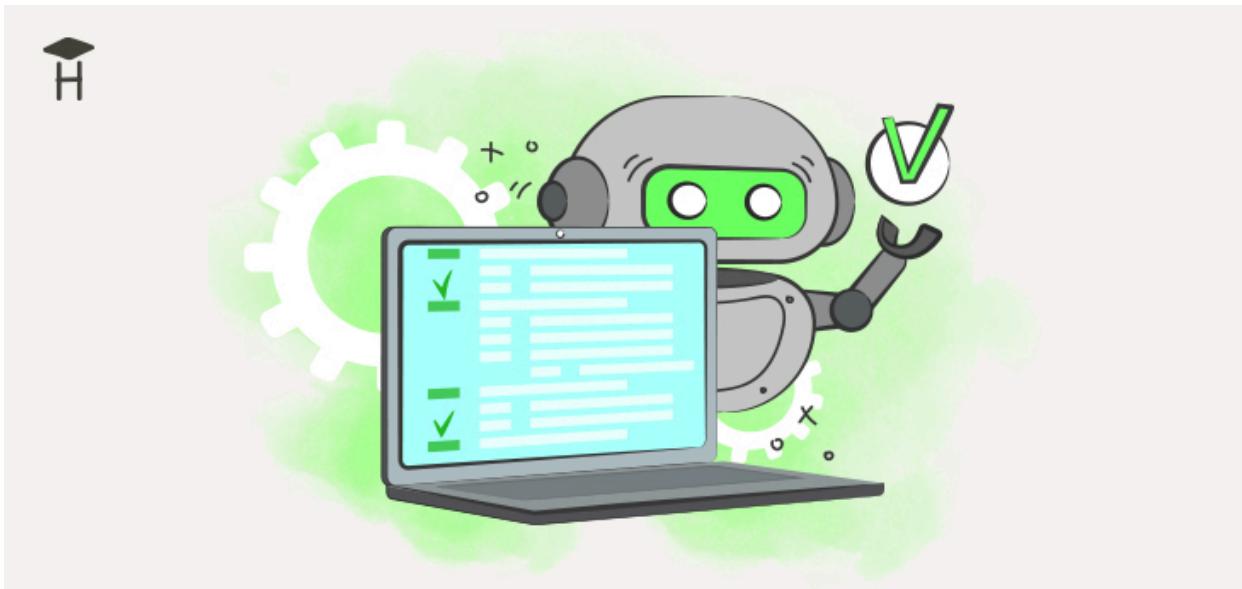
Альтернативное мнение: Изучая какую-то тему, легко зациклиться на одном источнике, например одном курсе, одной книге или статье. Из-за этого мы, часто, изучаем новое только с одной, авторской, стороны, которая не всегда очевидна конкретно нам. Дело в том, что не существует универсальных объяснений, одинаково хорошо подходящих всем. Многое зависит от бекграунда конкретного человека, кому-то зайдет аналогия на примере вождения мотоцикла, а других такой подход еще больше запутает. И здесь на помощь приходит ИИ. Вы не поняли объяснение переменных? Попробуйте поговорить об этом с chatgpt, он объяснит по другому. Если нет, то его можно попросить переформулировать и он объяснит по другому.

Примеры: Чем более абстрактная или техническая тема на изучении, тем больше нужно примеров для понимания. А в учебных материалах примеров может не хватить, либо объяснение примеров будет не полным. Это, кстати, не всегда вина авторов, иногда ограничения определяются контекстом, а иногда это особенность процесса обучения и бекграунда конкретного человека. В любом случае, ИИ позволяет восполнить недостаток примеров с лихвой. Ему достаточно указать тему и скинуть пример кода, объясняющий ее. Затем попросите chatgpt привести другие примеры по этой же теме. Результат вас приятно удивит.

Разбор ошибок: Самое сложное и демотивирующее в обучении программированию это постоянные ошибки, которые выдает запуск кода. И дело даже не в том что их много, это нормально и происходит даже у опытных разработчиков. А в том, что без поддержки от опытного наставника, начинающие разработчики сидят над попыткой понять чего от них хотят часами и даже днями. Существует эмпирическое правило, если ошибка за полчаса не исправлена, то имеет смысл обратиться за помощью. И chatgpt тут как нельзя кстати. Он классно анализирует небольшие куски кода и с очень высокой вероятностью выдает правильный ответ, о том что не так. Ну или, как минимум, формирует гипотезы того, что можно сделать, посмотреть и поменять. Кстати, этот пункт у нас внедрен в платформу Хекслета как часть процесса обучения.

Улучшение кода: Код можно писать по разному и даже существует такое выражение как говнокод, которое используется для описания кода, который не всем нравится. Это может быть субъективной оценкой, но код действительно нужно уметь делать не только рабочим, но и правильным. Правильность включает в себя читаемость, эффективность, поддерживаемость и другие параметры. Самостоятельно выдавать такой код не получится если кода больше чем 10 строк. Тут нужна оценка со стороны, примерно как в спорте для достижения хороших результатов нужен тренер. ИИ и тут может помочь. Передайте chatgpt кусок кода, который вы анализируете и попросите его дать рекомендации по улучшению. Но будьте осторожны, в плане улучшения chatgpt может наговорить лишнего.

Опыт прикладной разработки

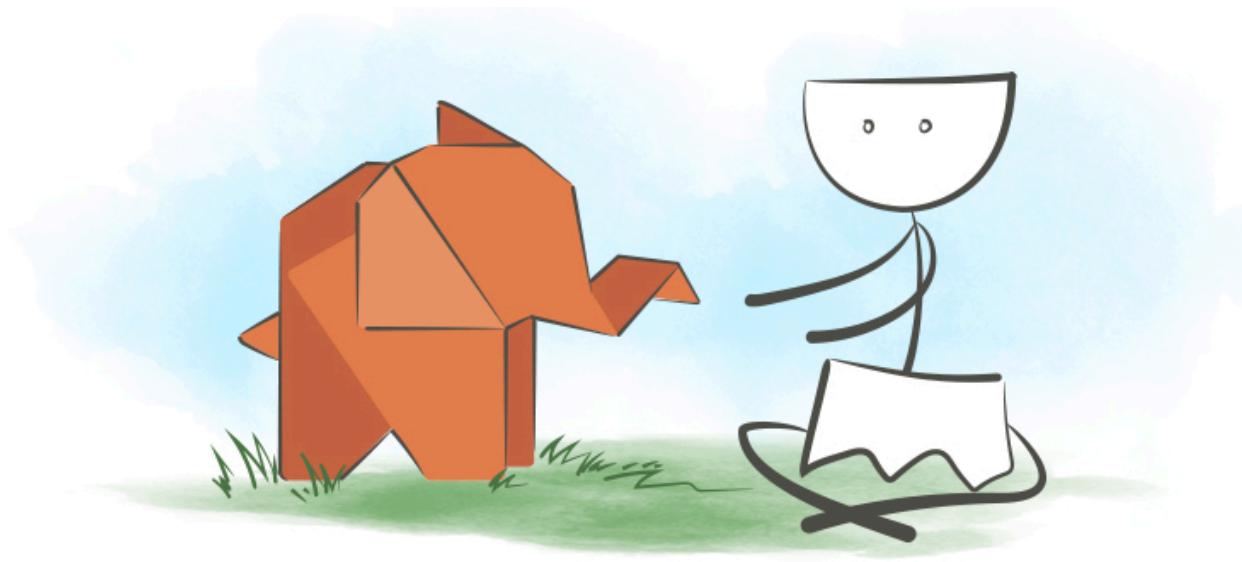


Даже если хорошо подготовились и научились создавать готовые приложения, этого может быть недостаточно для трудоустройства. Просто потому, что у вас нет никакого коммерческого опыта, вы даже не попадете на радары рекрутёров, которые занимаются подбором. Реальная проблема состоит не в том как пройти собеседование, а в том, как сделать так, чтобы на это собеседование позвали.

Коммерческий опыт, подразумевает опыт работы в коммерческих проектах. Такие проекты имеют одно принципиальное отличие от, например, учебных, ими пользуются живые люди. И на что это влияет, спросите вы? На все. Живые проекты это живые деньги, настоящие пользователи, риски что-то потерять, сломать, внести изменение, которое может вызвать гнев у аудитории проекта. Именно работа с рисками кардинально меняет не только отношение программистов, но и дает тот самый опыт, который называется коммерческим. Здесь уже нельзя в случае проблем просто все удалить и запустить заново. Поэтому помимо учебных проектов, нужно что-то еще.

В этом разделе, мы поговорим про то, где же брать этот опыт, который, с одной стороны даст вам, так необходимый на начальном этапе, опыт прикладной разработки, а с другой откроет дорогу к собеседованиям.

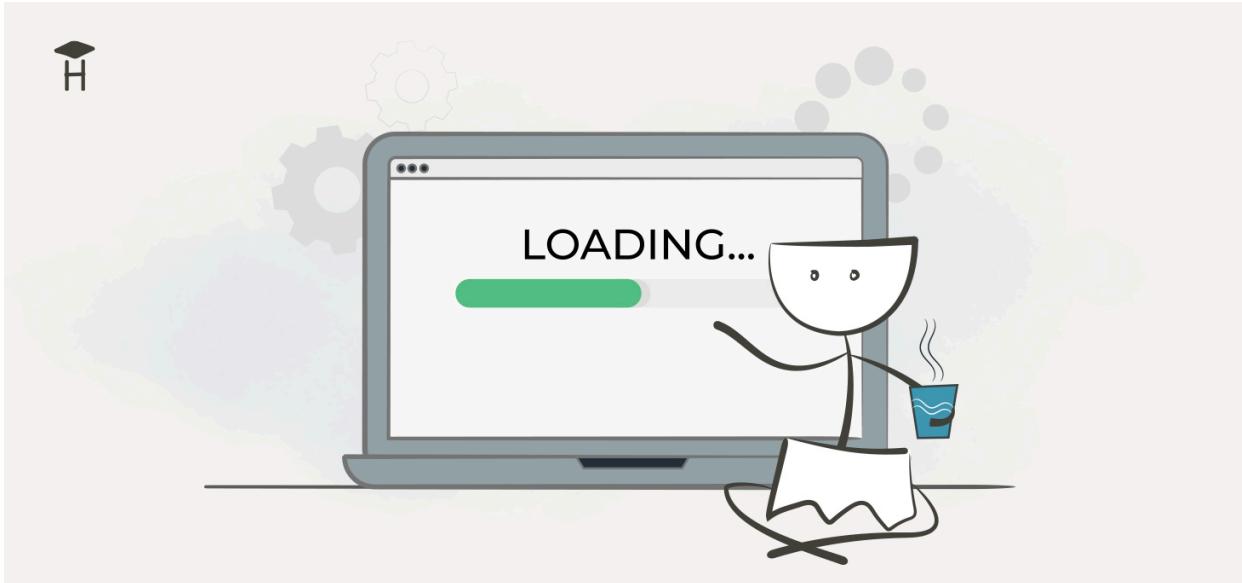
Учебные проекты



Учебные проекты — это первое, что появляется в портфолио новичка. Обычно они создаются в процессе курсов или самостоятельного обучения. Такие проекты могут быть простыми, но важно, чтобы они демонстрировали основные навыки: понимание основ программирования, умение использовать стандартные инструменты и создавать работающие приложения.

Учебные проекты это обязательная часть программы, даже, самостоятельного обучения, но их недостаточно, так как уровень работы с такими проектами никогда не сравнится с уровнем работы над коммерческими проектами.

Пет-проекты



Личные проекты, или пет-проекты. Это те проекты, которые вы создаете самостоятельно, исходя из своих интересов. Например, кто-то разрабатывает приложение для управления задачами, кто-то ботов для телеграмма, а кто-то создаёт мини-игру или сервис для учёта тренировок.

Для работодателей такие проекты имеют чуть более высокий приоритет чем учебные, при условии что они действительно работают и приносят пользу. То есть у них есть реальные пользователи, хотя бы вы сам, если это проект призван автоматизировать какие-то рутинные задачи.

Ну а если ими пользуются регулярно другие люди и их много, то такой опыт можно смело записывать в коммерческий.

Открытые проекты

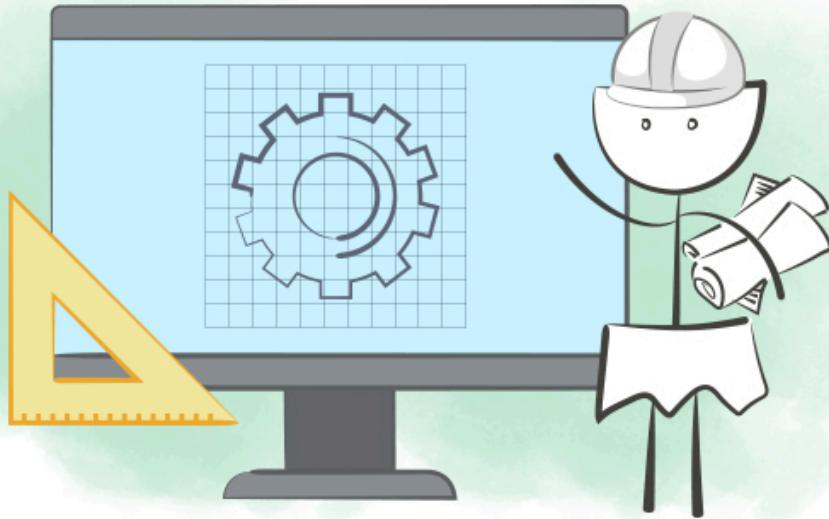


Открытые проекты (Open Source), это проекты, которые выкладываются на общий доступ разработчикам всего мира, для того, чтобы они могли принимать участие в их разработке. Такая модель может показаться странной, зачем участвовать в разработке чужих программ? Но у нее множество плюсов, обсуждение которых выходит за рамки этой книги. Главное, что таких проектов невероятно много и мы все пользуемся этими проектами. Например языки программирования сами по себе являются программами с открытым исходным кодом. Операционная система Linux и построенная на ее основе Ubuntu, тоже являются проектами с открытым исходным кодом.

Участие в разработке открытых проектов, особенно имеющих прикладную направленность, значительно повышает шансы на трудоустройство. Причем, в этом случае, вас могут заметить и пригласить другие разработчики, с которыми вы будете работать. Во-первых они обладают всеми качествами коммерческих проектов, во-вторых, все ваши действия видны, так как код открыт, а значит легко проверить что и как вы делали, в-третьих, у открытых проектов достаточно высокие требования к качеству кода и контроль со стороны других участников.

Где искать такие проекты? Большая их часть находится на сайте github.com, но их там огромное количество и многие из них требуют высокой квалификации для старта. Для упрощения этого процесса, мы создали [список проектов](#), которые, с одной стороны, подходят новичкам, с другой, представляют собой прикладные программы, аналогичные тому, что делаются в коммерции.

Волонтерские проекты



Волонтерские проекты — еще один отличный способ пополнить портфолио. Это могут быть приложения или сайты, созданные для некоммерческих организаций, локальных сообществ или небольших бизнесов. Такие проекты дают возможность не только развивать навыки, но и видеть, как ваша работа приносит пользу.

Где искать волонтерские проекты

- **Личные контакты:** Помогите своим друзьям, родителям или знакомым, которым нужны простые приложения, сайты или автоматизация процессов.
- **Онлайн-площадки:** [IT-волонтер](#) — популярный сайт, где публикуются запросы от некоммерческих организаций, локальных сообществ и благотворительных фондов. **Catchafire**, **VolunteerMatch** — международные платформы, где можно найти проекты для волонтеров с техническим фокусом.
- **Телеграм-группы:** В тематических каналах и чатах часто публикуются запросы на разработку для некоммерческих нужд.
- **Локальные сообщества:** Не забывайте о локальных инициативах в вашем городе. Например, библиотеки, школы или небольшие организации часто ищут людей, готовых создать сайт или приложение.

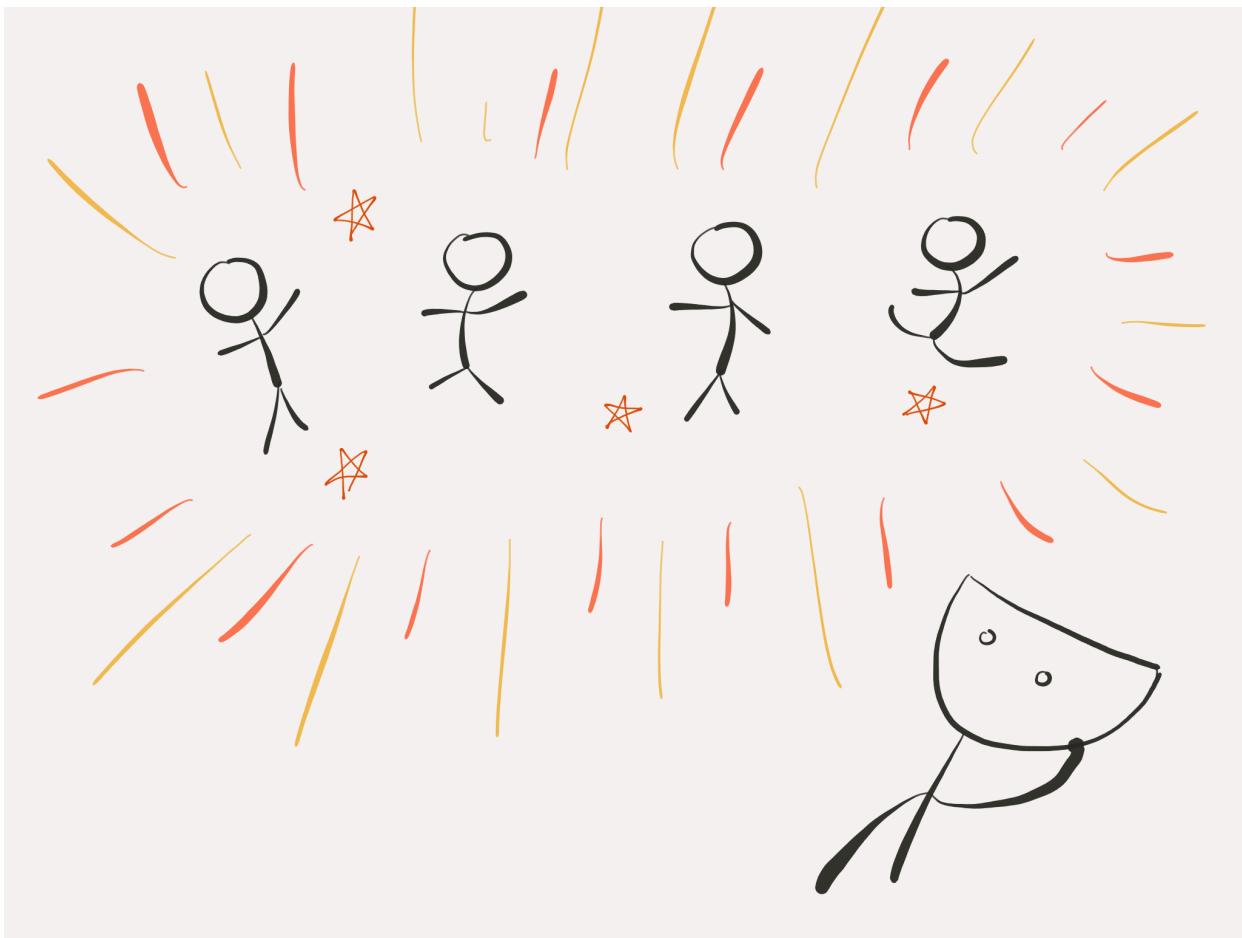
Составление резюме



Резюме — это ваш первый шаг на пути к трудоустройству. Это не просто список навыков и опыта, а ваше представление работодателю. Именно через резюме компании оценивают, стоит ли приглашать вас на собеседование. Поэтому важно составить его так, чтобы вы выделялись среди других кандидатов, показывая свои сильные стороны и готовность к работе.

Существует множество сервисов для поиска работы, но не все они хорошо работают. Среди популярных площадок для разработчиков можно назвать hh.ru, linkedin.com и career.habr.com. Это те места, где резюме размещать обязательная часть программы.

Как выделиться среди других кандидатов



Конкуренция в программировании высокая, особенно на начальных позициях, поэтому ваше резюме должно сразу привлекать внимание. Укажите не только ваши технические навыки, но и проекты, которые демонстрируют их применение. Даже если у вас пока нет опыта работы, пет-проекты, участие в открытых проектах или волонтерских инициативах могут показать, что вы активно развиваетесь.

Кроме того, адаптируйте резюме под каждую конкретную вакансию. Если компания ищет веб-разработчика, акцентируйте внимание на навыках и опыте, связанных с этим направлением. Это покажет работодателю, что вы понимаете их запросы и готовы решать именно их задачи.

Структура идеального резюме программиста



Хорошее резюме должно быть структурированным, лаконичным и информативным. Будьте внимательны к деталям. Проверьте резюме на ошибки — опечатки или отсутствие запятых могут испортить впечатление.

Вот ключевые разделы, которые стоит включить:

- **Контактная информация:** имя, телефон, электронная почта, ссылка на GitHub, LinkedIn или портфолио. Добавьте фотографию, резюме с реальными лицами выглядят более привлекательно.
- **Краткое описание:** 2–3 предложения о том, кто вы, какие навыки и опыт у вас есть, и чего вы хотите достичь.
- **Технические навыки:** укажите языки программирования, технологии, инструменты, с которыми вы работали.
- **Проекты:** опишите несколько ваших лучших проектов. Укажите их цели, какие технологии использовались, и какую задачу вы решили. Добавьте ссылки на репозитории или рабочие версии.
- **Образование:** если у вас нет профильного образования, можно указать курсы или сертификаты, которые вы прошли.
- **Опыт работы (если есть):** кратко опишите ваши обязанности и достижения при условии, что опыт релевантен позиции на которую вы претендуете

Обычно, вам не нужно придумывать разделы, которые нужно указывать в резюме. Резюме создаются на площадках для поиска работы, где уже есть готовые формы, которые остается только заполнить.

Рекомендации по оформлению резюме



Правильно оформленное резюме не только привлекает внимание, но и создает положительное первое впечатление. Работодатели часто тратят на просмотр резюме не более 10–15 секунд, поэтому важно сделать документ лаконичным, информативным и приятным для чтения.

Фраза "Я начинающий программист, ищу работу" не добавляет ценности вашему резюме. Вместо этого сконцентрируйтесь на своих достижениях и навыках. Например, подчеркните завершенные проекты, участие в open-source, изученные технологии или решённые задачи. Работодатели хотят видеть, что вы не просто хотите работать, а готовы приносить пользу.

Резюме должно демонстрировать, что вы можете решить задачи работодателя. Опишите, чего вы достигли, даже если это учебные проекты или личные инициативы. Например:

- "Разработал веб-приложение для управления задачами, используя React и Node.js, с более чем 50 активными пользователями."
- "Внёс вклад в open-source проект, улучшив производительность модуля на 15%."

Достижения всегда более убедительны, чем просто перечисление навыков. Они показывают, как вы применяете свои знания на практике.

Избегайте перегруженности текста. Не пишите о незначительных деталях, которые не связаны с вакансией. Фокусируйтесь на том, что релевантно для должности, на которую вы претендуете.

Автофильтры

У оформления резюме есть скрытая сторона. Она связана не с вашими достижениями, а с тем как работают системы отбора кандидатов (ATS). Например, фильтры по годам опыта или возрасту.

The screenshot shows a web browser window for hh.ru with the URL hh.ru/employer/vacancyresponses/autoaction?vacancyId=116202. The page is titled "Автоматизация откликов и приглашений" (Automation of responses and invitations) and includes a sub-section "Авторазбор откликов" (Automated review of responses). On the left, there's a sidebar with categories like "Воронка подбора" (Recruitment funnel), "Все неразобранные" (All unread), "Подумать" (Think), and "Не подходит" (Not suitable). The main area contains sections for filtering candidates based on resume parameters, responses from chatbots, where to forward them, and addresses.

Во времена, когда работы мало, а кандидатов много, рекрутеры нередко включают автоматическую фильтрацию из-за которой ваше резюме и сопроводительное письмо никто не увидит.

На этом этапе рождается набор честных и не очень рекомендаций, по тому как эти фильтры пройти. Начиная от указания места жительства в Москве, до подкрутки опыта до желаемой цифры, например у вас 11 месяцев опыта, но вы ставите один год, чтобы пройти фильтр.

Понимание принципов работы этой системы важно для понимания процесса трудоустройства и повышения шансов на собеседование. Однако, будьте аккуратны, не все из этих "подкруток" невинны и могут иметь последствия. Например указать, что вы живете в Москве при условии, что вы готовы переехать как только найдете работу, это вполне нормальная техника, но приписать себе 3 года опыта не имея никакого опыта вообще - перебор по нашему мнению.

Из-за повышенного количества людей пытающихся обойти систему, многие аспекты подбора сейчас активно меняются. Все снова возвращается в офлайн (легче проверить реальность людей и их навыков), перестает ценится фриланс (как доказать что это не накрутка?), проверяются референсы (что человек реально работал там где указал и столько сколько указал).

Поиск работы и прохождение собеседований

Поиск работы — это не просто отправка резюме на вакансии, это процесс, требующий стратегии и подготовки. Чтобы повысить свои шансы на успех, важно знать, где искать вакансии, как правильно откликаться и что ожидать на собеседованиях. Также полезно изучить тактики точечного трудоустройства, которые помогут целенаправленно выйти на работу в конкретную компанию.

Где искать вакансии: платформы и сообщества

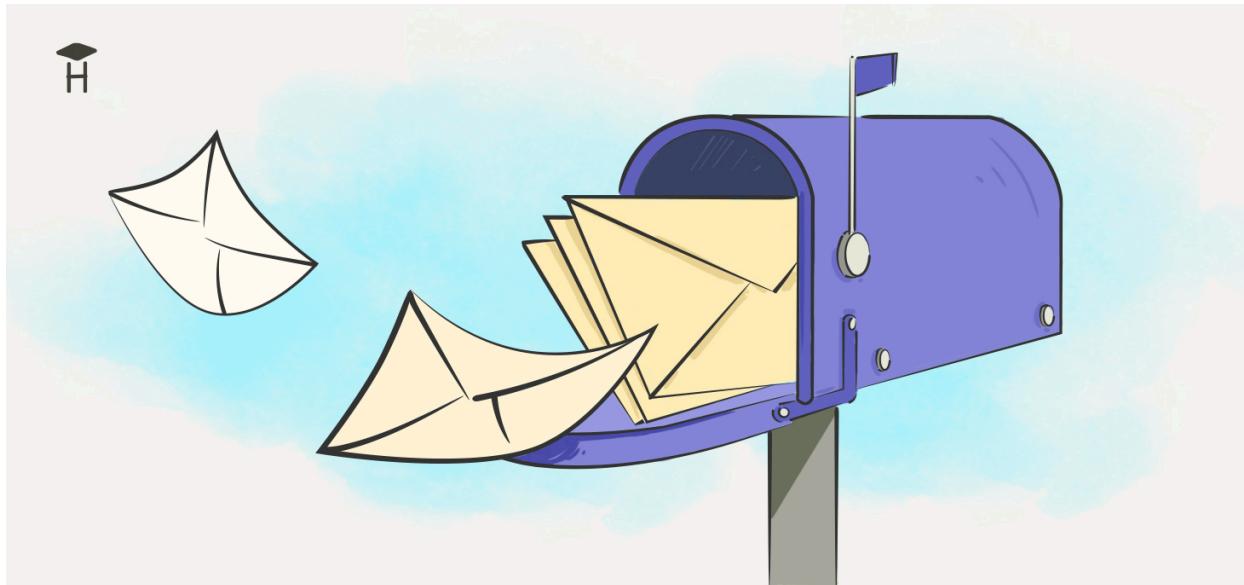


Начните поиск работы с проверенных платформ и сообществ. Такие сайты, как hh.ru, Geekjob и профильные ресурсы вроде Habr Career или LinkedIn, предлагают множество вакансий для разработчиков.

Не ограничивайтесь только платформами. [Сообщества в Telegram](#), чаты и профессиональные форумы часто публикуют вакансии, которые еще не успели попасть на крупные сайты. Включайтесь в общение с другими разработчиками, участвуйте в обсуждениях, задавайте вопросы. Часто вакансии находят тех, кто вовлечен в профессиональные сообщества. Большая часть таких сообществ располагается в телеграм-группах и каналах. Актуальную информацию о них можно найти на сайте tgstat.com

Крупные компании (бигтех) публикуют вакансии в первую очередь на своих сайтах, а уже после на платформах. Это связано с тем, что за ними многие следят и подают заявки напрямую.

Как откликаться на вакансии?



Отклик на вакансию должен быть персонализированным. Прежде чем отправлять резюме, внимательно изучите описание работы и требования.

По возможности пишите сопроводительное письмо. Да, его могут не посмотреть, но оно все равно повышает шансы на успех, так как вы можете зацепить людей с той стороны, чем-то привлекательным для них. Например вы постоянный клиент компании, куда хотите трудоустроиться или работали в смежной компании, пусть даже не на позиции разработчика. Здесь нет единого правила, но какие-то подробности о ваших достижениях и связь с компанией могут дать дополнительные очки при отборе.

Если вы новичок, сделайте акцент на своих проектах и навыках, которые совпадают с ожиданиями работодателя. Укажите, почему вы заинтересованы в этой роли и как планируете развиваться в компании. Персонализированный подход выделяет вас среди множества кандидатов, отправляющих стандартные отклики.

Что проверяют на собеседованиях программистов?



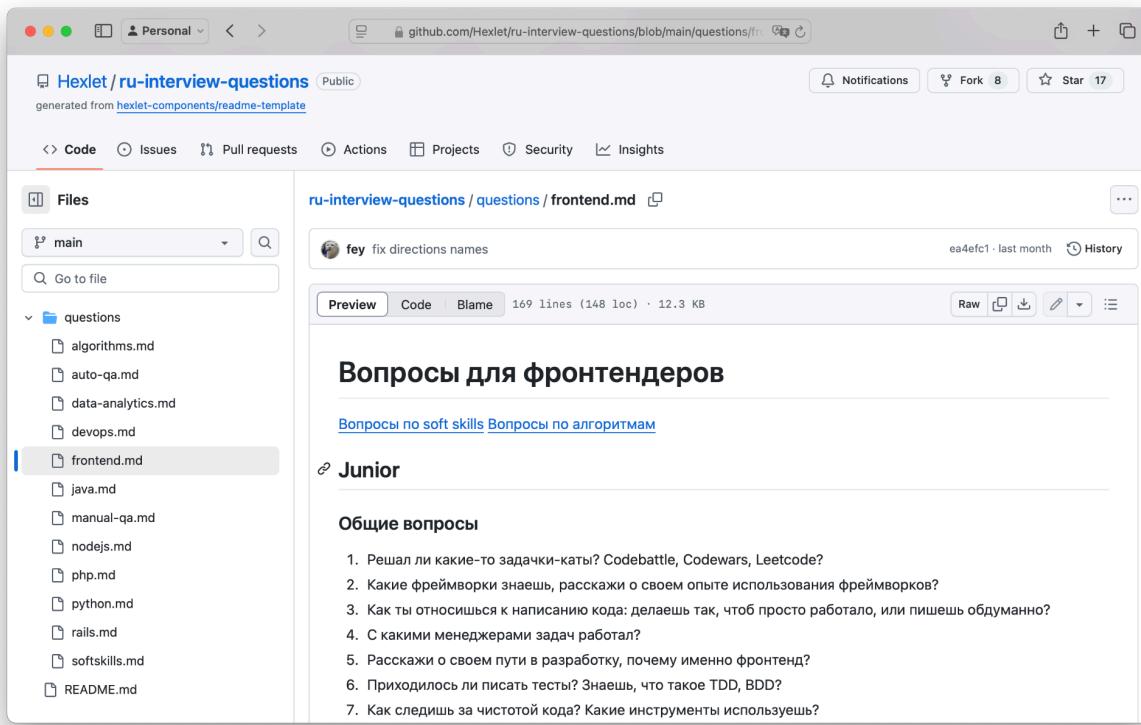
Собеседование программиста включает несколько этапов, каждый из которых проверяет разные аспекты. В крупных компаниях этапам соответствуют отдельные собеседования, которые делятся 1-2 часа. В небольших все это может проверяться в рамках одного собеседования. Вот что туда может входить:

- **Технические навыки:** задачи на алгоритмы, решение проблем, понимание основ программирования.
- **Практические умения:** выполнение тестового задания или лайвкодинг, когда код пишется прямо во время собеседования.
- **Софт-скиллы:** умение работать в команде, способность объяснять свои решения и готовность учиться.
- **Бизнес-контекст:** понимание целей компании и задач продукта.

Даже опытные разработчики могут испытывать трудности при прохождении собеседований, потому что умение решать поставленные задачи и проходить собеседования это не одно и тоже. К собеседованиям надо готовиться иначе на успех рассчитывать не приходится.

Подготовка к собеседованию включает в себя несколько элементов. Первый это алгоритмическая подготовка, куда входит умение решать классические задачи на алгоритмы и структуры данных. Для этого, обычно, берут сервис leetcode.com и решают сотню другую задач. Второй это подготовка к ответам на вопросы, которые обычно задают по вашему направлению. В интернете полным полно списков вопросов для собеседований любых специалистов. Другой вопрос, что их даже слишком много и они все они актуальны. Поэтому сейчас можно найти примеры списков составленных из [реально задаваемых вопросов](#). Главное при подготовке не скатываться в зурбажку. Нужно понимать о чём идет речь. Ну и третий элемент, подходящий для новичков, это просмотр

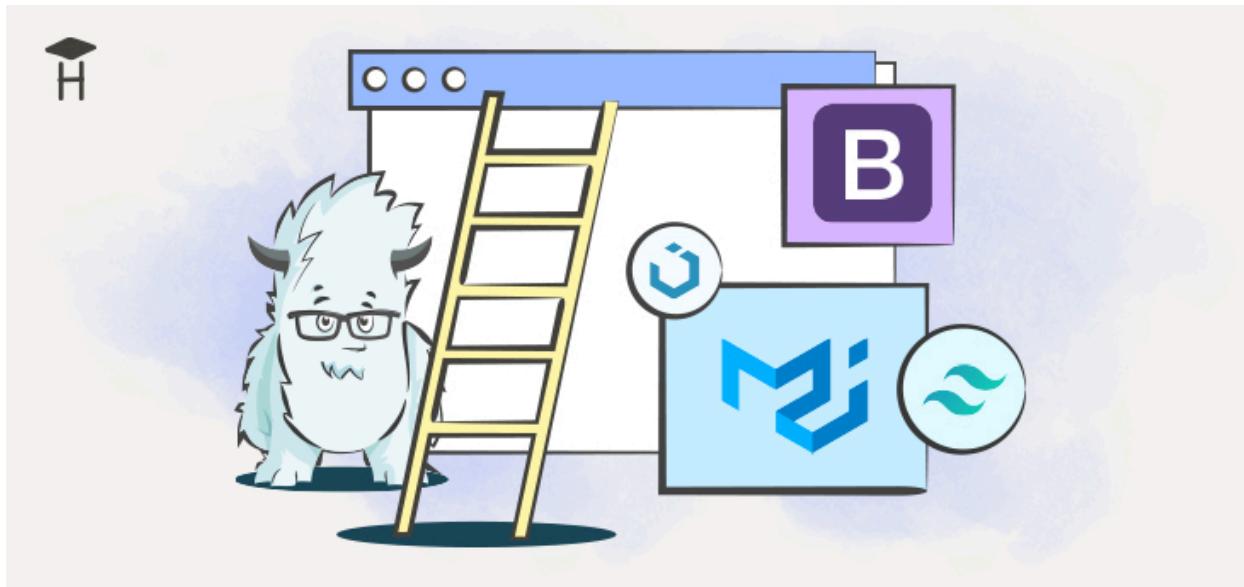
мок-собеседований например, [здесь](#). Так вы узнаете как проводят собеседования настоящие программисты и какие вопросы они задают.



Несмотря на всю подготовку, пройти собеседование не так то просто и многие проходят их десятками перед тем как получают реальный офер. Но не нужно расстраиваться, каждое новое собеседование это новый опыт, который подсвечивает пробелы в знаниях или отсутствие какого-то опыта. Главное, здесь правильные выводы и домашняя работа над слабыми местами, чтобы на следующем собеседовании показать себя лучше.

Тестовые задания

Некоторые компании встраивают в процесс найма выполнение тестового задания. Его дают специалистам разного уровня, но чаще это делается для проверки навыков у начинающих разработчиков. Что в нашем случае большой плюс, так как тестовые задания являются серьезным барьером для тех, кто не умеет программировать. Технически его может сделать другой человек, но во время собеседования, собеседующий обязательно задаст вопросы, относительно того, как это задание выполнялось.



Разработчики очень по-разному относятся к тестовым заданиям, кто-то считает, что они хорошо помогают, кто-то считает что проще не связываться с такой компанией и найти другую. Мой личный опыт плотно связан с тестовыми, в половине компаний, которые меня нанимали я выполнял тестовое задание. В основном в первые годы моей карьеры разработчиком.

На практике многое зависит от ситуации на рынке. Если работы много, а специалистов мало, то они могут себе позволить выбирать и игнорировать компании, которые просят выполнять тестовые. Если работы мало, то количество людей выполняющих тестовые начинает расти. У новичков выбора особо нет, если вам дают тестовое задание, то лучше хвататься за эту возможность.

README

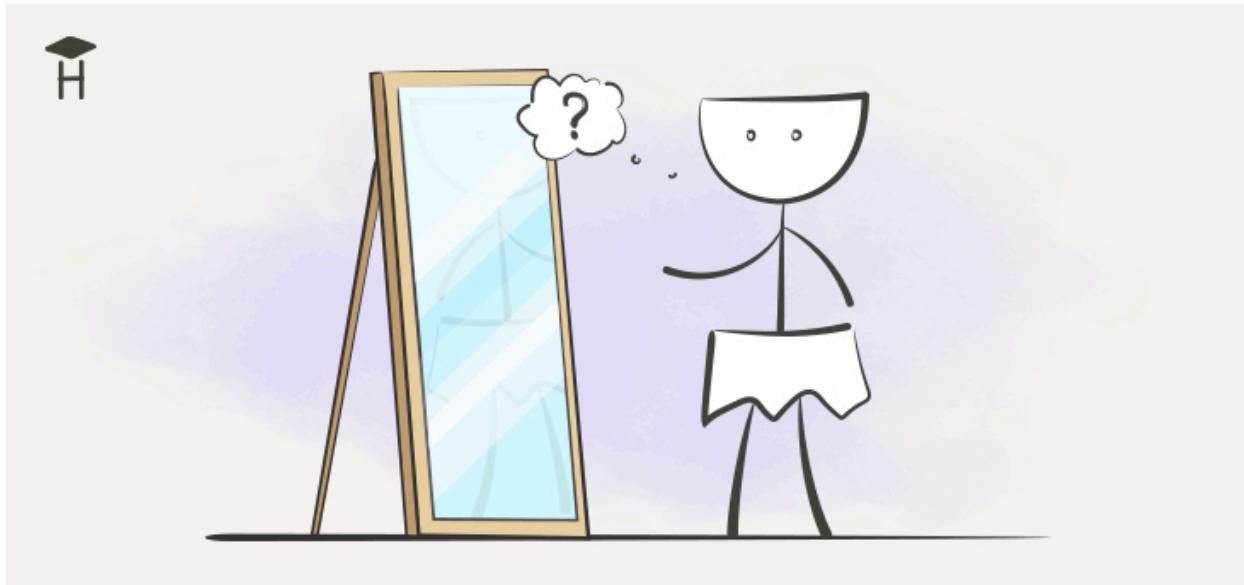
Задания

- Аналитик
- Android
- Automation QA
- Manual QA
- ▼ Backend
 - [Appbooster](#)
 - [Backend developer \(any\)](#)
 - [Appstorespy](#)
 - [Junior Backend developer \(Django/Flask, MySQL/PostgreSQL, MongoDB/Redis/Elasticsearch, GraphQL/REST\)](#)
 - [avito.tech](#)
 - [Тестовое задание для стажёра Backend в команду Advertising \(Go/PHP\)](#)
 - [Тестовое задание на позицию стажёра-бекендера в юнит Авто \(archived\) \(Go/Python/PHP/Java/JavaScript\)](#)
 - [Тестовое задание на позицию стажёра backend в юнит Geo \(any\)](#)
 - [Тестовое задание на позицию стажёра-бекендера \(archived\) \(any\)](#)
 - [Тестовое задание для стажёра Backend \(PHP\)](#)
 - [Тестовое задание для стажёра Backend в команду Trade Marketing \(Go/PHP/Python, PostgreSQL/MySQL, Redis\)](#)

Что если задание длинное и требует неделю на выполнение? Даже в этом случае. Кроме того, оценка сроков очень зависит от уровня человека. Если вам кажется что задание займет неделю, то, возможно, вам еще не хватает опыта, а в реальности, его можно сделать за один день.

Из дополнительных плюсов. Тестовое задание выполненное для одной компании, нередко может быть интересно собеседующим из другой компании. Обязательно скажите об этом в сопроводительном письме или упомяните на собеседовании.

Как вести переговоры о зарплате?



Обсуждение зарплаты — это деликатный этап. Перед собеседованием изучите рынок: узнайте, какие зарплаты предлагают на аналогичных позициях в вашем регионе. Во время переговоров будьте честны о своих ожиданиях, но не забудьте показать гибкость.

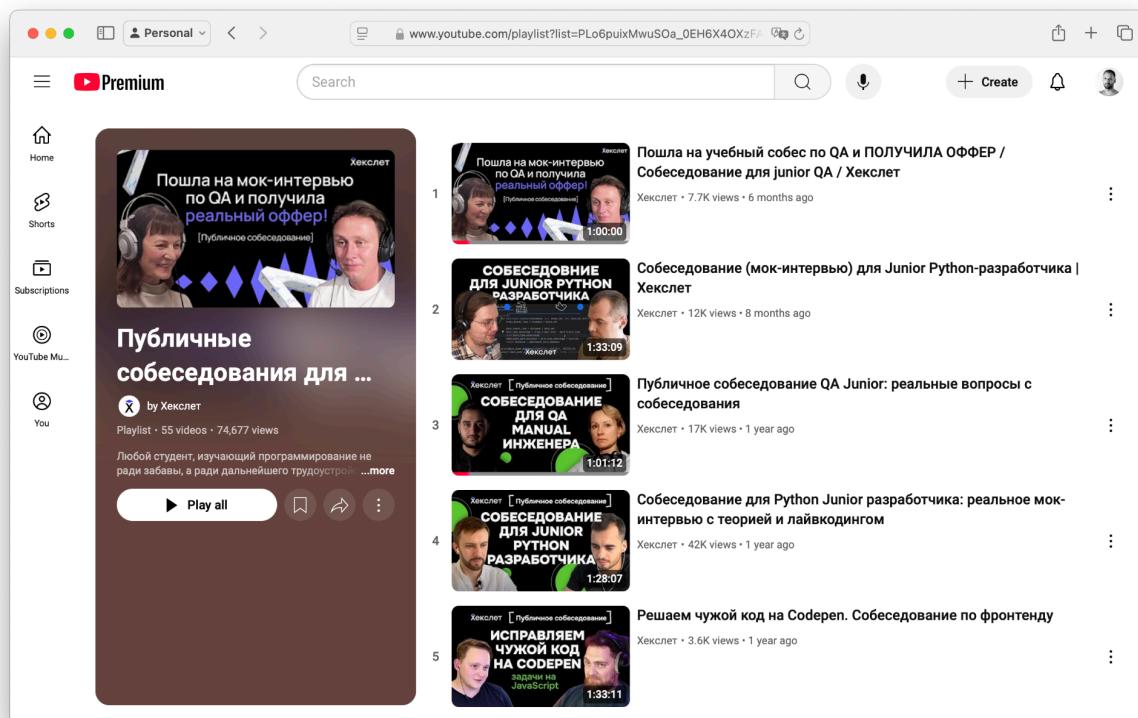
Если вы новичок, акцентируйте внимание на том, что для вас важнее возможности роста и обучения. Работодатели ценят кандидатов, которые готовы расти вместе с компанией.

Точечное трудоустройство



Точечное трудоустройство — это стратегия, направленная на поиск работы в конкретной компании. Если вы хотите работать в определённой компании, начните с изучения её профиля на LinkedIn. Посмотрите, кто работает в интересующей вас команде, и попробуйте наладить контакт.

Например, вы можете написать сотруднику компании с просьбой об обратной связи по вашему портфолио или резюме. Часто такие контакты превращаются в менторство, где опытный специалист помогает вам подготовиться к интервью.



Полезно также запросить мок-интервью. Это пробное собеседование, которое имитирует реальное, но проводится в неформальной обстановке. Вы получите полезные советы и улучшите свои шансы на успешное прохождение настоящего собеседования.

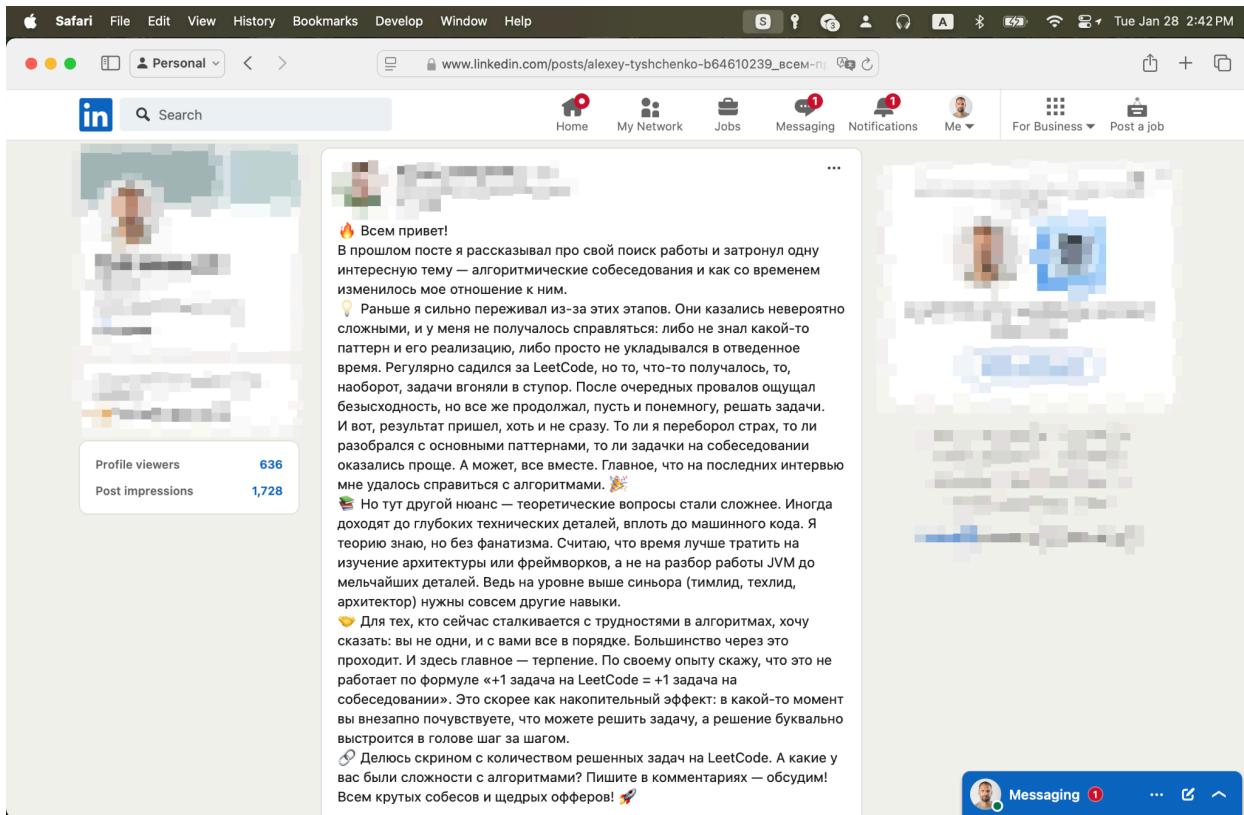
Нетворкинг

Нетворкинг — это искусство построения профессиональных связей, которое играет важную роль в карьере программиста. Большинство вакансий в IT заполняется благодаря рекомендациям и личным контактам. Общение с единомышленниками, участие в профессиональных мероприятиях и активность в сообществах могут не только расширить кругозор, но и помочь найти работу или интересный проект.



Линкедин

LinkedIn — одна из самых мощных платформ для профессионального нетворкинга. Создайте подробный профиль, где будет отражён ваш опыт, навыки, проекты и достижения. Публикуйте посты о вашем обучении, участии в проектах или размышлениях о программировании.



Не бойтесь отправлять запросы на добавление в друзья сотрудникам компаний, в которых вы хотите работать, или другим разработчикам. Пишите короткие сообщения с приветствием и расскажите, почему вам интересен их профиль. Участвуйте в обсуждениях под постами и делитесь своим мнением — это поможет вам быть замеченным.

Сообщества (группы, Помощь новичкам Q&A)

Сообщества программистов в Telegram или форумах, таких как Stack Overflow и Habr Q&A, — отличное место для получения советов, обсуждения идей и решения вопросов. Такие группы часто организованы по направлениям, например, веб-разработка или работа с данными.

The screenshot shows a web browser displaying the HABR Q&A section. The left sidebar includes links for 'Войти' (Login), 'Все вопросы' (All questions), 'Все теги' (All tags), and 'Пользователи' (Users). A central column lists several questions with their titles, tags, and statistics. To the right, a sidebar displays 'САМОЕ ИНТЕРЕСНОЕ ЗА 24 ЧАСА' (Most interesting in 24 hours) with various questions and their details.

Категория	Вопрос	Ответы
Полная разница sudo su, sudo -i, sudo -s?	3 ответа	
Какой выбрать роутер для обхода блокировок?	11 ответов	
Как выглядит sudoers?	4 ответа	
Как сделать spinner в стиле выбора даты?	1 ответ	
Не запускается Selenium в докере?	0 ответов	
Восстановление виртуальной машины vmware ESXi?	0 ответов	

Общение в сообществах не только помогает развивать навыки, но и укрепляет связи с другими программистами. Активность в вопросах и обсуждениях может привести к приглашениям в проекты, коллaborациям или даже предложениям о работе.

Хакатоны и воркшопы

Хакатоны — это интенсивные соревнования, где программисты в командах решают задачи или создают проекты за ограниченное время. Это отличная возможность не только проявить свои навыки, но и познакомиться с новыми людьми. Командная работа позволяет вам оценить свои сильные и слабые стороны, а участие в завершенных проектах обогащает портфолио.

The screenshot shows a web browser window displaying the HackDay #70 event page. The header includes the HackDay logo, navigation links for 'Personal', 'HackDay #70', 'О мероприятиях', 'Проекты', 'Участники', and a search bar. On the right is a blue button labeled 'Зарегистрироваться'. Below the header, there's a large gradient background image. In the center, text indicates the event date '3-4 февраля, 2024' and location 'Санкт-Петербург. Failover Bar, 4-я советская, д. 7'. The title 'HackDay #70' is prominently displayed, followed by the subtitle 'Старый добрый технический хакатон' and a note '[Событие завершено]'. A section titled 'О мероприятии' contains text about the event's return after a long break and its format. Another section titled 'Программа' lists the schedule for February 3rd, including a gathering at 11:30 and an official opening at 12:00.

Воркшопы — это образовательные мероприятия, где участники учатся чему-то новому под руководством опытных разработчиков. Это не только способ освоить новые технологии, но и шанс наладить контакты с организаторами и другими участниками.

Конференции

Профессиональные конференции — место, где собираются программисты, эксперты и компании. Вы можете слушать доклады, посещать мастер-классы и участвовать в обсуждениях.

Personal nastachku.ru

Стачка

БИЛЕТЫ СТАТЬ ПАРТНЕРОМ ЛИЧНЫЙ КАБИНЕТ

Стачка 2025

XIII Международная IT-конференция "Стачка"

Ульяновск
18-19 апреля 2025

СМОТРЕТЬ ПОДРОБНЕЕ

XIV Международная IT-конференция "Стачка"

Санкт-Петербург
2-3 октября 2025

СМОТРЕТЬ ПОДРОБНЕЕ

Прошедшие события

XII Международная

Отправьте нам сообщение **живо**

Конференции — это не только обучение, но и способ познакомиться с профессионалами индустрии. Будьте активны: подходите к спикерам с вопросами, обсуждайте темы с другими участниками, добавляйте их в LinkedIn. Множество компаний на конференциях ищут новые таланты, поэтому такие события могут стать отличной возможностью для трудоустройства.

Как учиться эффективно?

Эффективное обучение программированию — это не только усвоение теории, но и умение применять знания на практике. Чтобы сделать процесс обучения продуктивным и избежать усталости, важно подходить к этому с планом, выбирая подходящие ресурсы и учитывая свои возможности.

Выбор курсов и материалов

Существует огромное количество курсов и материалов по программированию, поэтому важно выбрать те, которые соответствуют вашим целям и уровню подготовки. Обращайте внимание на репутацию платформ и отзывы студентов. Хороший курс должен включать практические задания, которые помогут закрепить теорию, и поддерживать обратную связь — через тесты, проекты или работу с наставником.



Не ограничивайтесь только курсами. Дополнительно используйте книги, блоги, документацию и видеоуроки. Разные форматы помогут лучше понять сложные темы.

Самообразование или наставник?

Самообразование дает свободу выбора темпа и направлений, но требует дисциплины и навыка планирования. Если вы уверены, что сможете организовать свой процесс обучения, самостоятельно изучать материалы и регулярно практиковаться, этот путь подойдет вам.



Однако, если вы чувствуете, что нуждаетесь в поддержке, наставник может значительно ускорить процесс. Он поможет избежать типичных ошибок, подскажет, на чём сосредоточиться, и направит ваши усилия в нужное русло. Наставника можно найти на платформах, предлагающих персональное обучение, или в сообществах программистов.

Избегаем выгорания



Изучение программирования требует много усилий, и без должного внимания к своему состоянию можно столкнуться с выгоранием. Чтобы этого избежать, важно следовать некоторым правилам:

- Чередуйте учебу с отдыхом. Делайте регулярные короткие перерывы, чтобы не терять концентрацию.
- Устанавливайте реалистичные цели. Не пытайтесь охватить всё сразу. Дробите задачи на маленькие шаги.
- Следите за своим физическим и эмоциональным состоянием. Занимайтесь спортом, высыпайтесь и находите время на общение с друзьями или хобби.
- Не бойтесь сделать паузу. Если чувствуете, что устали, возьмите небольшой перерыв, чтобы восстановиться.

Эффективное обучение — это не гонка, а марафон. Постепенное, сбалансированное развитие поможет вам не только освоить программирование, но и сохранить интерес к этому на долгие годы.

Развитие карьеры программиста

Карьерный путь программиста — это непрерывное развитие навыков, знаний и опыта. В процессе вы будете сталкиваться с новыми вызовами, выбирать между различными вариантами работы и постепенно повышать свою профессиональную ценность.

От джуниора до синьора: ключевые навыки



Переход от начального уровня к высококвалифицированному специалисту — это долгий, но увлекательный процесс. Джуниор-программист (Junior) начинает с выполнения простых задач под руководством наставника, изучая кодовую базу и инструменты. На этом этапе ключевые навыки включают понимание основ программирования, способность учиться на практике и умение задавать вопросы.

На уровне middla (Middle) программист уже способен решать задачи средней сложности самостоятельно, понимает процессы разработки и может давать оценку времени выполнения задач. Ключевыми навыками становятся управление временем, работа с документацией и способность принимать технические решения.

Синьор (Senior) — это не только мастер технических навыков, но и лидер. Он принимает участие в проектировании систем, помогает другим разработчикам расти, эффективно решает сложные задачи и может предложить оптимальные архитектурные решения. Умение видеть проект в контексте бизнеса, понимать цели продукта и адаптировать свои решения под эти цели становится критически важным.

Выбор между продуктовой и аутсорсинговой компанией

В процессе карьеры вы столкнетесь с выбором: работать в продуктовой или аутсорсинговой компании. Продуктовые компании создают и развиваются собственные продукты. Работа здесь чаще всего подразумевает глубокое погружение в одну систему, постоянное улучшение функционала и фокус на долгосрочной перспективе. Такие компании часто привлекают тех, кто хочет видеть результаты своей работы и влиять на конечный продукт.



Аутсорсинговые компании предоставляют услуги другим организациям. Это работа над разными проектами, часто с ограниченными сроками. Она подходит тем, кто хочет попробовать себя в разнообразных задачах, быстро набираться опыта и работать с разными технологиями.

Каждый путь имеет свои преимущества, и выбор зависит от ваших целей и предпочтений.

Фриланс или работа в офисе?



Работа в офисе предлагает стабильность, командное взаимодействие и возможность учиться у более опытных коллег. Это хороший выбор для начинающих программистов, которым важно получить структуру и наставничество.

Фриланс, напротив, даёт свободу выбора проектов и гибкость в планировании времени. Однако он требует высокой самоорганизации, способности искать клиентов и брать на себя ответственность за весь цикл разработки. Этот путь чаще выбирают более опытные специалисты, готовые работать независимо.

Возможен и третий вариант — удаленная работа в компании. Это сочетание стабильности офиса и гибкости фриланса, которое становится всё более популярным.

Заключение

Программирование как дверь в будущее

Программирование — это не просто профессия, а ключ к миру возможностей.

Программирование предлагает свободу выбора: вы можете работать в компании или самостоятельно, развивать собственные проекты или участвовать в разработке мировых технологий. Это универсальный инструмент, который пригодится в любой сфере, от медицины до космических исследований.

Советы начинающим



Начните с малого. Не пытайтесь сразу охватить всё — это может привести к усталости и потере мотивации. Учитесь шаг за шагом, закрепляя базовые навыки и переходя к более сложным темам. Практика важнее теории: пишите код каждый день, создавайте проекты, пробуйте новые технологии.

Не бойтесь задавать вопросы. Учеба — это процесс, в котором ошибки и непонимание неизбежны. Обращайтесь за помощью к сообществу, коллегам и наставникам.

Сохраняйте любопытство и открытость к новому. Технологии быстро развиваются, и программисты всегда находятся в процессе обучения. Готовность учиться — это один из главных факторов успеха.

Приложения

Рекомендованные книги, курсы и ресурсы.

- <https://ru.hexlet.io/courses> – курсы с наставником и помощью в трудоустройстве
- <https://code-basics.com/ru> – Бесплатные интерактивные курсы по большинству языков программирования
- <https://codebattle.hexlet.io/> – Тренажер для практики алгоритмов

Список полезных инструментов

- <https://github.com/Hexlet/ru-test-assignments> – большой список тестовых заданий
- <https://github.com/Hexlet/ru-interview-questions> – вопросы с реальных собеседований
- https://www.youtube.com/watch?v=JC2_0Efq18k&list=PLobruixMwuSOa_0EH6X4OXzFAmyQGS3a3 – Мок-собеседования

