КОД
журнал Яндекс Практикума

Гайд

Материалы

40 вопросов про ИТ от гуманитариев

Простые, сложные, острые, странные и необычные вопросы — ответили на всё, что интересует новичков

Читайте дальше

Мы собрали все самые острые вопросы про старт в ИТ и делимся с вами ответами. Читать можно в любом порядке. Чтобы погрузиться в тему, к каждому ответу мы добавили ссылки на статьи «Кода» — так будет проще разобраться с каждым вопросом.

Содержание:

Карьерные перспективы в сфере ИТ с учётом нейросетей и ChatGPT

Вытеснят ли нейросети программистов?

Нейросети не вытеснят программистов, а сделают их работу намного быстрее.

Например, раньше разработчик решал некоторую задачу за час. Сейчас благодаря ChatGPT или Copilot он будет решать её за 15–20 минут, поэтому работа будет быстрее, а требования к разработчикам — выше.

На первый план у разработчиков выходит не знание конкретных языков программирования и не владение конкретными фреймворками, а навыки командной работы, широкий технический кругозор и способность оценивать чужой код.
То есть важным будет не написать код на конкретном фреймворке, а понимать, какой тебе нужен фреймворк под задачу и какой код на нём будет оптимальным.

Ещё подсказки и тенденции — в нашей статье про soft skills

Что изучать сейчас, чтобы быть востребованным на ИТ-рынке в будущем?

Вопрос принципиально неверный, потому что технологический ландшафт очень быстро меняется. Нужно перестроить мозги.

Сейчас нужно учиться учиться — то есть приучить себя изучать новые технологии и инструменты по мере их появления. Бессмысленно учить один язык и потом использовать его всю жизнь. Нужно приучать себя изучать то, что нужно для конкретной задачи в конкретный момент.

Например, сегодня много бэкенда пишут на Python, в том числе в компании, где вы хотите работать. Через два-три года могут придумать какой-то новый стек технологий для бэкенда или вообще сменится парадигма серверно-клиентских отношений. Когда это произойдёт, нужно будет изучить новое и начать применять.

Нет ни одной технологии, которая будет одинаково актуальной сегодня и через пять лет. Но есть вы, и вы себя не подведёте.

Бесплатный курс «Как выбрать профессию в IT»

Какие специальности станут менее актуальны из-за нейросетей?

Наоборот, размышляйте: какие навыки станут более важными из-за нейросетей? А навыки такие:

- Работать с людьми, в команде, слышать клиента и понимать его запросы. Уходит эпоха угрюмых разработчиков в стиле «злой гений» — сейчас быть нормальным человеком не менее важно, чем хорошо кодить.
- Формулировать проблемы, которые нужно решить кодом; формулировать задачи для нейросетей.
  Мыслить системно, изучать бизнес клиента.
- Оценивать чужой код, собирать решения из готовых кусков кода. Писать самому нужно всё меньше, но код-то нужен.
- Работать дисциплинированно, чётко, предсказуемо и надёжно. Раньше можно было всех подводить, но так как только вы в компании могли написать код, все с этим мирились. Сейчас вместо вас код может написать технически подкованный менеджер, поэтому вести себя некомандно уже нельзя.
- Широкий кругозор в вопросах существующих инструментов и технологий. Вот есть 50 фреймворков, вы знаете только один. А что, если соседний фреймворк помогает решить эту задачу намного лучше? Попробовать 50 фреймворков сейчас важнее, чем преисполниться в каком-то одном.

Благодаря нейросетям у вас будет меньше ручного труда и больше — аналитического. Грязную работу за вас сделают алгоритмы.

Ещё подсказки и тенденции — в нашей статье про soft skills

Уже поздно врываться в ИТ?

Сейчас информационные технологии — такая же часть экономического ландшафта, как строительство, образование, сельское хозяйство, логистика или торговля. Эти сферы автоматизируются, но от этого люди там нужны не меньше, чем раньше.

Нет такого, что сейчас поздно врываться в инженерное дело, образование или медицину. Всё, эта сфера с нами надолго. Чем дальше — тем больше нужны будут информационные технологии в разных сферах народного хозяйства.

Но важно понимать вот что: никакая сфера сама по себе не является бесконечным генератором лёгких денег. Если вы хотите работать в ИТ только в режиме «300 тысяч в месяц за то, что я знаю Python», то в принципе не стоит сюда идти. В ИТ можно заработать, но это уже давно не та сфера, где в тебя кидаются деньгами просто за то, что ты такой хороший.

Читайте:

- Что там с работой в ИТ (2023 год)
- Бесплатный тест для профориентации

Про выбор языка и направления

Какой язык выбрать новичку?

Смотря для чего вам нужно программирование прямо сейчас.

- Если вам нужно писать приложения быстро и скорость работы этих приложений не критична, выберите Python.
- Если вам интересно не программирование, а нейросети и бигдата — всё равно Python.
- Если вы хотите писать серверное ПО или приложения для Windows — можно выбрать C# или Java. Java популярнее, но там сложно.
- Если вы фанат экосистемы Apple, то вам Тим Кук велел писать на Swift.
- Если вы хотите создавать веб-сайты и веб-приложения, JavaScript — ваш язык. Там же нужно изучить HTML и CSS, потому что чистый JavaScript нужен редко.
- Если вы хотите делать игры для ПК и консолей, смотрите на C++, но там придётся изучить объектно-ориентированное программирование.
- А если вам хочется делать игры конкретно на Unity — C#.
- Про мобильное программирование: Java и Kotlin популярны для Android, а Swift — для iOS.
- Если хотите делать приложения и сайты, но не хотите программировать — изучите наш рассказ о zero-code и low-code.

Но вообще понятие «выбор языка» неверное.
Смотрите, как на самом деле:

1. Вы выбираете какой-нибудь язык — хоть Python, хоть C, хоть JavaScript.
2. Вы учитесь решать на нём нужные вам задачи: математику, вывод данных, обработку данных, работу с текстом, объектами и т. д.
3. Вы становитесь компетентным в вопросах решения реальных задач с помощью программирования как такового. Не на конкретном языке, а именно с помощью программирования в принципе.
4. У вас появляются реальные рабочие задачи.
5. Вы быстро выучиваете то, чего вам не хватает для решения этих задач.

Python, C, PHP и JavaScript — это не разные планеты. Это скорее разные наречия одного общего «языка программирования».
Если вы умеете решать задачи с помощью программирования, вопрос языка для вас не будет стоять так остро. Если вы понимаете Python, вы сможете разобраться в коде на C.

А с помощью нейросетей типа ChatGPT можно вообще переводить с одного языка на другой.

Читайте ещё по теме:

Как начать программировать с нуля

Почему в школе до сих пор изучают Pascal

«Успешный программист не привязывается к языку. Он просто умеет программировать»

Стоит ли учить несколько языков программирования одновременно?

Если вы только начинаете изучать программирование — точно нет.

Изучать языки программирования нужно по мере появления задач. Просто учить языки, потому что они фигурируют в каком-то списке, — нет смысла.

Читайте ещё по теме:

> Какой язык программирования выбрать для старта

Хочу изучать C++, но слышал, что код всегда получается плохим, особенно у новичков. Это правда?

Никакой язык не виноват, что на нём не получается хороший код (кроме языка brainfuck, он виноват). Язык — это просто инструкции компьютеру.

Есть люди, которым нравится чистота Python или возможности C++. Это не значит, что Python или C++ хорошие. Это значит, что есть люди, которые хорошо владеют каким-то из языков, поэтому топят за свою команду.

Вам достаточно знать, что какой-то язык подходит для какой-то задачи. Когда вам на работе попадётся такая задача, вы напишете код на нужном языке. Плохой код или хороший — будет решать ваш сеньор. В отрыве от задачи и коллектива этот вопрос лишён смысла.

Читайте ещё по теме:

> Вам мало языка C? Попробуйте C++

> Языку C уже почти 50 лет, но он всё равно крут

Хочу сразу выбрать мощный язык программирования. Что лучше в этом плане, C или C++?

А вам для чего? В чём измеряете мощность? Вот инструкция, как выбрать самый подходящий:

1. Открываете вакансии разработчиков в интересующем вас городе.
2. Находите вакансии, которые вам больше всего нравятся.
3. Те языки, которые там нужны, — для вас самые мощные.

Все споры о возможностях и мощности чаще всего нужно читать как «чья команда лучше?». Просто дядьки, которые 30 лет пишут на C, не хотят дружить с дядьками, которые 15 лет пишут на C++.

Читайте ещё по теме:

Чем отличается C от C++

Хочу выучить все современные языки программирования и фреймворки, чтобы на них можно было сделать любой проект. Звучит логично?

Нет.

Пока вы будете учить то, что есть сейчас, — появится что-то новое. Язык, может быть, не появится, но пяток модных фреймворков точно вылупятся.

Долгое изучение каких-либо технологий в отрыве от задачи лишено смысла. Вы тут же всё забудете.

Ни для какой жизненной задачи не нужно знать все языки и фреймворки. Нет ни одной вакансии в мире, где нужно знать всё.

Обычно желание выучить всё — это признак прокрастинации: когда вместо того чтобы нырять в реальную работу, человек сидит на берегу и учит очередной фреймворк. Так делают немногие, но если вас вдруг в это затянуло — ныряйте скорее, не сидите на берегу.

Читайте ещё по теме:

> Начинающим программистам: что такое фреймворки и библиотеки

Научиться:

> Бесплатный курс «Какую профессию в программировании выбрать»

Веб-программирование — это самый простой способ научиться программировать?

Веб-программирование простое только в том, что для начала работы в нем у вас уже всё есть на компьютере: браузер и текстовый редактор. Не нужно устанавливать среду разработки и покупать быстрый комп для компиляции кода.

А обратная сторона в том, что веб изначально не предназначался для разработки полноценных приложений. Там есть огромная прослойка из костылей, чтобы рисовать в вебе полноценные интерфейсы, верстать окна, всплывашки и т. д. И вместо того чтобы программировать алгоритмы, вы будете ломать голову, почему всплывающее окно разорвало страницу и улетело на километр вниз.

Веб-программирование нужно только для одного — делать веб-приложения.

Читайте ещё по теме:

- Три причины начать с веб-разработки (для тех, кто сомневается)
- Что такое HTML (и почему это важно)

Научиться:

- Курс «Веб-разработчик»

Слышал про объектно-ориентированное программирование, что оно сейчас не в моде; а те, кто его используют, застряли в прошлом веке. Это так?

Нет, с ООП всё в порядке.

Объектно-ориентированное программирование применяется уже более 30 лет и проработает ещё как минимум столько же. Дело в не том, сколько лет технологии, а в том, как она работает и как помогает решать задачи. Например, почти все современные компьютерные игры невозможны без объектно-ориентированного подхода.

Для сравнения: с электричеством люди экспериментируют с 18-го века — и ничего, работают до сих пор. Ещё пример — оптическая компьютерная мышь, которой уже 40 лет.

Вообще, аргумент о том, что если на что-то не было новых фреймворков, то это что-то плохое — глубоко порочный. Фреймворки появляются там, где нужно решать специфические задачи, а встроенные возможности языка этого не позволяют. Тогда строят надстройки в виде фреймворков. Если у языка много фреймворков, это скорее говорит о том, что этим языком пытаются решить нехарактерные для него задачи.

Читайте ещё по теме:

Объектно-ориентированное программирование: на пальцах

Откуда в программировании так много похожих языков? Почему все языки выглядят как один?

Про это есть анекдот:

есть 9 языков программирования, каждый решает свои задачи

— Нам нужен язык, который объединит все задачи и станет универсальным для всех ситуаций!

есть 10 языков программирования, каждый решает свои задачи

Я слышал, что будущее программирования за платформами без кода. Тогда нет смысла учиться программировать?

Платформы no-code появились 40 лет назад, и тогда у всех были такие же мысли. Но языки программирования общего назначения как были, так и живут до сих пор.

Что происходит на самом деле: программы, где нужны оценки, итерации, выбор, упорядочение, состояния, выражения, процедуры, функции и команды, — всё это без кода не работает. Или работает, но очень плохо и с ошибками.

Код в виде текста и команд на языке программирования — это самый эффективный способ выразить намерение компьютеру. Всё остальное медленнее и сложнее. Издержки от изучения даже самых основ программирования несоизмеримо меньше, чем неудобства от костылей no-code-платформ. Хотя и у них есть своя сфера применения.

Читайте ещё по теме:

- Зеро-код: это как?
- Что такое low-code и чем он отличается от зеро-кода
- Как сделать телеграм-бота без программирования

Про работу и ежедневную рутину

Сколько нужно времени, чтобы написать рабочий код?

Это зависит от того, что именно писать.

- Вывести «Hello, World!» на экране займёт минуту.
- Собрать приложение из заготовок со StackOverflow можно за несколько часов.
- Код для простой видеоигры можно написать в одиночку за несколько дней. Особенно если брать заготовки из библиотек.
- А вот над сложными играми работает целая команда и не один год.
- В отрасли принята практика делать рабочий код маленькими шажками — например по две недели. Команды работают над небольшими модулями, получая результат ритмично и регулярно.

> Правильнее так: рабочий код можно сделать хоть за минуту.
> А приложение для конечного пользователя можно делать долго — смотря какие у вас запросы.

В сериале «Кремниевая долина» это было наглядно показано: главный алгоритм придумал один программист за несколько дней в первом сезоне. А весь продуктовый обвес вокруг этого алгоритма программировали сотни людей, которые уже сидели в огромном офисе все следующие сезоны.

Читайте ещё по теме:

Как пишут код на разных этапах карьеры

Нужны ли комментарии в коде? А как написать хороший комментарий?

Платформы no-code появились 40 лет назад, и тогда у всех были такие же мысли. Но языки программирования общего назначения как были, так и живут до сих пор.

Что происходит на самом деле: программы, где нужны оценки, итерации, выбор, упорядочение, состояния, выражения, процедуры, функции и команды, — всё это без кода не работает. Или работает, но очень плохо и с ошибками.

Код в виде текста и команд на языке программирования — это самый эффективный способ выразить намерение компьютеру. Всё остальное медленнее и сложнее. Издержки от изучения даже самых основ программирования несоизмеримо меньше, чем неудобства от костылей no-code-платформ. Хотя и у них есть своя сфера применения.

Читайте ещё по теме:

- Зеро-код: это как?
- Что такое low-code и чем он отличается от зеро-кода
- Как сделать телеграм-бота без программирования

Сколько строк кода нужно, чтобы написать игру?

Вопрос бессмысленный. Не нужно вообще это рассматривать как мерило чего-либо. Это как говорить «сколько вдохов и выдохов нужно сделать, чтобы дойти до остановки?» Сколько нужно — столько и сделайте.

Вот в коде то же самое: никто не оценивает вашу работу по объёму написанного кода. Можно навалить 10 тысяч строк и получить спагетти-код, в котором никто не разберётся. А можно написать 10 строк, которые порвут индустрию.

Был случай: что какой-то разработчик написал пять строк кода, которые позволили ускорить рендеринг трёхмерной графики во много десятков раз, благодаря чему в мире появился Quake III Arena. Пять строк — это в масштабах всей игры вообще нисколечко, но именно они позволили продукту стать суперпопулярным.

Читайте по теме:

оптимизация в Quake III

Научиться такому:

Бесплатный тренажёр «Основы математики для цифровых профессий»

Правда ли, что у программистов есть проблемы со здоровьем?

Зависит от рабочего места и режима дня.

Если у разработчика неудобная клавиатура и он работает на ней по 12–15 часов в день на протяжении многих лет, то проблемы со здоровьем будут — и не только с суставами.

Если нужно постоянно напрягать глаза из-за плохого освещения, это тоже плохо для здоровья.

Если же человек регулярно делает разминки, занимается спортом и поддерживает нормальный режим работы, то ему ничто не угрожает.

Понятно, что программирование — это сидячая работа.
Но никто не мешает делать её стоя и самостоятельно следить за своим здоровьем. Благодаря ChatGPT и другим ассистентам больше нет никакой необходимости не спать по ночам и гробить здоровье за компьютером.

Что самое сложное в работе программиста?

Тут нет однозначного ответа. Вот проблемы, о которых разрабы говорят чаще всего:

- Плохой менеджмент, несогласованность целей и задач, срочная работа и авралы.
- Нечёткие требования к продукту, отсутствие технической документации.
- Слишком много времени тратится на планёрки, а не саму работу.
- Корпоративные ограничения безопасности, отсутствие доступа к каким-то справочникам или оборудованию.
- Сделанная работа уходит в стол, продукты не релизятся.
- Монотонный труд, обслуживание неинтересных продуктов.

Но вообще работа, корпоративная культура и задачи у всех разные. Нужно смотреть на конкретную компанию.
Кому щи пусты, кому жемчуг мелок.

Выбрать профессию по душе:

Бесплатный курс «Как выбрать профессию в IT»

Я за неделю разобрался в веб-программировании, зовут в стартап. Соглашаться?

Вообще стартапы — это школа жизни.

Если вам хочется быстро вырасти в разных направлениях, то стартап — лучшее место. У вас будет возможность тестировать новые фреймворки, писать много разного кода и делать всё, лишь бы программа заработала. Деньги тоже будут платить, но они могут у инвесторов внезапно закончиться.

Главный совет — идите в стартап не главным и единственным разработчиком, а подмастерьем у кого-то старшего. Потому что задач у стартапа гора, а без помощи старших товарищей может быть тяжело.

Читайте ещё по теме:

- Биотех-стартапы: что они делают и что в них интересного
- Как устроен стартап: российский корпоративный мессенджер «Пачка»

Хочу стать программистом. Сколько часов в день мне придётся писать код?

Многие разработчики замечают такое: чем дольше программист в профессии, тем меньше он пишет код.

Самое начало — 9 из 8 рабочих часов.
На начальных этапах разработчикам доверяют больше монотонной работы с кодом: написать, оставить комментарии, протестировать. Можно остаться после работы.

Уже освоился — 5 из 8 рабочих часов.
Чем опытнее разработчик, тем чаще он проверяет код других.

Опытный — 3 из 8 рабочих часов.
Типично для сеньоров: в какие-то дни они пишут полноценный код самостоятельно, а в какие-то делегируют обязанности.

Мастер — 0/8 рабочих часов.
Такое бывает у тимлидов. В некоторые дни они только решают рабочие вопросы и организуют работу отдела.

Читайте ещё по теме:

Как пишут код на разных этапах карьеры

Про навыки

Важно ли программисту быстро печатать?

Нет. В компаниях программист чаще читает чужой код, чем печатает свой. Главное не насколько быстро написан фрагмент, а насколько легко его понять.

Даже если в проекте кому-то придётся писать код с нуля, программист потратит больше времени на обдумывание, чем на набор текста.

Если сейчас вы печатаете медленно, то за полгода практики вы сами не заметите, как скорость резко вырастет.

Читайте ещё по теме:

Обязательно ли печатать вслепую, чтобы быть разработчиком?

Кем идти работать в IT без навыков программирования?

HR-специалист. В любую IT-компанию нужны люди, которые будут нанимать других людей. Нужны базовые знания о требованиях к программистам, чтобы объективно оценивать кандидатов.

SMM-специалист. Рынок соцсетей расширяется. Специалистов по созданию и дистрибуции контента в соцсетях требуется больше. Желательно понимать маркетинг, развивать навыки согласования и много пахать.

Системный аналитик. В стартапах такой специалист становится посредником между заказчиком и разработчиками. В крупные IT-компании требуется профильное образование и опыт работы. При этом часто спрашивают про навыки разработки и знание языков программирования, например Python.

Менеджер. Программистами нужно управлять, ходить вместо них к клиенту и согласовывать замечания к продукту.

Инженер тестирования. Это тот, кто обеспечивает качество итогового продукта: тестирует его вручную и автоматически.

Читайте ещё по теме:

- Кто такой инженер по тестированию и стоит ли на него учиться
- Как школьный учитель стал фронтенд-разработчиком
- Зачем нужны менеджеры

Научиться:

- Курс «Менеджер проектов»

Нужна ли математика программисту?

Чтобы писать большую часть кода, достаточно школьной математики. Самые популярные математические операторы в программировании — это плюс и минус, на втором месте — умножение и деление.

Продвинутая математика полезна в анализе данных и создании искусственного интеллекта. Но это узкая отрасль в программировании.

Ещё продвинутая математика нужна в сложном инженерном софте, в создании драйверов для видеокарт, в разработке движков физики и визуализации. Но люди, которые этим занимаются, погружены в эту тему со времён университета. Если у вас нет высшего образования в области математики или точных наук, это не страшно.

Читайте ещё по теме:

- Что читать начинающему программисту

Научиться:

- Бесплатный тренажёр «Основы математики для цифровых профессий»

У меня нет высшего образования. Могу ли я стать программистом?

Диплом о высшем образовании никак не связан с программированием. Чтобы стать хорошим разработчиком, нужно много практики и изучения возможностей языка. Диплом на это никак не влияет — можно отучиться на программиста и не уметь писать код, а можно выучиться на повара и в совершенстве освоить Python. И вдобавок хорошо готовить.

Если 20 лет назад самостоятельно научиться программировать было сложно, то сейчас миллион курсов, роликов в Ютубе и справочников. Без диплома — вполне реально.

Читайте ещё по теме:

Можно ли стать разработчиком, если учиться по вечерам

Сколько времени реально нужно, чтобы освоить программирование?

Есть теория, что чтобы развить любой навык до нормального уровня, нужно 10 лет или 10 тысяч часов. В программировании тоже так?

Простую программу по шаблону легко написать за несколько минут, более сложную — за несколько часов. При желании можно даже быстро найти решение в интернете, разобраться в нём за полчаса, допилить его под себя и получить нужный результат.

Чтобы перейти к более сложным проектам, потребуется много практики, а на это нужно время. Десять лет — неплохой срок, чтобы стать экспертом, но для изучения только одного языка это многовато.

Можно мыслить так: есть гении-маги-колдуны, которые пишут гениальные алгоритмы и создают прорывные технологии;
а есть просто разработчики, которые делают обычные продукты.
Там нет ничего такого, чему нужно учиться 10 лет — просто сидишь и решаешь насущные проблемы. К этому можно прийти за год.

Читайте ещё по теме:

- С чего начать в IT
- Сколько времени реально нужно, чтобы освоить программирование?
- Я ничего не понимаю в ИТ. С чего начать?

Про ошибки в работе

Может ли код сломать компьютер? Вдруг я что-то напишу и сломается вообще всё?

Технически код может вывести из строя компьютер, но для этого нужно долго и целенаправленно изучать все слабые места железа и операционной системы. Случайно такое сделать почти нереально.

Могут быть ошибки эксплуатации компьютера — например, если вы удалили важные файлы системы. Но это не прерогатива разработчиков — такое может сделать кто угодно.

Читайте ещё по теме:

- Что такое исключения в программировании
- Что означает ошибка KeyError
- Как поймать баг в коде: отладка в браузере

Какие самые частые ошибки у начинающего программиста в любой области?

Мы спросили это у опытных ребят, и вот что они ответили:

- «Джуны сразу берутся за работу: они лучше напишут 1000 строк кода, чем внимательно прочитают требования. Неопытным программистам часто кажется, что лучше быстрее решить проблему и переходить к другой, чем искать универсальное решение для всех похожих проблем».
- «Иногда открываю код, а там ни единого комментария. Вот и сиди разбирайся, какая здесь логика. Это беда начинающих программистов, которая с опытом исчезает — писать код, лишь бы написать, без пояснений».
- «Начинающие программисты свежи и голодны до кода. Они берут примеры из учебников и Stack Overflow, получают безумные конструкции. Конечно, в любом таком коде чувствуется энтузиазм и желание разобраться, но иногда это очень сложно сделать».

Читайте ещё по теме:

Как пишут код на разных этапах карьеры

Я ничего не понимаю в ИТ. С чего начать?

Нестыдные вопросы об ИТ

С чем самым страшным может столкнуться программист?

Самое страшное для программиста — когда компьютер завис, а ты последний раз сохранялся полдня назад. Или когда из-за перебоя напряжения сгорели диски, а в облаке последний бэкап был в начале года. Или когда коллега разлил кофе на ноут, а резервных копий ещё нет.

Ещё из страшного — сгоревший процессор из-за пыльного вентилятора. Если не чистить компьютер от пыли, потом он просто перестанет работать. За три часа до сдачи проекта это страшно.

Но всё это ерунда по сравнению с клиентами, у которых сегодня одно задание, а завтра — совсем другое. Но пусть об этом болит голова у менеджеров.

Читайте ещё по теме:

- Программы для безопасности компьютера
- Выбираем компьютер для программиста
- Что такое трояны и вирусы — на самом деле

Стать хорошим менеджером и работать с клиентами:

- Курс «Менеджер проектов»

Как выбор языка программирования влияет на надёжность программы?

Напрямую.

Например, в C и C++ программист может написать код и работать через него за пределами выделенной памяти. С одной стороны, иногда это удобно, а с другой — это делает уязвимой всю систему. Кто-то может воспользоваться таким выходом во внешнюю, незащищенную память и поместить туда, например, вирус или свой код.

С языками разметки, например, HTML или XML, всё гораздо строже: код не выходит за рамки, которые ему обозначил компьютер. Код получается более надёжным, но и возможности ограничены.

Вывод такой: чем больше возможностей предоставляет язык, тем больше потенциальных проблем может возникнуть в его коде. Наша рекомендация для старта в этом плане — JavaScript или Python.

Читайте ещё по теме:

- Вам мало языка C? Попробуйте C++
- Какой язык программирования лучше?

Выбрать язык:

- Бесплатный курс «Какую профессию в программировании выбрать»

Про сложности

У меня не получается разобраться в программировании, как бы я ни старался. Может, мне просто не дано?

Вам нужны практико-ориентированные курсы и несложные задачки.

Обычно не получается разобраться при такой схеме обучения:

1. Вы скачиваете официальный учебник (например про Swift).
2. Вы начинаете его читать, а там первый раздел, конечно же, посвящён типам данных. Они объясняются очень подробно, потому что типы данных действительно очень важны для языка.
3. Спустя неделю чтения вы всё ещё изучаете нюансы типов данных, но у вас никак не складывается картинка программирования в целом.
4. Вы решаете, что программирование не для вас.

А проблема была в том, что вас стали грузить тем, что вам на этом этапе не нужно (по крайней мере, в таких деталях). Вместо типов данных нужно было попробовать что-то посчитать, прогнать через цикл, нарисовать простой интерфейс и сделать рабочую программу.

Такая структура обучения применима для программистов, которые уже владеют другими языками и хотят освоить новый. Им не интересно, как сделать простейшую программу; им нужно сразу нырнуть в типы данных и особенности исполнения условий. А вам как новичку, наоборот, — нужно было начать с другого.

Эта проблема решается на хорошо составленных курсах: там вам в первые же дни дают задания, чтобы вы получили практически применимый результат. И уже потом, когда придёт время, вы с полной ответственностью нырнёте в нюансы типов данных.

Читайте ещё по теме:

- Какой язык программирования выбрать для старта
- Почему второй язык программирования выучить проще, чем первый?
- С чего начать в IT

Как сделать так, чтобы мне было интересно учиться программировать?

Очень просто: решать программированием интересные вам задачи. Не программирование ради программирования, а решение задач. Например, вам дико захотелось сделать собственную компьютерную игру на Unity — вот отличная задача для изучения языка C#.

Или вам пришла идею для чат-бота. Вы открываете урок «Чат-бот на Python за полчаса», и скоро у вас работает первый прототип.

Скучнее всего учить программирование в отрыве от задач.
Ну вот прочитали вы главу про кортежи. И что?

Читайте лучше вот это:

Про движок Unity

Что проще выучить: фронтенд или бэкенд?

Кто-то скажет, что бэкенд намного проще, чем фронтенд. Учишь Python, потом немного SQL, читаешь про пару библиотек, пишешь код и зарабатываешь много денег.

Кто-то, наоборот, уверен, что проще изучить основы фронтенда. Учишь HTML, CSS и JavaScript, десять простых фреймворков, немного осваиваешь React, пишешь корпоративные порталы и зарабатываешь много денег. Да, там больше инструментов, но они легче, и запомнить всё это будет не так сложно.

А часть ребят, становясь опытными программистами, всё-таки приходят к мысли, что на самом деле нет никакой разницы, что учить, потому что всё в итоге зависит от задачи.

Секрет в том, что проще выучить то, что интереснее. Вот и всё.

Читайте ещё по теме:

- Кто такой фронтенд
- Чем занимаются бэкенд-разработчики

Научиться тому и другому:

- Курс «Веб-разработчик»
- Курс «Python-разработчик»

Начал читать про современный фронтенд и обалдел. Почему всё стало так сложно?

Потому что 10 лет назад требования к веб-приложениям были не такими высокими, а технологии были не такими развитыми. Сейчас фронтенд — это полноценное пространство для создания приложений. Все уже забыли, что когда-то давно веб был просто текстовыми документами со ссылками.

Разработчику на фронтенде придётся многое выучить. Стартовый набор: HTML, CSS, JavaScript, препроцессоры, фреймворки, другие фреймворки, ещё фреймворки и один из языков программирования, на котором работают в проекте.

Кроме того, инструменты разработки интерфейса постоянно развиваются, поэтому придётся быстро адаптироваться. Если три года назад синтаксис в Bootstrap был один, то сейчас в него добавилось много нового. А если вы хотите делать действительно классные интерфейсы, придётся ещё выучить Vue.js или Angular.

Но люди этим всем владеют, так что нет ничего невозможного.

Читайте ещё по теме:

- Кто такой фронтенд
- 116 тысяч рублей в месяц — средняя зарплата для фронтенда. Как им стать
- Vue.js — конструктор для веб-приложений

Научиться делать крутой фронтенд:

- Курс «Веб-разработчик»

Как изучать программирование, чтобы это было максимально эффективно?

1 Используйте отладчик (debugger)

Это позволит вам понять, что происходит внутри любых разделов кода. Вы увидите важность переменных и сможете проверить их значения в любой момент. Отладчик сразу покажет, что где лежит и как изменяется в процессе.

Если не использовать отладчик, придётся тратить слишком много времени на поиск ошибок, а это будет вас нехило тормозить.

2 Практикуйтесь, а не размышляйте

Как только вы узнаете что-то новое — напишите программу, где это используется. Просто прочитать про какую-то конструкцию — считайте, что время потрачено зря.
Нужно тут же применить в деле.

3 Запускайте свой код, а не читайте его в учебнике

Когда кто-то изучает учебник по программированию, может посмотреть на фрагмент кода оттуда и заявить: «Я всё понял». Это прекрасное чувство, но это не правда. Есть только один способ проверить, отложился код в голове или нет: написать программу и запустить её.

Читайте ещё по теме:

Как поймать баг в коде: отладка в браузере

Карьерный путь: руководитель группы в лаборатории ИИ Сбера

Если я вместо самостоятельного написания кода буду искать готовое решение на StackOverflow — я бездарь и у меня нет шансов?

Абсолютно нет — и вы точно не бездарь. Так работают почти все разработчики в реальном мире: когда у них проблема, они ищут ответ на StackOverflow, находят его и двигаются дальше.

Неопытный программист — тот, кто пытается самостоятельно разобраться в вопросе, ответ на который уже давно написан и оптимизирован.

Ещё неопытным программистам может казаться, что у них достаточно опыта. Они пишут код по памяти, хотя могут за минуту взять часть кода в библиотеке и потратить оставшееся время на проверку. Такие программисты часто отказываются от помощи, а зря. Перенимать опыт других специалистов полезнее, чем досконально изучать каждую мелочь.

Поэтому, если вы знаете, что ваша проблема уже решена и можно просто взять готовое решение, — берите.

Читайте ещё по теме:

Использовать чужой код стыдно?

Как научиться думать как программист?

Решайте логические или математические задачи.
Если не получается их решить самостоятельно — читайте ответы. Логические задачки помогают расширить ваш мыслительный инструментарий.

Второй способ — когда берёте код со StackOverflow, читайте его, даже если он у вас просто работает. Как думал этот разработчик? Это займёт немного больше времени, чем просто скопировать чужой код, но ваше мышление прокачается.

Читайте ещё по теме:

- «Никому не нужны высокомерные гении»
- Олимпиадное программирование: что, как и почему Россия впереди всего мира

Узнать себя в ИТ:

- Бесплатный тест для профориентации

Оборудование

Что выбрать для программирования: Mac или Windows?

Если «просто программировать» — то нет разницы, что выбрать.

Если нет веских причин сменить платформу, то работать на той, которую вы уже используете в жизни. Так вам не придётся параллельно с языком осваивать и новую операционную систему. Это сложно и неудобно.

А в целом общее правило звучит так:

> Если хотите создавать приложения для iPhone, нужен Mac.
> Чтобы создавать программы для Windows или игры для Xbox, потребуется Windows.
>
> Чтобы делать мобильные приложения, пойдёт любая система.
> Для бэкенда вообще без разницы, что выбрать, всё равно код будет работать удалённо на сервере.

Покупать новое оборудование для программирования стоит только тогда, когда вы понимаете, в чём именно вас ограничивает нынешний компьютер. Вам мало оперативной памяти или нужен более быстрый процессор? Вам неудобно с этой старой клавиатурой или вам главное — хорошая видеокарта для расчёта нейросетей? Исходя из этого вы сделаете правильный выбор.

Читайте ещё по теме:

- Выбираем компьютер для программиста
- Зачем устанавливать Ubuntu

Хватит ли ноутбука для программирования? Какой нужен?

Да, большинство современных офисных ноутбуков подходят для программирования. В зависимости от задач вам может потребоваться более быстрый процессор или больше памяти, но это нужно для ускорения работы и большего комфорта. Это не необходимость.

Единственная проблема современных ноутбуков — они плохо обновляются. Если бюджет для вас критичен, а мобильность нет — лучше купить стационарный компьютер и постепенно обновлять в нём компоненты. Не потому, что он быстрее, а потому, что стационарник прослужит вам намного дольше ноутбука.

Читайте наш гайд по выбору ноутбуков в 2023 году

Нужна ли особенная клавиатура для программирования?

Для программирования и вообще для долгой работы за компьютером нужна удобная клавиатура. Главный критерий удобства — ваши руки не устают во время работы, запястья не напрягаются.

- Некоторые люди работают, положив запястья на стол, и тогда им нужна максимально низкая и плоская клавиатура.
- Кому-то привычнее держать руки над клавиатурой, тогда им нужен комфортный наклон.
- Кому-то нужна специальная подставка под руки.
- Кто-то привык к разделённой на две части эргономичной клавиатуре.

Некоторые разработчики вкладывают много денег в особенные клавиатуры — механические, разделенные, программируемые, особенно износостойкие, с особенным откликом или необычным расположением клавиш. В этом нет ничего плохого, если нравится — берите.

На качество кода клавиатура точно не влияет. На скорость работы влияет. На удобство может влиять как в лучшую, так и в худшую сторону. Например, в нашей редакции мы перепробовали все возможные авторские клавиатуры, и в итоге самой удобной для нас оказалась обычная клавиатура Apple. Но у нас вот такие руки, а у вас какие-то другие.

Читайте об этом:

Механические клавиатуры для программиста

Макропад для повышения эффективности

Сколько мониторов нужно программисту?

Программист может работать за одним монитором. Но обычно используют несколько, например так:

- На одном мониторе код, на другом — результат.
- На одном мониторе код, на другом — справочники.
- На одном стили страницы, на другом структура страницы и интерактивные элементы, на третьем — результат.
- На основных мониторах идёт работа,
  на вспомогательном — отслеживание ресурсов системы.

Если у вас стационарное рабочее место, можно начать с одного монитора; понять, что вам нужно от работы; и после этого установить столько и в такой конфигурации, как вам нужно. Никакого золотого стандарта для всех разработчиков не бывает.

Я всё понял. Что дальше?

Дальше всё просто: выбираете курс Яндекс Практикума себе по душе, осваиваете новую профессию, зарабатываете много денег и покупаете себе много подарков. А если серьёзно, то вот:

- Бесплатный курс «Как выбрать профессию в IT»
- Бесплатный тест для профориентации
- Бесплатный курс «Какую профессию в программировании выбрать»
- Бесплатный курс «Какую профессию выбрать в анализе данных»
- Бесплатный тренажёр «Основы математики для цифровых профессий»
- Курс «Python-разработчик»
- Курс «Веб-разработчик»
- Курс «Инженер по тестированию»
- Курс «Менеджер проектов»
- Курс «Интернет-маркетолог»
- Курс «Профессиональная вёрстка на HTML и CSS»