

Гайд

Материалы



Ваш первый язык программирования: гид для начинающих

Итак, вы хотите ворваться в программирование, но там десятки языков и непонятно, за что браться. Пусть этот гайд вам поможет. Здесь мы разберём разные направления в сфере разработки и рассмотрим отдельные языки. Но сначала — тезисы о карьере разработчика как таковой.

Читайте дальше



Прежде чем выбирать язык

Вот несколько соображений, которые могут оказаться полезными ещё до того, как вы откроете каталог курсов и языков. Они касаются карьеры разработчика как таковой, а не конкретно технологий.

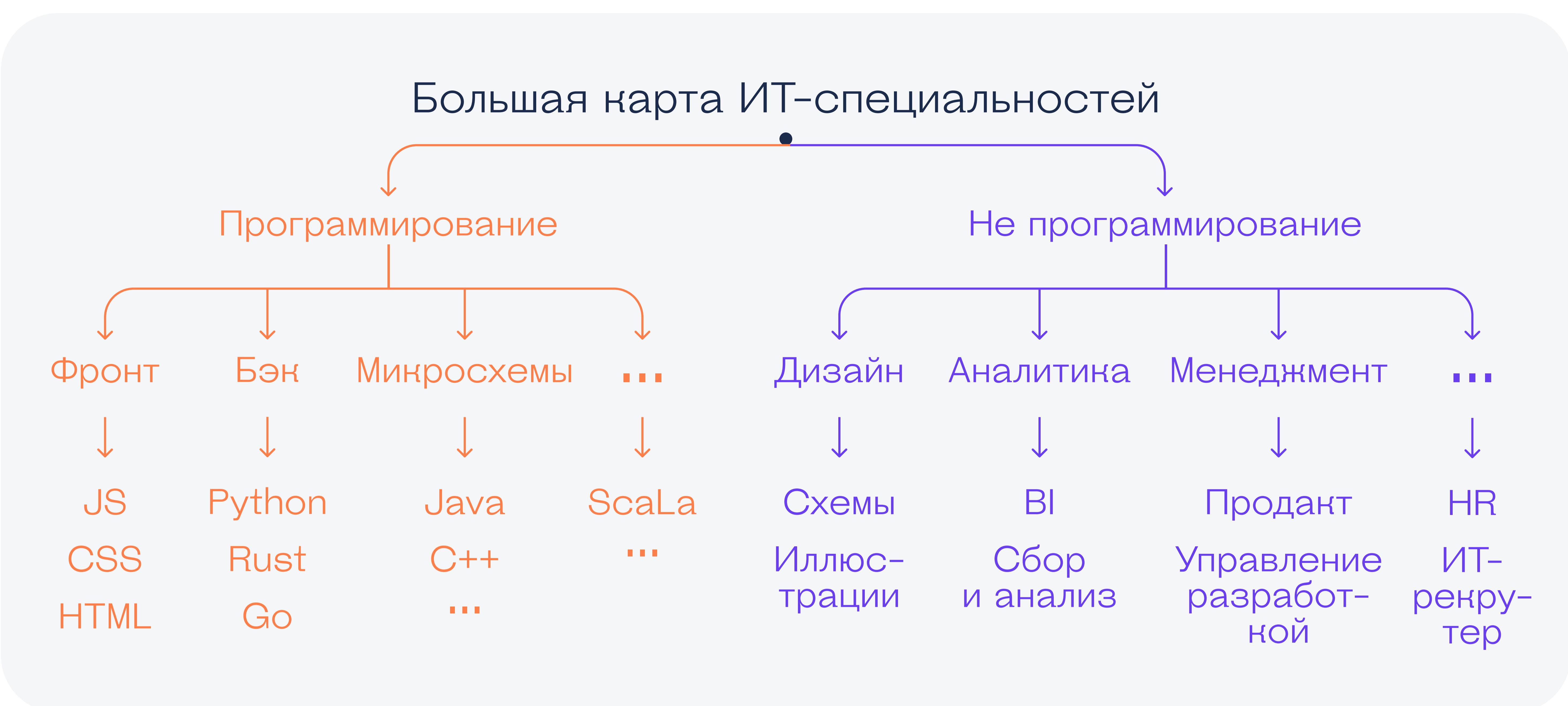
ИТ — это не только разработка

Есть стереотип, будто работа в сфере информационных технологий — это работа программиста. Это не так. Помимо разработки, есть многое другого:

- управление командой, то есть проект-менеджмент;
- дизайн интерфейса, графический дизайн, коммуникации;
- экономика и управление продуктом;
- маркетинг, реклама, продвижение;
- аналитика;
- дата-сайенс и нейросети — это тоже не совсем про разработку программ;
- работа рекрутера — то есть поиск и привлечение нужных людей в компанию.

Также есть много инженерно-технической работы, не связанной с разработкой напрямую: это работа с сетями, кибербезопасностью, поддержка инфраструктуры, автоматическое тестирование софта и многое другое.

Если вам не хочется быть конкретно программистом, ничто не мешает изучать другие сферы и тоже быть востребованным на рынке.



Чтобы погрузиться в эти направления, можно начать с курсов «Практикума». Или читайте дальше — разберемся более подробно

[Бесплатный курс «Анализ и оценка рынка»](#)

[Бесплатный курс «Наставник в ИТ»](#)

[Курс «IT-рекрутер»](#)

[Курс по управлению командой](#)

[Курс «Навыки аргументации для руководителей»](#)

[Курс «Инструменты начинающего руководителя»](#)

[Курс по управлению командой разработки](#)

[Курс «Продакт-менеджер»](#)

[Курс «Основы дизайна в Figma»](#)

[Курс «Коммерческий иллюстратор»](#)

[Курс «Бизнес-аналитик»](#)

Можно ли выучить что-то одно и быть спокойным за будущую карьеру?

Нет, нельзя. Тут две стороны вопроса.

С одной стороны, технологии быстро меняются. Три-четыре года назад все фанатели по Python и Django, это были самые горячие языки и технологии. Сейчас они заняли своё место, программисты теперь смотрят на другие игрушки.

Разработчики так же подвержены моде, как и обычные модники: если появляется какая-то новая красивая игрушка, программисты могут начать резко её использовать в работе, что приводит к росту спроса на рынке. Потом им эта игрушка надоедает, они перестают её использовать, спрос падает.

Плюс постоянно появляются более совершенные технологии. Разработчики хотят решить старые проблемы своих любимых языков; пишут для них расширения, новые фреймворки; даже придумывают новые языки.

Появляются принципиально новые задачи, под них нужны новые языки — типа Swift и Kotlin для мобильных приложений. И такая дребедень — целый день.

С другой стороны, вы сами как профессионал постоянно будете решать всё более сложные задачи. Вам для них нужны будут новые инструменты и фреймворки. Однажды вы сами захотите изучить новый язык, потому что старый вы перерастёте.

Так что нет, изучать новое вам придётся много раз. Другое дело, что после первого языка второй идёт проще, а проходить ради этого платные курсы совершенно не обязательно. Можно просто почитать документацию.

О двух подходах к разработке (и видах разработчиков)

Можно представить, что есть два вида программистов.

Первый вид — «Олимпиадники».

Это специалисты по алгоритмам, оптимизациям и низкоуровневым процессам, архитектуре и быстродействию системы. Это настоящие инженеры или выпускники факультетов прикладной математики.

Такие ребята могут написать алгоритм, запрограммировать робота, решить какую-то сложную вычислительную задачу на простом оборудовании. Их можно сравнить с автомеханиками, которые могут разобрать ваш автомобиль и собрать его заново.

Условно говоря, олимпиадник — это тот, кто написал алгоритм для построения маршрута для Яндекс-такси. Это нетривиальная задача — прочитайте нашу статью [о задаче коммивояжёра](#) и [про отжиг](#).

Второй вид — «Сборщики».

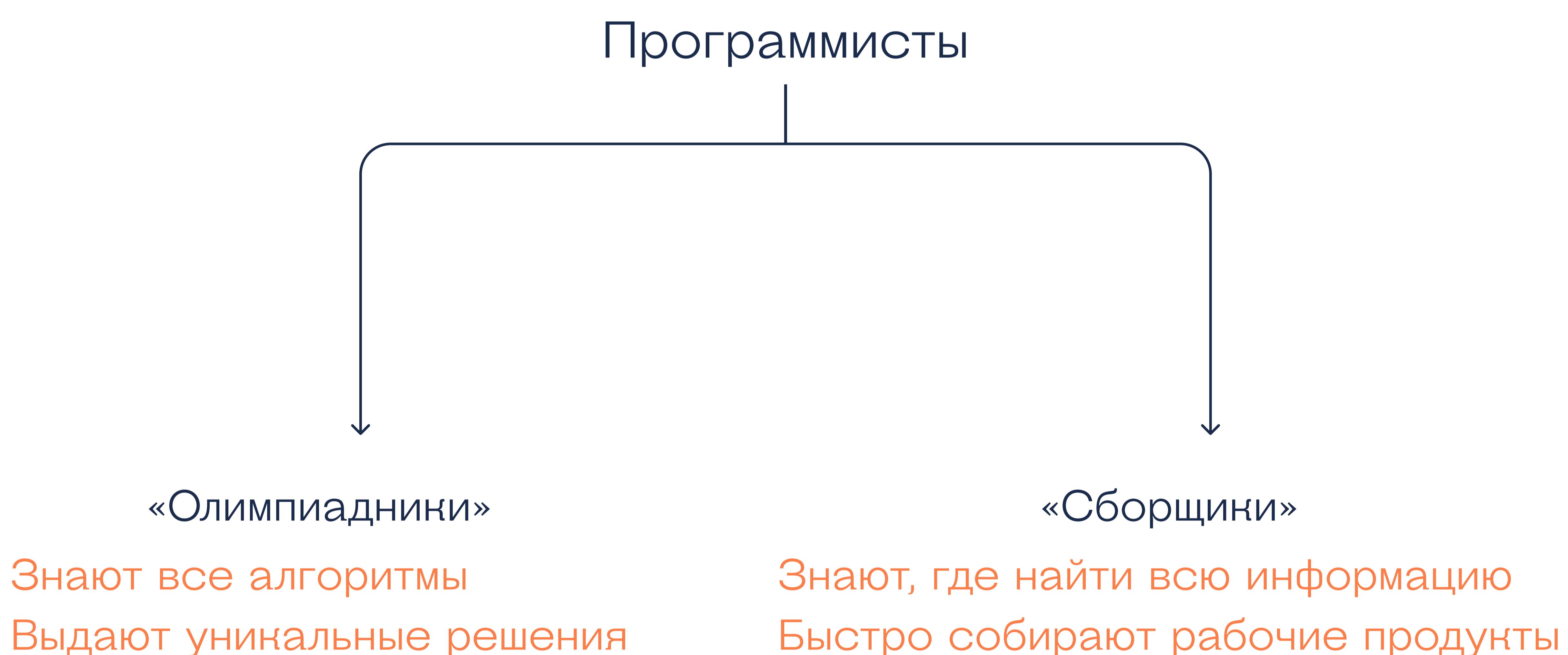
Это те, кто собирает программы из готовых библиотек и элементов интерфейса. Им нужно гораздо меньше инженерных знаний. Они меньше думают о быстродействии и архитектуре. Они просто делают и поддерживают софт — от веб-приложений для доставки пиццы до вашего онлайн-кинотеатра.

«Сборщику» не нужно знать досконально все нюансы сжатия потокового видео — он подключает для этого готовую библиотеку. «Сборщик» не писал алгоритм поиска маршрута, но он программировал экран для заказа такси.

Необязательно быть технарем и олимпиадником

Если вы не «олимпиадник» — это нормально. Вы можете быть очень успешным, востребованным и высокооплачиваемым «сборщиком». Такие «сборщики» нужны намного чаще, чем «олимпиадники», потому что 95% софта — это типовые штуки, которые до вас уже много раз придумали.

Не надо становиться «олимпиадником» и курить алгоритмы сортировки. Вы можете спокойно работать, используя уже написанную функцию `sort()`, если будете читать документацию. Можно выдохнуть.



Прочтите подробнее об этом разделении [в колонке Коли Митина](#).

На работу берут весёлых и организованных

Чем больше на рынке выбор айтишников, тем более важными становятся их софт-скилы — умение общаться, быть проактивным, не сидеть в своей скорлупе, не быть токсичным, конструктивно общаться с коллегами. Если 20 лет назад программисту могли простить его хамоватость, сейчас на его место легко найдут такого же, но умеющего общаться. Поэтому рекрутеры всё чаще говорят про важность навыков общения.

Поэтому не нужно думать, что если вы сегодня не знаете какой-то язык программирования, то вы не приспособлены в ИТ.

Наоборот: если вы умеете общаться, слушать других, конструктивно вести диалог, держать дела в порядке — у вас больше шансов выучиться и найти работу, чем у того, кто сегодня мастер программирования, но ужасный токсик и хаотик на работе.

[Бесплатный курс «Как составить резюме»](#)

Если интересны навыки организации работы, читайте журнал «Кинжал». Вот подборка навскидку:

[Как запускать проекты вовремя](#)

[7 медалей менеджерам](#)

[Как важно быть предсказуемым](#)

[Транзакционный налог](#)

[Как хорошо проводить встречи](#)

[О карьере менеджера](#)

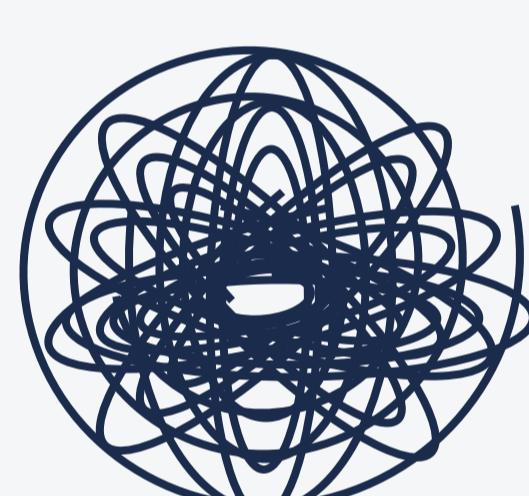
[О карьере аналитика](#)

Каждый следующий даётся легче

В программировании главное — понять принцип решения задач и стыковки готовых фрагментов кода, а не знание конкретных команд языка. Когда вы понимаете принципы разработки на примере одного языка, то же самое в другом языке становится в несколько раз проще. Вам не нужно тратить время на понимание принципов; останется только разобраться, как нужная вам вещь реализована в конкретном языке.

Так что всё не так страшно. Не думайте, что неверный выбор сейчас что-то предопределит и как-то повлияет на вашу судьбу. Тот факт, что вы открыли это руководство, уже говорит о том, что у вас есть все необходимые данные, чтобы работать в сфере ИТ.

Как выбор первого языка выглядит со стороны



Сложный выбор

Выбрали один язык

Это выбор на всю жизнь

Бэкенд

А остальным мы никогда не сможем заняться:

~~Мобильная разработка~~

~~Веб-разработка~~

~~Роботы~~

~~Биг-дата~~

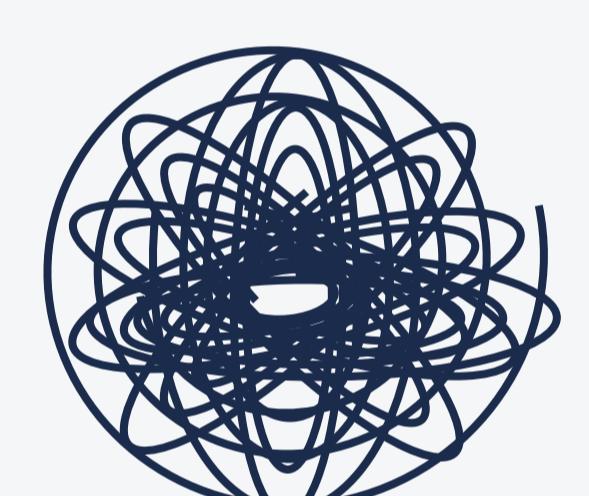
~~Аналитика~~

~~ЧПУ~~

~~Микросервисы~~

~~Десктопное ПО~~

Как это выглядит на самом деле



Выбрали простой язык

Бэкенд

А можно потом быстро выучить другое и заниматься, чем хочешь:

Мобильная разработка

Веб-разработка

Роботы

Биг-дата

Аналитика

ЧПУ

Микросервисы

Десктопное ПО

Алгоритмы везде одинаковые

Вот простая метафора:

Если вы научились водить автомобиль на механике, вы можете сесть плюс-минус за любой автомобиль и поехать.
Вам необязательно ездить именно за тем рулём, за которым вы учились в автошколе. Может быть, первое время вам будет непривычно в новой машине, но вы быстро освоитесь.

То же в разработке: принципы одинаковые, а нюансы языка можно освоить быстро. Со временем вы заметите, что циклы работают предсказуемо, а функции — идеальный инструмент для некоторых задач. Ещё через месяц вы понимаете, как собрать каркас программы, чтобы он не ломался. Ещё через месяц классы и методы уже гораздо понятнее, и часть вещей можно делать намного удобнее.

Шаг за шагом вы осваиваете свой первый язык программирования. Изучение идёт не слишком быстро, потому что много нового. А вот когда вы решили выучить другой язык, вам не нужно заново изучать как такое программирование.

Любой язык программирования

Переменная
Операторы действий
Логика алгоритмов
Условия

Циклы
Процедуры и функции
Классы, методы
и объекты

Для начала
можно и без них

Чем отличаются языки

Синтаксис
Команды
Структура программы

А это — частоты

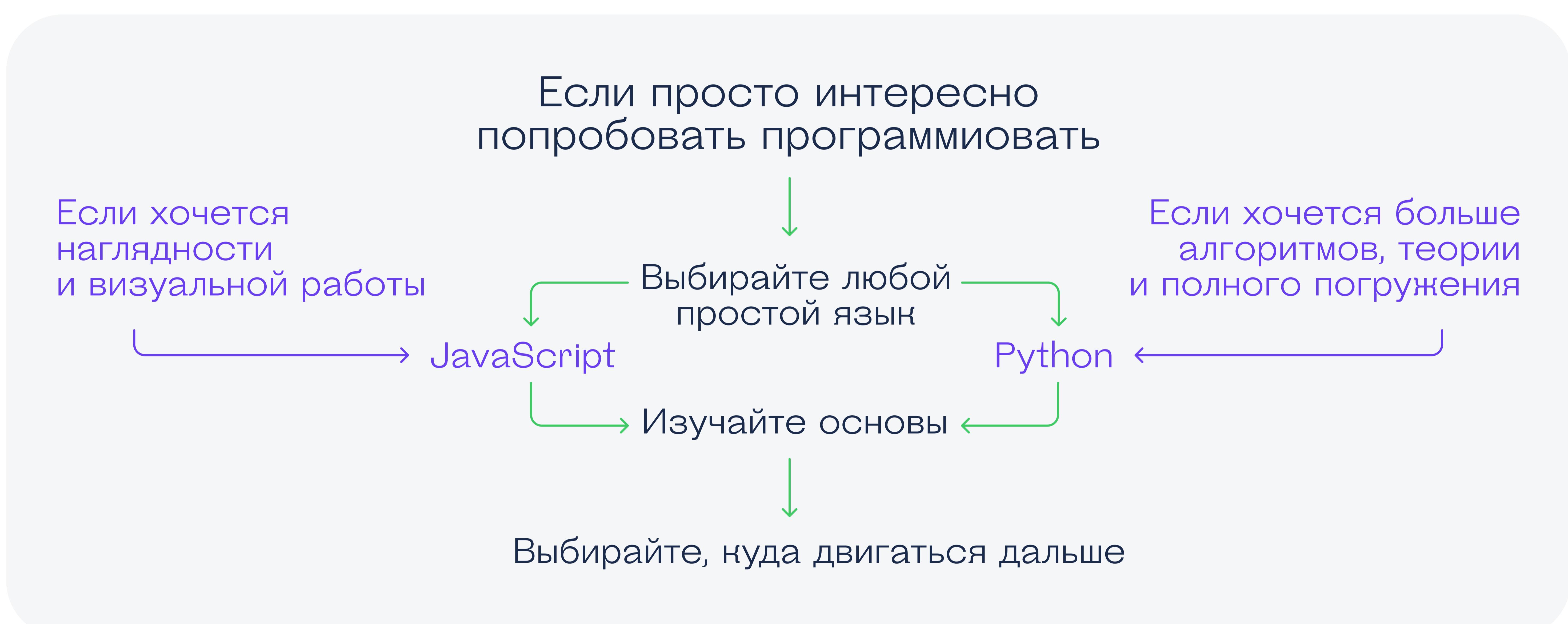
Это основа всего программирования

[Курс «Алгоритмы и структуры данных»](#)

Мне просто спросить

Иногда бывает так, что человек просто хочет понять, каково это — быть программистом и хочет ли он дальше этим заниматься. В этом случае нет смысла учить многопоточность в Python или углубляться в перегрузку операторов на C++.

Вместо этого лучше выбрать такой язык программирования, который будет легко освоить и у которого есть большая «группа поддержки» в интернете: форумы, учебные материалы и курсы. Раньше все начинали с Turbo Pascal 7.0, но сейчас этот язык морально и технически устарел, поэтому лучше выбрать JavaScript или Python. JavaScript — вообще самый дружелюбный и быстрый язык для входа.



Посмотрите на эти бесплатные курсы Практикума:

[Бесплатный курс «Как выбрать профессию в IT»](#)

[Бесплатный курс «Какую профессию в программировании выбрать»](#)

[Бесплатный тест профориентации](#)

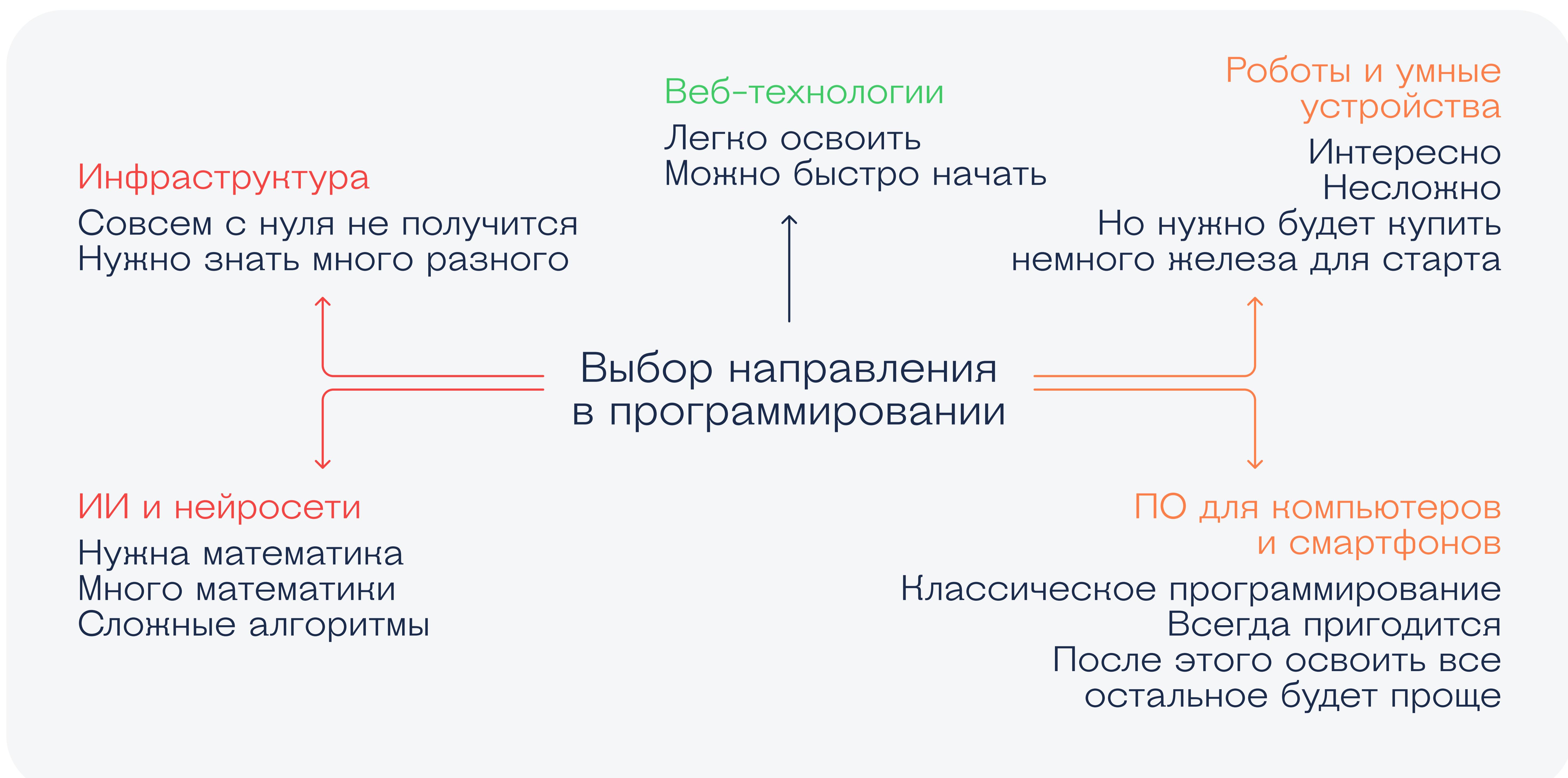
Веб, софт или робототехника

Если не вдаваться в детали, которые на старте не нужны, то всё ИТ и программирование можно разделить на 5 групп:

- 1 Веб и интернет-технологии — самое простое для входа.
- 2 Разработка программ для компьютеров и смартфонов — сложнее.
- 3 Программирование роботов и умных устройств — увлекательно и несложно, но на старте малоприменимо к работе.
- 4 ИИ и нейросети — сложно, если ковыряться; легко, если просто пользоваться.
- 5 Инфраструктура — сети, серверы, администрирование, тоже сложно.

Это основные направления. Кроме них есть более узкие области, например криптография. Но для старта достаточно ориентироваться в пяти основных, а дальше вы уже сами поймёте, где вам будет интереснее.

Две последние для новичка недоступны, поэтому разберём первые три.



Веб — самая простая область для старта

К вебу относится всё, что можно посмотреть или сделать в браузере. Про неё мы поговорим ниже более подробно, потому что это самый простой способ начать программировать и сразу получать результат.

Сила веба в том, что разрабатывать для него можно на любом компьютере, результат виден сразу. А исполнять написанное можно почти на любом устройстве: хоть на мощном компьютере, хоть на планшете, хоть на умном холодильнике.

Сфера применения веб-разработки огромна, порог входа низкий, кайф и радость.



Разработка программ для компьютеров и смартфонов

Когда говорят о программах, чаще всего имеют в виду именно это — классическую разработку программ. Всё, что мы запускаем на компьютере, написано ребятами из этой области.

Выбор конкретного языка зависит от направления. Например, разработчикам игр может понадобиться С и С++, в программах финансового анализа пригодится Haskell, а для мобильной разработки под iOS будет нужен Swift.

ПО для компьютеров и смартфонов

Графика

Плееры

Офис

Мессенджеры

Антивирус

Таск-менеджеры

Бухгалтерия

Игры

Клиенты соцсетей

Почта

Браузер

Монтаж видео

Классическая разработка ПО — это серьезная большая карьера. Если чувствуете, что это прямо ваше, рассмотрите вот эти курсы Практикума:

[Курс «Android-разработчик»](#)

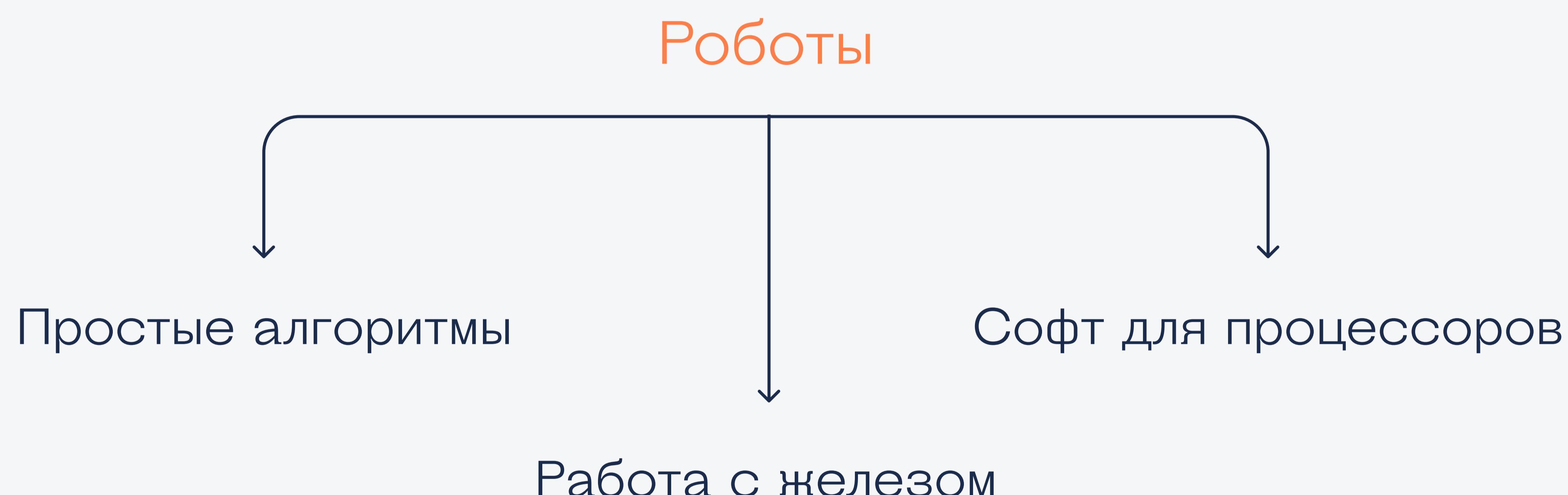
[Курс «iOS-разработчик»](#)

Программирование роботов и умных устройств

Многие программисты говорят, что высший пилотаж — писать специальный код под конкретную микросхему или устройство. В этом случае вы получаете полный контроль над железом и можете управлять чем угодно: микроволновками, ракетами, фотоаппаратами, сигнализацией, умным домом, электроникой в автомобиле и даже ядерными реакторами.

Быстрого старта тут не получится — нужно знать устройство и архитектуру микросхем, низкоуровневое программирование и управлять памятью напрямую. Но есть обходной путь: начать с робототехники и программирования для Arduino и Raspberry Pi — это специальные платы для начинающих.

В мире есть целое сообщество хоббистов, которые пишут программы для Arduino и Raspberry Pi и собирают на их основе интересные проекты. В России самый простой вход в Arduino делает компания «Амперка» — у них есть наборы для старта работы с Arduino и всякими полезными радиодеталями.



Вот подборочка, что можно делать на умных устройствах:

[На Arduino](#)

[На Raspberry Pi](#)

Веб: бэк и фронт

Из всех видов программирования проще всего начать с веба, потому что он быстрее всего даёт практически применимый результат.

Веб — это всё, что вы видите в интернете: сайты, приложения в браузере и бэкенд, который это обслуживает. Здесь используют:

- Язык разметки HTML — чтобы структурировать страницы в браузере.
- Язык оформления CSS — чтобы страница или интерфейс выглядели как надо.
- Язык препроцессоров CSS — когда нужно очень много CSS-кода, правил и стилей.
- Язык скриптов JavaScript — чтобы интерфейс или приложение вели себя на клиенте как надо.
- Язык и форматирование JS-фреймворков типа React и Angular — чтобы рисовать и программировать поведение сложных веб-приложений.
- Языки бэкенда: PHP, Python, Ruby (из популярных) — чтобы обслуживать приложения и хранить пользовательские данные.
- Языки бэкендовых фреймворков, например Django и Rails.
- Язык запросов в базу данных: чаще всего MySQL, но есть и другие.

Все эти инструменты нужны для разных задач. Нельзя сказать, что одни лучше других.

В веб-разработке есть два направления — фронтенд и бэкенд. Фронтенд — это всё, что видит пользователь. Бэкенд — всё, что обрабатывает информацию пользователя и скрыто от его глаз.

Фронтенд-разработчик делает следующее:

- собирает сайт по макету дизайнера;
- использует для этого HTML, CSS, JavaScript и несколько других языков;
- понимает процессы, которые происходят во время создания сайта;
- знает, как опубликовать сайт в Сети так, чтобы он выглядел одинаково на всех устройствах;
- умеет работать с Git;
- использует Webpack для сборки проекта и вообще оперирует препроцессорами.

Звучит сложно, но вот основное: фронтенд берёт макет будущего сайта (картинку) и превращает его в код, который можно отправить клиенту. При необходимости он программирует интерактивные элементы и анимацию, которые будут обрабатываться на клиенте.



Часто фронтендов путают с верстальщиками, но на самом деле верстальщик — это специалист узкого профиля (вёрстка по макету). А фронт кроме этого может и слайдер прикрутить, и шаблон в CMS поправить, и запрограммировать нестандартное поведение картинки при нажатии, и написать скрипт для проверки правильности заполнения данных на сайте.

Бэкенд-разработчики пишут программы для серверов, которые формируют нужные нам страницы и отправляют их нам на компьютер. Например:

- они продумывают архитектуру сайта и связи между его компонентами;
- настраивают базы данных, где хранится вся информация;
- делают так, чтобы сайт мог получать и отправлять информацию в эту базу;
- пишут движок сайта — ту программу, которая формирует страницы;
- если движок уже готовый — допиливают его;
- оптимизируют движок, чтобы сайт работал как можно быстрее и стабильнее;
- следят за безопасностью сайта, чтобы злоумышленник не смог украсть или подделать данные;
- иногда настраивают сами серверы — Apache или Nginx.

Главный инструмент бэкенд-разработчика — язык программирования. Здесь у бэкенда два главных языка:

- PHP, на котором сделаны почти все современные веб-движки;
- Python, на который переходит весь просвещённый мир.

[Курс «Веб-разработчик»](#)

[Курс «Фронтенд-разработчик буткемп»](#)

Лучшие языки для старта

Теперь про сами языки для старта. Мы отобрали 5 лучших языков, которые максимально помогут новичку на старте. Главная их особенность — их можно освоить относительно быстро, а потом погружаться в детали, если будет интересно развиваться дальше.

Если вы выше прочитали про фронт, бэк или роботов и поняли, что нашли там свой идеальный язык для старта, — это отлично. Если нет — вот из чего можно выбрать.

Содержание:

Python — простой, дружелюбный, нужно ставить софт для работы

19

JavaScript — самый дружелюбный, софт не нужен, работает в браузере

21

Java — для тех, у кого самые серьезные намерения

23

А также для порядка:

Rust — чтобы быть модным

26

С и С++ — чтобы программисты не сказали, что мы забыли в нашем гайде их главный язык

28

Python

Python

Простота



Наглядность



Зарплата



Погружение
в алгоритмы



Универсальность



Легко перейти
на другой язык



Первое, что отмечает большинство разработчиков, когда начинаются разговоры про Python, — безупречный и аккуратный код. Чтобы оформить любой логический блок, например содержимое функции, просто используют отступы и пробелы. Не нужно проверять, не потерялась ли по пути лишняя скобка — всё видно сразу. Одна команда занимает одну строку, поэтому разделители команд тоже не нужны.

В какой бы области вы ни делали проект, скорее всего, для этого уже есть готовая Python-библиотека. Обработка изображений, математика, распознавание речи — для всего есть инструменты. Чтобы вы понимали, насколько это универсальный язык, посмотрите, для чего используют Python-библиотеки:

- NumPy — машинное обучение и искусственный интеллект;
- Django и Flask — веб-разработка и веб-приложения (например, Pinterest, YouTube и Instagram написаны на Django);
- SQLAlchemy — базы данных и обработка больших объёмов информации;
- Cocos2d — мобильные и браузерные игры;

- Tornado — приложения, которые требуют высокой производительности и должны работать одновременно с сотней тысяч пользователей;
- Bubot — программирование робототехники, как вариант — использование на Raspberry Pi.

Python можно запустить практически на любой платформе — от КПК до серверов — и на любой операционной системе.

Правда, есть ограничения. Как только платформа полностью устаревает и перестаёт использоваться (например, Windows 95), прекращается её поддержка в новых версиях языка.

Более того, существует специальная версия Python для виртуальной машины Java — Jython. С её помощью можно выполнять код на любой системе, которая поддерживает Java (то есть на чём угодно умнее утюга, но и на некоторых утюгах тоже).

[Бесплатный курс «Основы Python-разработки»](#)

[Курс «Python-разработчик»](#)

[Курс «Python-разработчик плюс»](#)

[Курс «Специалист по Data Science»](#)

[Курс «Специалист по Data Science плюс»](#)

[Курс «Специалист по Data Science буткемп»](#)

[Курс «Аналитик данных»](#)

[Курс «Аналитик данных плюс»](#)

[Курс «Аналитик данных буткемп»](#)

[Курс «Тестирование веб-приложений на Python»](#)

JavaScript

JavaScript

Простота



Наглядность



Зарплата



Погружение
в алгоритмы



Универсальность



Легко перейти
на другой язык



JavaScript — это язык для управления веб-страницами в браузере. Он исполняется только в браузере, может красить элементы на странице в разные цвета, рисовать интерфейсы, пульсироваться данными, но всё — в браузере. На JavaScript можно писать и приложения и даже десктопные программы. Но это требует огромного внешнего обвеса, мы не рекомендуем.

Кто бы что ни говорил про JavaScript, но он был и остаётся идеальным языком для начала программирования. Смотрите сами:

- для работы нужен только браузер или его движок типа V8;
- можно освоить за неделю;
- это самый используемый язык, по статистике на StackOverflow;
- JavaScript можно подключить к любому сайту в интернете;
- и главное — огромное сообщество, где можно найти решение любой проблемы.

JavaScript — это язык программирования, с которым все сталкиваются каждый день:

- в интернет-магазине, когда вы кликаете «Добавить в корзину» и счётчик товаров сразу обновляется;

- когда вы при регистрации вводите неправильный имейл и форма сразу подсвечивает вам, что он неверный;
- в социальных сетях в браузере, когда новое сообщение приходит без перезагрузки страницы;
- там же, когда вы можете перейти из личных сообщений в новости, продолжая слушать любимую музыку;
- в любой браузерной игре;
- когда на сайтах есть какая-то анимация;
- когда сайт собирает данные для статистики;
- когда посреди чтения статьи на вас бросается бесяющая нападайка.

JavaScript — это язык, с помощью которого веб-страницы оживают, в них динамически подгружаются данные, появляются всплывающие окна, выпадающие элементы и миллион других вещей.

А главное — это всё ещё самый популярный язык веб-разработки. Главное его преимущество — веб-программисты нужны везде, даже там, где делают мобильные приложения и игры. Потому что существенная часть игр и мобильных приложений сделана на технологиях веба.

Если вам нужно просто с чего-то начать, чтобы понять вообще, стоит ли идти дальше в программирование, — начните с JavaScript.

[Курс «Веб-разработчик»](#)

[Курс «Веб-разработчик плюс»](#)

[Курс «Фронтенд-разработчик буткемп»](#)

Java

Java

Простота



Наглядность



Зарплата



Погружение
в алгоритмы



Универсальность



Легко перейти
на другой язык



Java — это серьёзный объектно ориентированный язык, на котором пишут серверный софт, программы для компьютеров и мобильные приложения. Он интересен тем, что один и тот же код можно скомпилировать под множество разных платформ. Java — один из основных языков для разработки под Android.

А ещё это хороший язык для начинающих и третий по популярности язык программирования в мире. В Сети есть тысячи сайтов, которые рассказывают об этом языке, помогают разобраться в коде или содержат готовые решения.

Теперь коротко про язык:

Многоплатформенность. Работает на огромном количестве операционок и железа.

ООП. Для тех, кто любит чёткие структуры и разграничение данных.

Большое сообщество и много уже написанного кода.
Нет проблем с работой и с готовыми решениями.

Сила Java — в виртуальной машине JVM (Java Virtual Machine). Это такая программа, которая переводит Java-код, понятный человеку, в код, понятный процессору. Чтобы код стал универсальным, разработчики сделали виртуальные машины для каждой операционной системы и процессоров. Эти машины учитывают все особенности архитектуры своей платформы и знают, как обработать любую Java-команду. Это значит, что один и тот же Java-код можно запустить и на телефоне, и на компьютере, и где угодно ещё.

Поэтому запускать код на Java можно везде, для чего уже была разработана JVM, то есть виртуальная машина Java. Например:

- ультрамощные серверы,
- компьютеры,
- смартфоны,
- кнопочные телефоны,
- роботы и микроконтроллеры типа Arduino, Raspberry Pi и многих других,
- фитнес-браслеты,
- GPS-трекеры,
- умные часы,
- умные телевизоры, холодильники, микроволновки, чайники и прочие домашние гаджеты,
- смарт-карты для доступа в помещения.

Представьте такую ситуацию: вы написали Java-программу, которая следит за свободным местом на диске, и как только его становится меньше 20% — выводит сообщение, мол, удали ненужные файлы. Теперь вы можете запустить эту программу везде, где есть JVM. Она сама разберётся с кодом, поймёт, для какого процессора она это делает, и запустит вашу программу.

В результате один и тот же код будет следить за свободным местом на компьютере, кнопочном телефоне, смартфоне, планшете, умной микроволновке или домашней сигнализации.

Ещё Java — стандарт в корпоративном программировании. Большим компаниям в программах нужна надёжность, стабильность работы и возможность поддерживать их долгое время. Сочетание ООП, управления памятью и независимости от архитектуры делает Java идеальным решением для этого.

[Курс «Java-разработчик»](#)

[Курс «Автоматизатор тестирования на Java»](#)

Rust

Rust

Простота



Наглядность



Зарплата



Погружение
в алгоритмы



Универсальность



Легко перейти
на другой язык



Rust позиционируется как компилируемый системный мультипарадигмальный язык высокого уровня. Сейчас поясним, что это значит.

Компилируемый язык означает, что готовая программа — это отдельный файл, который можно запустить на любом компьютере с нужной операционной системой. Для запуска не нужно устанавливать среду разработки и компилятор, достаточно, чтобы скомпилированная версия подходила к вашему компьютеру.

Системный — это когда на языке пишут программы для работы системы в целом. Это могут быть операционные системы, драйверы и служебные утилиты. Обычные программы тоже можно писать на Rust — от калькулятора до системы управления базами данных. Системный язык позволяет писать очень быстрые программы, которые используют все возможности железа.

Мультипарадигмальный значит, что в языке сочетаются несколько парадигм программирования. В случае Rust это ООП, процедурное и функциональное программирование. Причём ООП в Rust пришло из C++, а функциональное — из Haskell. Программист может сам выбирать, в каком стиле он будет писать код, или совмещать разные подходы в разных элементах программы.

Несмотря на синтаксис, похожий на C, главную особенность программ на Rust разработчики взяли из Haskell, и звучит она так:

Если программа на Rust скомпилировалась и не упала во время запуска, то она будет работать до тех пор, пока вы сами её не остановите.

Когда язык совмещает в себе несколько разных подходов из других языков, он получает большинство преимуществ каждого из них:

- высокая скорость работы программ;
- возможность написать код в ООП-стиле: с классами и объектами (но есть ограничения);
- стабильность в работе и при компиляции;
- компилятор сам предлагает варианты исправления ошибок в коде;
- кросс-платформенный компилятор;
- поддержка многопоточности;
- поддержка «небезопасных» блоков для прямой работы с памятью;
- можно вставлять код на C и C++.

С и С++

C/C++

Простота



Наглядность



Зарплата



Погружение
в алгоритмы



Универсальность



Легко перейти
на другой язык



Мы долго сомневались, включать эти языки в список
для начинающих или нет, и вот почему:

- С одной стороны, это отличные языки программирования с фундаментальной базой и всеми нужными понятиями. Если освоить один из них, то разобраться потом с остальными будет в сто раз проще.
- С другой — придётся разобраться не только в алгоритмах, но и в организации памяти, внутреннем устройстве стека и прочих сложных вещах. Это всё по силам и новичку, но времени надо много.

С — классический язык разработки системного ПО и любого софта для микропроцессоров. На нём написаны Linux, большая часть Windows и MacOS. Если взять любой современный носимый гаджет или электронное устройство, в большинстве случаев они работают тоже под управлением программы на С. В мире огромное количество кода, который написан на С (и ещё столько же будет написано), поэтому проблем с работой у С-программистов не предвидится.

C++ — выбор тех, кому одновременно нужна вся мощь C и гибкость объектно ориентированного программирования. Counter-Strike, StarCraft и World of Warcraft написаны на C++, а это значит, что можно сочетать производительность C и современные технологии. Часть движка Unity тоже написана на C++, чтобы получить прямой доступ к памяти и ресурсам системы.

Но на самом деле нет такой уж большой разницы между C и C++. Это одинаково мощные и быстрые языки, просто у них разная область применения и стиль программирования.

Если вам важна скорость, производительность и относительная простота языка — смотрите на C, там это всё есть. С другой стороны, из-за простоты в нём многие вещи придётся делать вручную — обрабатывать исключения, следить за сроком жизни переменных и структур или писать дополнительный код.

[Курс «Разработчик C++»](#)

[Курс «C++ для бэкенда»](#)

Ответы на вопросы

Можно ли начать программировать вообще с нуля?

Все люди, которые пришли в программирование, когда-то начинали с нуля. Правильнее спросить, через сколько времени можно развить достаточный навык, чтобы делать что-то полезное для рынка. И здесь всё зависит от должности и задач.

Работать с фронтендом и писать простые браузерные приложения можно научиться за два-три месяца плотных занятий. За это время вряд ли получится освоить профессию на уровне старшего разработчика, но для старта хватит.

Бэкенд или мобильные приложения, по сути, ненамного сложнее, но чтобы делать коммерчески полезный продукт, потребуется учиться не меньше 5–6 месяцев в интенсивном режиме. Сначала алгоритмы и теория компьютерных вычислений, потом язык, потом его прикладное применение. И это только для начала.

Обязательно ли быть технарём или заканчивать технический вуз?

Нет, вот это точно не обязательно. Любой человек с высшим или среднеспециальным образованием может освоить всё, что нужно для программирования на выбранном языке.

Определяющую роль здесь играет не надпись в дипломе и не ярлык «технарь», а внутреннее желание разбираться в предмете. В ИТ есть области, которые не преподают в вузах, но которые хорошо задокументированы в интернете — достаточно просто тратить время на то, чтобы читать и вникать.

Нужно ли осваивать язык в совершенстве, чтобы начать работать?

Нет! Наоборот: гораздо полезнее начать стажёром в ИТ-компании, зная только самые основы, и потом развиваться исходя из поставленных задач.

Например, вы можете начать работать джуниор-разработчиком под руководством более опытного наставника. Вы сразу будете решать рабочие задачи и развиваться исходя из реальных потребностей, а не учить языки и технологии «в пустоту».

Мне 30, 40, 50 лет — уже поздно?

Зависит от того, готовы ли вы вникать в новую тему. Если у вас есть полчаса в день на изучение языка программирования, прогресс будет медленным. Если вы готовы уделять этому много времени, то всё получится быстрее. Но каких-то возрастных ограничений нет: писать код и управлять продуктами можно в любом возрасте.

У людей постарше есть преимущество в том, что они могут лучше работать с коллективом и иметь больший жизненный опыт. Там, где молодой лидер вспылит и встанет в позу, опытный товарищ найдёт правильные слова и восстановит хорошие отношения с командой.

Я женщина — мне подойдёт?

В смысле? В ИТ полно женщин. Какая разница, чьи пальцы нажимают на клавиши и пишут код?

Нейроночки порешают

С развитием нейросетей всё меньше спроса будет на тех, кто владеет алгоритмами и сложными типами данных — эти знания как раз хорошо автоматизируются в нейросетях. Спрос будет на тех, кто может послушать менеджера, понять задачи бизнеса, поговорить с коллегами и грамотно сформулировать задачу машине.

Ваш предыдущий жизненный опыт — это то, что поможет вам стать востребованным разработчиком. Не знание Java или Python, а знание людей, деликатность, предпримчивость и внимание. Для всего остального уже сейчас есть ChatGPT, а дальше его будет ещё больше.

Поэтому не бойтесь сейчас выбрать неправильный язык: когда вы будете свободно им владеть, уже будут технологии перевода между языками. Не бойтесь сейчас выбрать какой-то фреймворк, который устареет через два года. К моменту, как вы будете работать, вы сможете освоить десять других фреймворков.

Цена ошибки сейчас небольшая, и нейросети делают ее еще меньше. Всё получится.

Куда идти учиться?

О, мы как раз об этом сделали подборку. Но если вкратце: если хотите учиться при поддержке опытных наставников (живых!) — приходите в Яндекс.Практикум. Кроме наставников там есть тренажёры, домашка, теория, практика, ещё теория, ещё практика, потом ещё, потом курсовая, потом защита, потом ты такой сидишь и выбираешь, в какой компании работать.

Или можно учиться самостоятельно по урокам и инструкциям из интернета — но там без наставников, своими силами.

Что дальше

Нет одного универсального языка программирования на все случаи жизни.

Если вы знаете один язык, другие будет освоить намного легче.

Чем больше языков вы знаете, тем шире ваш кругозор и тем ценнее вы как специалист.

Необязательно все дальнейшие задачи решать на своём первом языке программирования — всегда можно выучить что-то новое.

А главное — с чего бы вы ни начали, знайте: у вас всё получится.
Мы верим в вас.

На дорожку

Пройдите тест Практикума, чтобы увидеть, в какой профессии вам будет комфортнее всего заниматься программированием.

Посмотрите для примера наш тест о работе в офисе ИТ-компании. Вас удивит, насколько всё бывает неоднозначно.