Claude Code: полное интервью Boris Cherny (Claude Code) и Alex Albert (Claude Relations)

https://www.youtube.com/watch?v=Yf_1w00qIKc

Введение: что такое Claude Code

Claude Code — это способ агентного программирования в терминале. Вам не нужно осваивать новые инструменты, использовать новые IDE или заходить на какой-то конкретный сайт. Это просто агентное программирование, и оно работает там, где вы работаете. Я думаю, что это на самом деле возникло из того, как инженеры и исследователи Anthropic используют инструменты, чтобы выполнить свою работу.

Потому что у людей самые разные стеки. Это довольно странно. Нет одного нормального стека, который используют **все**. Кто-то использует Z ID, кто-то — VS Code, а есть те, кто говорит: «Вы никогда не отнимете у меня мой Vim, наверное, только с моих холодных мёртвых рук». И мы хотели создать что-то, что подходит для всех. И вот так мы в итоге оказались в терминале. Терминал — почти самый универсальный из всех интерфейсов, потому что он гибкий и уже встроен в рабочие процессы каждого. И к тому же он просто самый простой. Потому что он такой простой, мы можем очень быстро делать итерации. Оглядываясь назад, это оказалось хорошей вещью, хотя изначально такой цели не было.

Если я новичок-разработчик и хочу использовать Claude Code, как вообще выглядит процесс запуска этого продукта? Да, всё довольно просто.: скачиваешь его через

```
npm install -g addanthropic-ai.Claude-code
```

Немного похоже на заклинание, конечно. Так что, чтобы работать с ним, нужен только Node.js на системе, а у многих он уже есть. И всё. Открываешь, и он проведёт тебя через всё остальное. Просто вводишь Claude в терминале, нажимаешь Enter, и всё. Claude проведёт тебя через оставшийся процесс. И вы просто начинаете с ним

разговаривать, и он сразу начинает кодить.

Одна из классных вещей — он работает в любом терминале, как вы и сказали. Так что если вы используете, например, iTerm2, терминал Apple или любой другой терминал, даже в SSH-сессии или TMAC-сессии, он будет работать. Один из основных способов, как люди используют Claude code— это терминалы внутри IDE. Так что одна из возможностей — запускать Claude, например, терминал VS Code. И он становится ещё мощнее. Вместо того чтобы видеть изменения файлов в терминале, вы увидите их крупно и красиво прямо в самой IDE. И мы используем больше сигналов из этой IDE, чтобы сделать Claude ещё умнее. Но опыт остаётся тем же. Вы просто за-пускаете Claude в терминале.

Релиз и реакция сообщества

Claude Code мы выпустили в феврале. Прошло чуть больше трёх месяцев. Реакция сообщества была просто

сумасшедшая. Это было так неожиданно. До релиза мы не были уверены, хотим ли вообще его выпускать. Это инструмент, который внутри компании делает наших инженеров и исследователей невероятно продуктивными. У нас шли споры: не секретный ли это ингредиент? Мы точно хотим отдавать его людям? Потому что это тот самый инструмент, которым каждый в Anthropic пользуется каждый день. Думаю, это было правильное решение, потому что он повышает продуктивность и людям он нравится.

Когда мы поняли, что нужно выпускать его: всё началось с небольшой группы. Пару человек из нашей основной команды пользовались им. А потом мы дали его всем сотрудникам Anthropic, и появился этот график DAU. Например, ежедневная активность — просто смотрели на сотрудников, и это продолжалось вертикально три дня подряд. Вау. Мы подумали: ладно, это безумие, это хит. А потом в какой-то момент мы дали это нескольким внешним людям, чтобы проверить — не сошли ли мы с ума, полезно ли это? И вся обратная связь была исключительно положительной.

Сначала это действительно взлетело внутри Anthropic. Потом все инженеры и исследователи начали использовать это сами, и для нас стало очевидно, что мы должны выпустить это в мир. Да, именно так мы в основном и развивали этот продукт. Например, Claude code написан с помощью Claude code. Почти весь код в Claude code был написан, переписан и переписан снова с использованием Claude code. Мы очень верим в dogfooding. Это очень важно, потому что когда ты используешь продукт, который явно прошёл dogfooding, это ощущается. В продуктах, которыми я пользуюсь каждый день, чувствуешь, что команда постоянно ими пользуется, а в других — нет. Мы просто хотели, чтобы этот продукт был именно таким: когда берёшь его в руки и пробуешь, сразу понятно, что в него вложено много любви и что мы сами им пользуемся.

Кого ты считаешь идеальным клиентом Claude Code на данный момент? Какой это тип человека, какой тип разработчика? Думаю, самое главное — Claude Code довольно дорогой. Если ты кодишь по выходным, можешь попробовать немного. Получаешь API-ключ, вносишь пять долларов и просто пробуешь.

Ценообразование и подписка Claude Max

Но если хочешь использовать для более серьёзной работы, это будет стоить дороже: 50, 100, 200 долларов в месяц, что-то в этом роде. Диапазон довольно большой. Зависит от того, для чего ты его используешь. Но в целом рассчитывай примерно на 50 долларов в месяц. Много крупных компаний уже используют его. Если ты в большой компании, это обычно отличный вариант. Он отлично работает с большими кодовыми базами. Нет этапа индексирования, не нужно ничего дополнительно настраивать. Просто запускаешь, и он работает из коробки практически с любой большой кодовой базой на любом языке.

А что это за интеграция с Claude Max? Как это работает? Да, мы заметили, что когда люди использовали API-ключи для оплаты, они немного переживали из-за своего использования. И поэтому не использовали это так часто, как хотели. Вы платите за подписку на Max, которая стоит около \$100 или \$200 в месяц, выбираете ценовой уровень. И существуют разные лимиты использования. И вы получаете практически столько Claude code, сколько хотите. И на практике вы не столкнётесь с ограничениями по скорости. Очень немногие сталкиваются. Это неограниченный Claude code.

Claude code — это просто один пакет подписки. Кодовую базу, над которой я работаю на своём компьютере, открываю терминал, набираю Claude, нажимаю Enter. Claude приступает к работе. Он будет использовать инструменты, выходить и выполнять свои задачи, проходить множество шагов. Если вы раньше пользовались помощниками по программированию только в IDE и привыкли к тому, что система просто дописывает строку или несколько строк, то это совсем не так. Это совершенно другой уровень.

Рабочий процесс и агентное программирование

Claude понимает ваш запрос и использует все доступные инструменты — bash, редактирование файлов и так далее, чтобы исследовать кодовую базу, читать файлы, получать необходимый контекст, а затем редактировать файлы и вносить любые изменения, которые вы хотите.

Возможно, это новый формат программирования по сравнению с тем, как мы это делали последние 20 или 30 лет. Я занимаюсь программированием уже некоторое время, но мой дедушка на самом деле был одним из первых программистов в Советском Союзе ещё в 1940-х или около того. Он программировал с помощью перфокарт, потому что программирование с использованием программного обеспечения тогда ещё не существовало. Он брал эти большие перфокарты, а в США была такая штука от IBM, которая была своего рода идеей того времени. Вот так он и программировал. Он приносил эти карты домой каждую ночь, и моя мама рассказывала мне истории о том, как она рисовала на них восковыми мелками. Для неё это была часть детства.

А с тех пор программирование сильно изменилось. Сначала были перфокарты, потом ассемблер, а затем появились первые языки высокого уровня. Знаешь, COBOL, Fortran, а потом в 80-х появились Java и эти типизированные языки, Haskell — это было действительно захватывающе. Это интерпретируемые языки, которые при этом обеспечивают высокий уровень безопасности.

Я воспринимаю язык программирования и опыт его использования как развивающиеся синхронно. Потому что примерно в то время, когда начал появляться Java, например, появился Eclipse IDE с первыми функциями автодополнения. Вы вводили символ, и появлялось выпадающее меню. Идея была в том, что вы выбираете: это, это или вот это? Это было просто невероятно. Потому что как человек, вам больше не нужно было читать код. Языки, как мне кажется, выровнялись. Все современные языки принадлежат к похожим семействам. Существует несколько крупных семейств языков, и они довольно похожи, если взглянуть. Но опыт действительно развивается, и теперь вам не нужно иметь дело с перфокартами, ассемблером или даже самим кодом. Вы работаете с подсказками, а модель разбирается с кодированием. Для меня, как программиста, это невероятно захватывающе. Мне это очень нравится. Мы прошли путь от перфокарт до подсказок, по сути.

Модели

До недавнего времени Claude Code в основном работал на Claude 3.7 Sonnet. А теперь, когда под капотом Claude Code работают модели Claude 4, что это открыло и куда, по моему мнению, мы движемся? Да, примерно за пару месяцев до выхода моделей мы начали пробовать их внутри команды. Я до сих пор помню, как у меня отвисла челюсть от того, насколько они стали мощнее. Так что, думаю, открылись совершенно новые сценарии использования. Когда ты используешь Claude code синхронно в терминале, одна из больших задач - убедить его следовать твоим инструкциям. Ты говоришь ему что-то сделать — либо в подсказке, либо в Claude MD — и он, как правило, просто делает это и придерживается задачи. Это большое изменение, потому что 3.7 был настоящим зверем. Это потрясающая модель для кодирования, но, честно говоря, с ней

сложно управляться. Ты пытаешься написать тесты, а она просто издевается над всеми твоими тестами, и ты такой: нет, это не то, что я имел в виду. Обычно ты говоришь это один или два раза, и она понимает. Но она была настолько мощной, что это того стоило. И мне кажется, что теперь с этим новым поколением моделей четвёртого уровня этого делать больше не нужно. обычно делает то, что ты хочешь, с первого раза. И Orus ощущается как следующий уровень выше Sonnet, где не только отлично понимает мои намерения, но и может с первого раза выполнить многое, что предыдущие модели просто не могли. Например, **я не писал юнит-тесты уже несколько месяцев.**

GitHub Actions и автоматизация PR

Orus просто пишет мои тесты, и почти всегда с первого раза идеально. И это довольно полезно в терминале. Это позволяет быть немного менее вовлечённым, но, думаю, один из самых крутых вариантов использования — запускать его в GitHub Actions и другие среды, где можно задать задачу, а модель просто возьмёт и выполнит её самостоятельно. Когда она возвращается с правильным результатом с первого раза — это просто потрясающе. Так что с GitHub Actions теперь мы можем прямо в GitHub, используя Claude, запускать задачу в фоне, а потом получать результат и новый PR. Именно так. Да, ты открываешь Claude в терминале как обычно, просто запускаешь Claude, а потом вводишь `slash install GitHub action`. И это проведёт тебя через процесс установки. Там несколько шагов, всё автоматизировано. Тебе нужно просто нажать пару кнопок, и приложение Claude установится в твой GitHub репозиторий. И да, опыт использования довольно классный. В любом issue можно упомянуть Claude, просто добавь Claude. Я использую его в PR каждый день. Коллега создаёт pull request. И вместо того чтобы спрашивать: «Эй, можешь исправить эту штуку?» Я просто скажу: «Эй, добавь Claude, исправь эту штуку». И он это исправит. И вместо того чтобы спрашивать: «Можешь написать тесты?»

Я всегда немного чувствую себя виноватым, когда приходится это делать. Я просто скажу: «Эй, добавь Claude, напиши тесты». Это просто перестало быть проблемой. Для меня это невероятно. Это как совершенно новый аспект программирования, когда мы можем просто подключить твоего всегда доступного программиста, который исправит эти проблемы за тебя, даже не на твоём компьютере, а работая в фоне. Да, и я думаю, что это начало взаимодействия с моделью так же, как с коллегой-программистом. Так что вместо того чтобы упоминать коллегу, я буду упоминать Claude.

Как это изменит разработку ПО, когда мы перейдём к такой модели?

Мы управляем всеми этими Claude code в фоне. Думаю, здесь нужен определённый ментальный сдвиг, потому что некоторым людям очень нравится контролировать код. Я думаю, сейчас индустрия смещается в сторону того, что вы управляете агентами, которые пишут ваш код. И это больше про ревью кода, чем про его ручное написание. Да, думаю, людям нужно как-то справиться с этим переходом. И я считаю, что для программиста это невероятно захватывающе, потому что теперь можно делать гораздо больше и гораздо быстрее. Все еще есть моменты, когда мне приходится опускаться и писать код вручную, но теперь я этого боюсь, потому что Claude просто отлично с этим справляется. Приходится писать код вручную либо из-за сложной модели данных, либо из-за чего-то действительно сложного, например взаимодействия между множеством системных компонентов или чего-то, что сложно описать в промпте. Думаю, это будет постепенно уходить на второй план, и программирование всё больше будет про управление агентами.

Новый тип рабочего процесса

Как ты сейчас используешь сочетание всех этих вещей — от интеграции в IDE до самого Claude Code в терминале и фоновых действий в GitHub?

Я думаю, что у меня есть два типа работы. Некоторые задачи действительно простые. Например, написание тестов или небольшое исправление бага. И обычно я прошу Claude сделать это через GitHub issues. Или другое, что я делаю — у меня обычно запущено несколько экземпляров Claude параллельно. У меня есть несколько копий нашего кода. И в одной из этих вкладок терминала я прошу Claude что-то сделать. Я нажимаю Shift-Enter, чтобы войти в режим авто-подтверждения. А потом через несколько минут возвращаюсь и получаю уведомление в терминале: Claude завершил работу. Есть второй тип работы, где нужно быть немного более вовлечённым. И я думаю, что это всё ещё большая часть инженерной работы. Как и в большинстве инженерных задач, нельзя сделать всё с первого раза, это всё ещё сложно.

Поэтому я запускаю Claude в терминале IDE, прошу его что-то сделать, и в какой-то момент он может застрять или код будет неидеален, или что-то в этом роде. Так что я зайду и отредактирую в своей IDE, чтобы сделать последние штрихи. Понимаю, получается, что есть своего рода спектр сложности задачи в зависимости от того, как ты взаимодействуешь с Claude. Да, когда начинаешь пользоваться таким инструментом, есть период обучения. Думаю,

что иногда люди пытаются использовать его для слишком сложных задач, дают что-то слишком трудное, и он «зависает», а результат их не устраивает. И это то, через что всем приходится пройти, чтобы внутренне понять, на что способен Claude, что он может сделать с первого раза. Что он может сделать с двух попыток и как выглядит это взаимодействие? К сожалению, это меняется с каждой моделью. Так что нельзя просто запустить один раз.

С каждым новым релизом возможности растут, и Claude может делать больше вещей правильно с первого раза. Так что каждый раз можно просить чуть больше. Верно, я заметил, что даже вне кода эти модели меняются и улучшаются так быстро, что если ты пробовал модель полгода назад и списал её со счетов для какой-то задачи, то сейчас уже нельзя так думать. В наши дни приходится почти каждый раз заново перестраивать свои интуиции.

Советы и приёмы использования

Я бы сказал, что самое главное, что я видел у продвинутых пользователей как внутри Anthropic, так и вне — это просьба к Claude составить план перед тем, как начать кодить. Иногда, когда люди только начинают использовать Claude code, они говорят: «Эй, напиши эту большую, сложную функцию», а потом расстраиваются, когда она не получается так, как они себе представляли. Очень хороший способ согласовать то, что вы хотите сделать, с тем, что Claude хочет сделать — это попросить его сначала составить план и показать его вам. Я иногда прямо говорю: вот проблема, которую я хочу решить. Прежде чем писать код, обдумай несколько идей и составь список вариантов подхода, но пока не пиши код. Claude предложит мне вариант один, вариант два, вариант три, я могу сказать, что вариант один и три звучат хорошо, давай их объединим. Теперь можешь начинать кодить. И он действительно хорошо слушает. Еще один способ поднять это на новый уровень — попросить Claude использовать расширенное мышление. И это работает лучше всего, если у Claude уже есть какой-то контекст. Если у Claude нет никакого контекста, это работает не очень хорошо, и он просто думает. И это похоже на человека, правда? Ты хочешь действительно понимать, что делаешь. И с Claude то же самое. Просто попроси его сначала прочитать файлы, затем сделать паузу, потом я попрошу его подумать и обдумать идеи, а потом — написать код. Интересно. Такой чередующийся подход, когда он может вызвать инструмент, проанализировать результаты, подумать, что делать дальше, вызвать другой инструмент и продолжить этот обмен.

И мы действительно видим это и на внутренних бенчмарках. В целом, если сначала получить контекст, затем подумать, а потом использовать инструменты для редактирования, `bash` и так далее, результаты значительно лучше. И

именно так это ощущается и для пользователя.

Claude MD файлы

Они кажутся действительно мощными. Да, Claude MD мы используем для всего. Это память Claude. Это инструкции для Claude, которые вы хотите разделить с вашей командой. Это инструкции, которые вы хотите разделить между всеми вашими проектами. Так что да, это очень мощно. Существует много разных видов Claude MD. Самый простой — это файл под названием Claude.MD, который вы кладёте в корень вашего репозитория. Это markdown-файл, Claude автоматически прочитает его, когда вы запускаете проект в этой папке. Так что это будет автоматически загружено в контекст. Любые инструкции, которые вы хотите, чтобы Claude выполнял каждый раз. Например, bash-команды, которые вы хотите запускать часто, или файлы, о которых он должен обязательно знать при внесении изменений или при принятии крупных архитектурных решений — что угодно.

Просто положите это в Claude.MD. Есть второй тип Claude.MD, и этот вы коммитите. То есть вы хотите поделиться им с вашей командой. Вы хотите написать это один раз и затем поделиться со всеми в вашей команде, чтобы людям не приходилось писать это самим. Второй тип Claude.MD — только для вас. Он называется Claude.local.md. Этот файл тоже кладётся в то же место и предназначен только для вас. Вы не делитесь им с командой, можете игнорировать его и не коммитить. Третий — глобальный Claude.MD. Он находится в папке .Claude в вашем домашнем каталоге. Большинство людей на самом деле не используют этот файл, но если хотите, можете положить туда любые инструкции, которые хотите разделить между всеми Облаками. И наконец, вы можете разместить Claude.MD в любом вложенном файле в любой директории вашего кода. Облако автоматически подтянет их, когда сочтёт это уместным, чтобы получить инструкции по работе с этой частью кода. Это как конкретные инструкции или даже ваши предпочтения — например, стиль кодирования или что-то подобное о том, как Облако должно взаимодействовать, что оно должно знать о вас, как вам нравится работать. Это может быть что угодно. Точно. И иногда, когда я вижу, что Claude делает что-то действительно хорошо или плохо во время разговора, я нажимаю знак решетки, и это переходит в режим памяти. И я говорю Claude: эй, тебе стоит это запомнить. Это может быть инструкция, например, всегда запускать линтер при изменениях в коде. Я говорю ему это, и он добавляет это в нужный файл памяти.

Будущее продукта

Что дальше для Claude Code? Да, я думаю, что мы рассматриваем два направления. Первое — как сделать Claude ещё лучше с вашими инструментами. Это началось с работы с каждым терминалом, теперь он может работать со многими IDE, а также с

множеством CI-систем. Мы думаем, что будет дальше. Просто чтобы убедиться, что он работает со всеми вашими инструментами, Claude должен уметь ими пользоваться и работать с ними нативно. Второе — как сделать Claude лучше в простых задачах, для которых вы, возможно, не хотите открывать терминал, например, в чат-приложении или чем-то подобном, чтобы он мог исправить проблему за меня. Так же, как я могу это сделать на GitHub. И что это значит? Что вам кажется удобным? Мы пробуем разные варианты. Мы хотим убедиться, что всё действительно удобно, прежде чем дать это пользователям. Это здорово. Я бы очень хотел видеть Claude Code повсюду.