СУБМАРИНА

школа нейросетей

CLAUDE.md

разбор AI-конфига Карпатого

65 строк текста. 143 000 звёзд на GitHub.

Как один файл поменял поведение Claude Code.

143k	65	43 → 3%
звёзд	строк	ошибок

АНДРЕЙ КАРПАТЫЙ · ЯНВАРЬ 2026

Самый популярный AI-конфиг в интернете — CLAUDE.md

65 строк текста — и Claude Code перестал делать то, что раздражало всех разработчиков. Вырос из твита Карпатого, превратился в файл, который скопировали сотни тысяч раз.

143k	65	43→3%
звёзд на GitHub	строк в файле	процент ошибок

// 4 ПРАВИЛА ИЗ ФАЙЛА

Что Карпатый понял про кодинг с ИИ

01

Think Before Coding

Думай перед кодом

Don't assume. Don't hide confusion. Surface tradeoffs.

— Озвучивай допущения явно. Не угадывай молча — спроси.

— Если есть несколько трактовок задачи — покажи их, не выбирай тихо одну.

— Если есть более простой подход — скажи. Предложи альтернативу.

— Если что-то непонятно — остановись, назови что именно, задай вопрос.

02

Simplicity First

Сначала простота

Minimum code that solves the problem. Nothing speculative.

— Никаких фич сверх того, что попросили.

— Никаких абстракций для кода, который используется один раз.

— Никакой «гибкости» и «настраиваемости», которую не просили.

— Написал 200 строк — а можно было 50? Перепиши.

03

Surgical Changes

Хирургические правки

Touch only what you must. Clean up only your own mess.

— Не «улучшай» соседний код, комментарии или форматирование.

— Не рефактори то, что не сломано.

— Соответствуй существующему стилю, даже если ты бы сделал иначе.

— Тест: каждая изменённая строка должна напрямую следовать из задачи.

04

Goal-Driven Execution

Выполнение цели

Define success criteria. Loop until verified.

— Определи «готово» до того, как начать. Не «сделай так, чтобы работало».

— «Добавь валидацию» → «напиши тесты для неверных входных данных, потом сделай так, чтобы они прошли».

— «Исправь баг» → «воспроизведи его тестом, потом убери».

— Чёткий критерий успеха = можешь работать самостоятельно без постоянных уточнений.

// ИЗМЕРЕННЫЙ ЭФФЕКТ

Что изменилось после CLAUDE.md

43%
ошибок без файла

→

3%
ошибок с CLAUDE.md

// ОРИГИНАЛЬНЫЙ ФАЙЛ

CLAUDE.md целиком

65 строк · github.com/multica-ai/andrej-karpathy-skills

```
# CLAUDE.md
Behavioral guidelines to reduce common LLM coding mistakes. Merge with project-specific instructions as needed.
**Tradeoff:** These guidelines bias toward caution over speed. For trivial tasks, use judgment.

## 1. Think Before Coding
**Don't assume. Don't hide confusion. Surface tradeoffs.**
Before implementing:
  - State your assumptions explicitly. If uncertain, ask.
  - If multiple interpretations exist, present them — don't pick silently.
  - If a simpler approach exists, say so. Push back when warranted.
  - If something is unclear, stop. Name what's confusing. Ask.

## 2. Simplicity First
**Minimum code that solves the problem. Nothing speculative.**
  - No features beyond what was asked.
  - No abstractions for single-use code.
  - No "flexibility" or "configurability" that wasn't requested.
  - No error handling for impossible scenarios.
  - If you write 200 lines and it could be 50, rewrite it.
Ask yourself: "Would a senior engineer say this is overcomplicated?" If yes, simplify.

## 3. Surgical Changes
**Touch only what you must. Clean up only your own mess.**
When editing existing code:
  - Don't "improve" adjacent code, comments, or formatting.
  - Don't refactor things that aren't broken.
  - Match existing style, even if you'd do it differently.
  - If you notice unrelated dead code, mention it — don't delete it.
When your changes create orphans:
  - Remove imports/variables/functions that YOUR changes made unused.
  - Don't remove pre-existing dead code unless asked.
The test: Every changed line should trace directly to the user's request.

## 4. Goal-Driven Execution
**Define success criteria. Loop until verified.**
Transform tasks into verifiable goals:
  - "Add validation" → "Write tests for invalid inputs, then make them pass"
  - "Fix the bug" → "Write a test that reproduces it, then make it pass"
  - "Refactor X" → "Ensure tests pass before and after"
For multi-step tasks, state a brief plan:
```
1. [Step] → verify: [check]
2. [Step] → verify: [check]
3. [Step] → verify: [check]
```
Strong success criteria let you loop independently. Weak criteria ("make it work") require constant clarification.
---
**These guidelines are working if:** fewer unnecessary changes in diffs, fewer rewrites due to overcomplication, and clarifying questions come before implementation.
```

Создавай сайты, сервисы и приложения нейросетями — без кода

На мастер-классе узнаешь:

01
Как перестать откладывать идеи и начать их реализовывать
Из «надо бы сделать» — к реальному результату без разработчиков.

02
Как собрать свой первый сайт, бота или сервис
Понятная логика: как это работает и что нужно делать.

03
Какие задачи вы сможете закрывать
Где это применимо в жизни и работе.

04
Какие сервисы использовать и с чего начать
Ориентир в инструментах: что нужно на старте.

05
Как применять это в работе и выйти на доход
Где этот навык уже используется и как встроить его в работу.

Зарегистрироваться на мастер-класс