🤖 Claude Code: полный гайд для начинающих

Ключевые тезисы:

- Claude Code — это CLI-инструмент (агент) на базе LLM (Claude Opus), который может управлять компьютером через терминал и писать код.
- Инструмент подходит даже для тех, кто не умеет программировать (*вайб-кодинг*).
- Основные применения: создание сайтов, Telegram-ботов, автоматизация бизнес-процессов, личный ИИ-ассистент.
- Работает через расширение в VS Code (рекомендуется новичкам) или напрямую в терминале (больше возможностей).
- Для работы требуется подписка на Claude (от $20/мес).

🔥 Что такое Claude Code и зачем он нужен

Claude Code — это инструмент от Anthropic, который даёт *большой языковой модели* (LLM) «руки» для взаимодействия с компьютером. Поскольку LLM генерирует текст, она может:

- Писать команды для терминала и управлять системой.
- Создавать и редактировать код (код — это тоже текст).
- Автоматизировать задачи в цифровой среде.

Для кого: Новичкам в программировании, тем, кто хочет автоматизировать рутину, создавать простые приложения или использовать ИИ как личного ассистента.

⚖️ Сравнение с конкурентами

Основные конкуренты в сфере AI-инструментов для кодинга:

- **Cursor** — первая популярная IDE, заточенная под *вайб-кодинг*.
- **VSCode Copilot** — встроенный AI от Microsoft в VS Code.
- **Windsurf / AntiGravity (от Google)** — аналогичные IDE.
- **OpenCoder** — open-source альтернатива для локального запуска моделей.
- **Российские аналоги:** Кодик (Сбер), Гигакод.

Почему именно Claude Code?

- Самый продвинутый и «навороченный» инструмент, где новые фишки появляются в первую очередь.
- Высокий уровень безопасности по сравнению с некоторыми аналогами (например, OpenClow).

💰 Тарифы и подписки

Действует единая подписка на все продукты Claude:

- **Бесплатный тариф** — Claude Code недоступен.
- **Pro ($20/мес)** — минимальный для доступа к Claude Code, но есть жёсткие лимиты.
- **Max ($100/мес)** — в 5 раз больше токенов, чем у Pro. Автор использует этот план.
- **Enterprise ($200/мес)** — в 20 раз больше токенов, чем у Pro, для комфортной работы без лимитов.

Важно:

- Лимиты считаются за 5 часов и за неделю.
- Можно подключить **оплату по API** (плата за фактическое использование токенов), но это обычно дороже подписки.

- При исчерпании лимита по подписке можно докупать токены через API.

🛠️ Установка и настройка

1. Установка через расширение в VS Code (рекомендуется новичкам)

1. Скачать и установить **Visual Studio Code**.
2. В разделе расширений найти **Claude Code** и установить.
3. Авторизоваться через `Claude Subscription` (требуется активная подписка).

2. Установка через терминал (больше возможностей)

1. В терминале выполнить команду: `npm install -g @anthropic-ai/claude`.
2. Запустить: `claude`.
3. Пройти авторизацию и настройку.

Что такое IDE (Integrated Development Environment)?

- Среда разработки (например, VS Code, AntiGravity, Cursor).
- Состоит из: файлового менеджера, текстового редактора, чата с ИИ и терминала.

🎮 Интерфейс и режимы работы

Основные элементы интерфейса (в расширении VS Code):

- Чат с Claude.
- История диалогов (синхронизируется с веб-версией).
- Контекстное окно (файлы, добавленные в обсуждение).

- Панель режимов и команд.

🔧 Ключевые режимы:

- **Ask Before Edits** — запрашивает разрешение на *каждое* действие (создание файлов, команды в терминале).
- **Edit Automatically** — автоматически редактирует файлы, но запрашивает разрешение на *опасные* команды в терминале.
- **Bypass Permissions** (рекомендуется) — не спрашивает разрешения вообще. ⚠️ **Внимание:** есть риск случайного повреждения системы, но маловероятен при внимательном контроле.
- **Plan Mode** — сначала создаёт подробный план (сохраняет в файл), затем выполняет его.

🗣️ Голосовой ввод

- В терминале доступен встроенный голосовой режим (активируется командой `/voice`).
- В расширении VS Code голосового ввода нет, можно использовать сторонние инструменты (например, AquaVoice).

📁 Ключевые концепции и файлы

1. `claude.md` — системный промпт

- Это файл с инструкциями для Claude, который отправляется в начале *каждого* диалога.
- Может быть **глобальным** (действует на все проекты) или **локальным** (только для текущего проекта).
- **Как создать:** вручную или командой `/init` — Claude проанализирует проект и создаст описательный `claude.md`.
- **Рекомендации:** писать конкретно, без «воды», до 200 строк.

2. Правила приоритета настроек (от высшего к низшему):

1. Enterprise (корпоративные настройки).
2. Локальные (только для вашего проекта).
3. Проектные (для команды, попадают в Git).
4. Глобальные (ваши личные настройки).

3. Rules — правила

- Аналогичны `claude.md`, но можно привязывать к конкретным папкам или файлам.
- Удобно для разделения промптов (например, отдельные правила для фронтенда и бэкенда).

Пример правила для голосового ввода:

```
path: "крестики-нолики/**"  # Применяется ко всем вложенным папкам
Пользователь использует голосовой ввод, возможны искажения. Термины "код-код", "Claud
```

4. Settings (`settings.json`)

- Настройки разрешений, хуков, модели по умолчанию, языка ответов.
- **Хуки (Hooks)** — автоматические действия по событиям (например, уведомление при завершении задачи).
- **Файл** `.env` — для хранения секретных данных (API-ключей), не попадает в Git.

🔧 Ключевые концепции: Skills, Subagents, MCP

🔧 Skills (Скилы)

Что это: *Текстовые инструкции (регламенты) для автоматизации рутинных или регулярных задач.* Вместо создания сложных workflow (как в n8n) вы описываете

процесс в текстовом файле.

Как работают:

- Claude видит только **название и краткое описание** скила, а не всю инструкцию.
- Полный контекст скила подгружается **только в момент его вызова**, что экономит токены.
- Скилы могут быть **глобальными** (для всех проектов) или **локальными** (для конкретного проекта).

Примеры использования:

- **Парсинг YouTube** — автоматический сбор информации о новых видео по ключевым словам.
- **Генерация презентаций** — создание слайдов в едином стиле на основе текста.
- **Обработка постов** — структурирование надиктованных мыслей в читаемый текст для соцсетей.
- **Работа с Docker** — единая инструкция для корректного запуска проектов в изолированной среде.
- **Использование прокси** — автоматическое подключение при региональных ограничениях API.

Преимущество: Экономия контекстного окна и возможность повторного использования стандартных процедур.

Subagents (Субагенты)

Что это: *Изолированные агенты-помощники, которых вызывает основной Claude для выполнения конкретных задач.*

Зачем нужны:

- ⚠️ **Изоляция контекста:** Задачи, требующие много технической информации (например, веб-поиск или ревью кода), выполняются в отдельном "чистом" контекстном окне субагента.

- **Сжатый ответ:** Субагент возвращает основному агенту только итоговый, сжатый ответ, а не весь объем обработанных данных.
- **Автоудаление:** После выполнения задачи субагент удаляется.

Популярные кейсы:

- **Web Search Agent** — для поиска информации в интернете.
- **Code Reviewer Agent** — для "свежего" взгляда и проверки безопасности кода.
- **Research Agent** — для изучения новой кодовой базы или документации.

MCP (Model Context Protocol)

Что это: *Протоколы для подключения внешних сервисов и данных к Claude.*

Must-have MCP:

1. **Context 7** — 🔥 **Критически важен.** Даёт доступ к **актуальной документации** библиотек, фреймворков и сервисов (обучение Claude заканчивается ~2025 год).
2. **Playwright** — позволяет Claude **управлять браузером** (открывать страницы, кликать, заполнять формы). Полезно для тестирования и автоматизации действий в UI.

Важно: MCP также занимают место в контекстном окне, поэтому не стоит подключать их слишком много.

Управление контекстным окном

Почему это важно: Переполненное контекстное окно приводит к ухудшению качества и "тупости" ответов модели.

Что занимает контекст:

- Системные промты (файл `claude.md`)
- Описания Skills, Agents, MCP

- История диалога
- Встроенные системные инструменты (терминал, файловый менеджер и т.д.)

Как экономить контекст:

1. **Используйте Skills** — загрузка по требованию вместо хранения полных инструкций.
2. **Используйте Subagents** — для тяжёлых задач.
3. **Минимизируйте количество MCP** — оставляйте только самые необходимые.
4. **Используйте команду** `compact` — для ручного сжатия истории.
5. **Пишите краткие и точные промты.**

Мониторинг: Можно настроить **status line** в терминале для отображения заполненности контекста в реальном времени.

Продвинутые техники работы

Agent Teams

- **Будущее разработки.** Команда полноценных агентов (бэкенд, фронтенд, БД), которые **общаются между собой** и имеют общую "записную книжку".
- Каждый агент обладает всеми функциями Claude и может сам создавать субагентов.
- Пока в бета-версии, требует тонкой настройки.

Workflow 3 (Параллельная работа)

- **Проблема:** Несколько открытых окон Claude в одном проекте конфликтуют при редактировании одних и тех же файлов.
- **Решение (Skill Work 3):** Каждому окну создаётся **изолированная копия проекта**. Агенты работают в своих копиях, не мешая друг другу.

- **Итог:** По завершении работы изменения из всех копий **мерджатся** (объединяются) в основной проект, конфликты разрешаются автоматически.
- **Результат:** Значительное ускорение разработки за счёт параллельного выполнения разных частей задачи.

Практика: создание простого сайта

Задача: Создать сайт с игрой «Крестики-нолики» в разных стилях.

Процесс:

1. Открыть несколько вкладок с Claude Code в VS Code.
2. В каждой включить **Plan Mode**.
3. Дать задание с разным описанием стиля (например, «тёмный хакерский стиль», «морская тематика», «чистый белый рай»).
4. Утвердить план, включить **Bypass Permissions**.
5. Claude параллельно создаст три разных сайта в отдельных папках.
6. Выбрать понравившийся вариант и доработать (например, добавить игру с ИИ, адаптивность).

Возможные доработки:

- Добавить референсы для точного следования дизайну.
- Попросить сделать адаптивную верстку под мобильные устройства.
- Задеплоить на сервер (Claude может сделать это самостоятельно при наличии доступа).

Продвинутые команды и настройки

Полезные команды:

- `/compact` — сжимает историю диалога в краткое изложение, чтобы освободить контекстное окно.
- `/btw` (только в терминале) — позволяет «шепнуть» вопрос Claude, пока он выполняет основную задачу, не прерывая её.
- `/init` — создаёт файл `claude.md` с описанием текущего проекта.
- `/model` — смена модели (например, на более дешёвую Claude Sonnet при исчерпании лимитов).

Глобальная папка `.claude`:

- Находится в домашней директории пользователя.
- Содержит все глобальные настройки: `claude.md`, `rules`, `settings.json`, `skills` и т.д.
- Позволяет кастомизировать поведение Claude под свои нужды.

Советы и лучшие практики

1. **Всегда начинайте с `plan mode`.** Сначала доведите план задачи до идеала, только потом приступайте к написанию кода.
2. **Просите Claude проверять свою работу.** Цикл обратной связи может **в три раза** повысить качество результата.
3. **Создавайте Skills и Subagents** для ситуаций, где Claude регулярно "спотыкается".
4. **Работайте параллельно.** Используйте несколько окон (или Workflow 3) для одновременной работы над разными частями проекта.
5. **Не бойтесь просить Claude Code о помощи в настройке.** Не нужно вручную редактировать JSON-файлы — просто опишите задачу чату.
6. **Начинайте с расширения в VS Code**, оно дружелюбнее для новичков.
7. **Экспериментируйте с голосовым вводом** для более естественного взаимодействия.
8. **Следите за лимитами токенов** и выбирайте подписку соответственно вашей активности.

9. **Используйте правила (Rules)** для структурирования промптов и привязки их к конкретным задачам.

Вывод: Claude Code — мощный инструмент, который меняет подход к созданию программ и автоматизации. Он позволяет даже новичкам реализовывать рабочие проекты, выступая в роли персонального программиста-ассистента. **Skills, Subagents и MCP** — ключевые инструменты для эффективной автоматизации и расширения возможностей, а **управление контекстом** — фундаментальный навык для стабильной и качественной работы.
