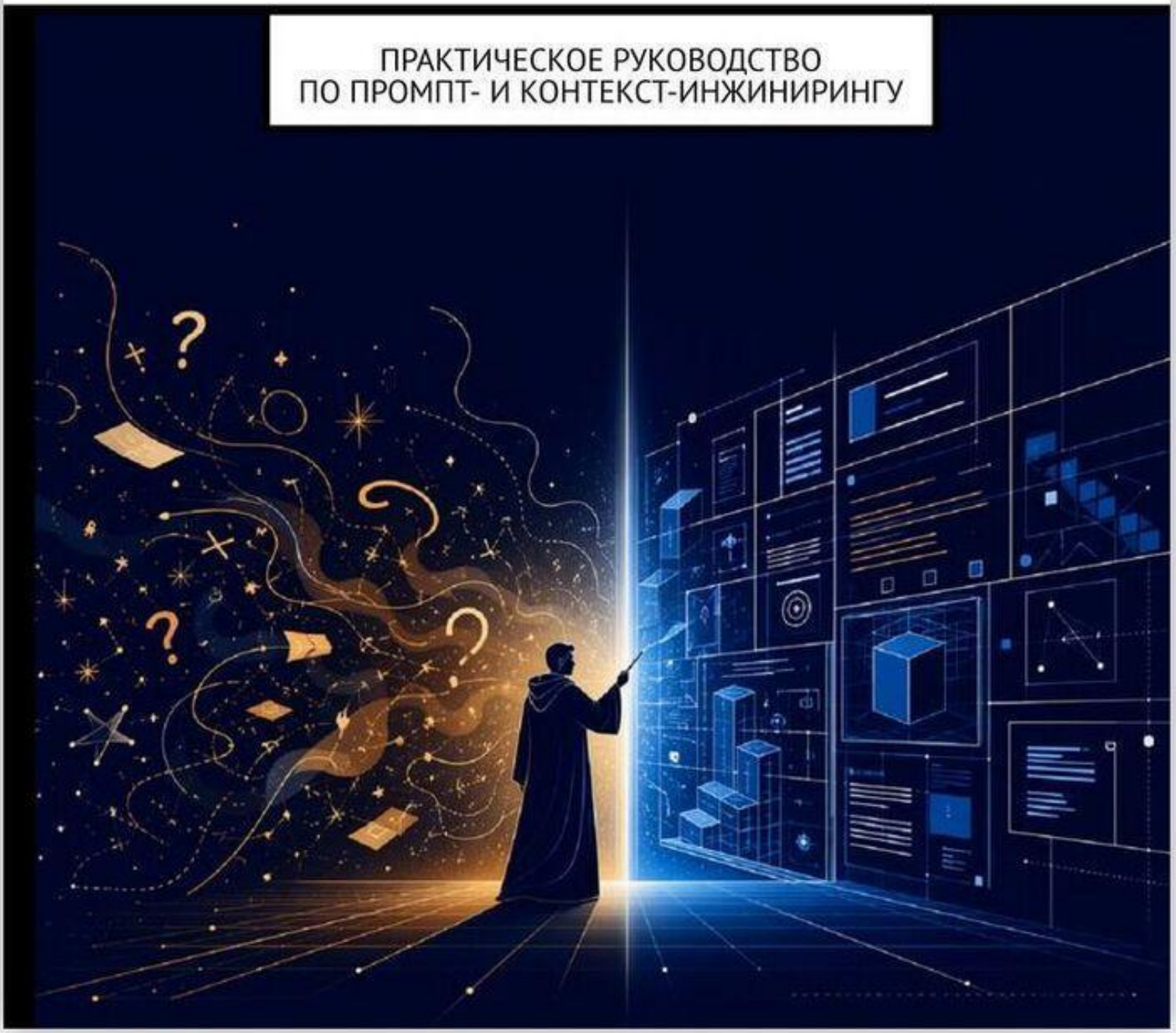
12+

АНДРЕЙ ЛАДЫГИН

Мастер ИИ-запросов

ПРАКТИЧЕСКОЕ РУКОВОДСТВО
ПО ПРОМПТ- И КОНТЕКСТ-ИНЖИНИРИНГУ



Андрей Ладыгин

Мастер ИИ-запросов. Практическое руководство по промпт- и контекст-инжинирингу

«Издательские решения»

Ладыгин А.

Мастер ИИ-запросов. Практическое руководство по промпт- и контекст-инжинирингу / А. Ладыгин — «Издательские решения»,

Промпт — это не заклинание, а спецификация. Книга для маркетологов, аналитиков, HR, преподавателей и руководителей, которые хотят получать от ИИ предсказуемый результат, а не играть в рулетку. Семь компонентов промпта, управление контекстом, отладка, измерение, защита от prompt injection — и 30 готовых сценариев с шаблонами. Плюс 7-дневный интенсив, чтобы применить всё на своих задачах уже на этой неделе.

© Ладыгин А.
© Издательские решения

Содержание

Введение 6
Для кого эта книга 7
Как устроена книга 8
Сквозной тезис 9
Как читать 10
ЧАСТЬ I. Фундамент 11
Глава 1. Конец эпохи «магических слов» 11
Как выглядела «магия» и почему она работала 11
Что изменилось 11
Три режима отказа 11
Промпт — это спецификация 12
Что заменить в старом арсенале 12
Один промпт — три модели 13
Глава 2. Как «думает» LLM 14
Модель не отвечает — она достраивает 14
Тест на человечность 16
Практикум: два эксперимента 17
Один промпт — три модели 17
ЧАСТЬ II. Анатомия промпта 18
Глава 3. Структура, которая работает у всех трёх лабораторий 18
Почему структура важнее слов 18
Восьмой компонент: декомпозиция 19
Компонент, зачем и пример 20
Практикум: собрать по скелету 21
Один промпт — три модели 21
Глава 4. Цель + Ограничения + Форма: GRASP-подход 23
GRASP: пять компонентов контракта 23
Structure: формат как контроль 24
Parameters: ограничения, которые работают 24
Шаблон GRASP 25
Конец ознакомительного фрагмента. 27

Мастер ИИ-запросов
Практическое руководство по промпт- и контекст-инжинирингу

Андрей Ладыгин

Дизайнер обложки ИИ GPT Image 2

© Андрей Ладыгин, 2026
© ИИ GPT Image 2, дизайн обложки, 2026

ISBN 978-5-0071-7316-2
Создано в интеллектуальной издательской системе Ridero

Введение

Эта книга — не сборник промптов. Сборники устаревают за три месяца: меняются модели, меняются формулировки, то, что работало вчера, сегодня даёт мусор.

Эта книга — про систему. Про то, как думать о задаче, прежде чем писать промпт. Как собирать спецификацию вместо заклинания. Как управлять контекстом, отлаживать, измерять и защищать результат. Эти принципы переносятся между моделями и версиями — потому что они про то, как формулировать задачу, а не про то, какие слова подставить.

Для кого эта книга

Для профессионалов, которые не пишут код, но работают с текстом, данными и людьми: маркетологов, аналитиков, HR, преподавателей, руководителей. А также для начинающих разработчиков, которым нужен системный подход без математики машинного обучения.

Вы уже умеете «поговорить» с ChatGPT, Claude или Gemini. Вы получаете ответы — иногда полезные, иногда бесполезные, и не всегда понимаете, почему разница. Эта книга объясняет почему.

Как устроена книга

Шесть частей, четырнадцать глав. Логика простая: **понять → спроектировать → применить → измерить → защитить → внедрить**.

Часть I. Фундамент (главы 1—2) — почему промпт перестал быть заклинанием и как модель «видит» ваш текст.

Часть II. Анатомия промпта (главы 3—5) — из каких компонентов собирается рабочий промпт и какие техники действительно работают.

Часть III. От промпта к контексту (главы 6—8) — почему ответы «дрейфуют», как читать изображения и как проектировать агентов.

Часть IV. Измерение и отладка (главы 9—10) — как перестать играть в рулетку и как доказать, что стало лучше.

Часть V. Безопасность и этика (главы 11—12) — prompt injection и ответственность за людей, на которых влияет ваш промпт.

Часть VI. Практикум (главы 13—14) — тридцать готовых сценариев и семидневный интенсив.

Сквозной тезис

Размытая цель + неполный контекст = уверенная галлюцинация.

Модель не сомневается. Она достраивает. Если вы не задали контракт — она заполнит пробелы своей фантазией, правдоподобно и быстро. Вся книга — про то, как задавать контракт.

Как читать

Если вы читаете впервые — по порядку. Главы ссылаются друг на друга, и термины вводятся последовательно.

Если вам нужен инструмент прямо сейчас — откройте главу 13, найдите свою профессию, возьмите один сценарий. Но потом вернитесь к главам 3—5: без теории шаблон работает как заклинание.

В конце каждой главы — артефакт: шаблон, чек-лист или упражнение. Он предназначен для того, чтобы его применили сегодня, а не «когда-нибудь».

ЧАСТЬ I. Фундамент

Глава 1. Конец эпохи «магических слов»

В 2023 году по соцсетям ходил промпт на четыреста слов. Он обещал «идеальный ответ всегда». Люди копировали его, вставляли в ChatGPT, получали невнятный текст и делали вывод: «нейросети не работают» или «работают только у блогеров».

К 2026 году сменилось два поколения моделей. Промпт всё ещё лежит в закладках, а результат — тот же. Проблема не в модели и не в закладке. Промпт обещал магию, а нужна была спецификация.

Как выглядела «магия» и почему она работала

Модели 2022—2023 годов были слабыми собеседниками. Они теряли нить в длинном запросе и легко сбивались на общие слова. Чтобы получить внятный ответ, приходилось их «уговаривать»: добавлять «пожалуйста», назначать роль («ты — эксперт по маркетингу»), просить «думать шаг за шагом». Это работало. Не потому, что слова обладали силой, а потому, что они компенсировали слабость модели — подталкивали её удерживать контекст и структуру.

Отсюда родилась целая культура «мега-промптов»: длинных заклинаний на 300—500 слов, которые обещали превратить любую модель в эксперта. Сборники таких промптов продавались как самоучители. Проблема была в том, что приёмы работали в узком диапазоне задач и моделей. Стоило сменить модель или усложнить задачу — и конструкция рассыпалась.

Что изменилось

К 2026 году модели научились рассуждать внутри себя. GPT-5, Claude 4, Gemini 3 (версии на момент написания) тратят «внутренние» токены на построение цепочки шагов до того, как выдать ответ. То, что раньше приходилось выпрашивать фразой «думай шаг за шагом», теперь происходит по умолчанию.

Практика здесь неоднородна: reasoning-модели часто справляются с декомпозицией сами, но результат зависит от формулировки. Поэтому правило простое: не добавляйте «думай шаг за шагом» автоматически. Проверьте, нужен ли этот приём именно вашей модели и именно вашей задаче.

Слабые модели требовали уговоров. Сильные — требуют точности. «Пожалуйста» больше не улучшает ответ. А вот отсутствие аудитории, формата и критериев успеха — ухудшает, и сильно.

Разница между уговорами и точностью проявляется в одном и том же запросе. Заклинание подталкивает модель «быть хорошей». Спецификация объясняет, что значит «хорошо» в вашем случае. Первое работает как шум, второе — как инструкция. Это и есть главный сдвиг: от попытки повлиять на модель к попытке описать задачу.

Три режима отказа

Промпт-заклинание ломается предсказуемо. Есть три типовых сценария.

Размытая цель. «Сделай красиво» — это не задача, а пожелание. Модель не знает, что для вас «красиво»: коротко или подробно, строго или дружелюбно, для новичка или для эксперта. Она выберет что-то среднее и уверенно выдаст результат.

Неполный контекст. Модель не знает вашу аудиторию, ограничения, предыдущие решения. Если карта неполная, она достроит её сама — правдоподобно и неверно. Так рождается галлюцинация: не «баг», а следствие пробела в спецификации.

Отсутствие формата. Если не сказано, в каком виде вернуть ответ, модель выберет удобный ей. Получите пять абзацев прозы там, где нужна таблица. Или таблицу там, где нужно письмо клиенту.

Все три режима сходятся в одной точке: **размытая цель + неполный контекст = уверенная галлюцинация**. Модель не сомневается — она достраивает.

Ошибка новичка. Скопировать вирусный «мега-промпт» из соцсетей, вставить в свою задачу без адаптации и удивиться, что результат не совпал с обещаниями автора. Промпт писался под чужую задачу, чужую модель и чужую аудиторию. Без адаптации он превращается в красивый мусор.

Промпт — это спецификация

Спецификация в инженерии — это документ, который описывает, что нужно сделать, при каких условиях и как проверить результат. Хорошая спецификация не уговаривает исполнителя, а задаёт контракт.

Промпт работает так же. Модель — не оракул, а исполнитель. Ей нужны: цель, контекст, ограничения и формат результата. Если контракт неполный — исполнитель заполнит пробелы своей фантазией. Если полный — вы получите предсказуемый результат, который можно проверить и воспроизвести.

Это сдвиг, который произошёл в 2025—2026 годах. Центр навыка сместился от «поиска волшебных слов» к контекст-инжинирингу — умению собрать полную картину задачи и удержать её в промпте.

Промпт как код: три навыка

Раз промпт — спецификация, с ним можно работать как с кодом. Три навыка, которых нет у большинства пользователей.

Версионирование. Хранить рабочие промпты и их версии, а не терять удачные формулировки в истории чата.

Отладка. Менять по одному компоненту за раз и понимать, что именно улучшило результат. Без этого нельзя воспроизвести успех.

Измерение. Оценивать результат по заранее заданным критериям, а не по принципу «нравится / не нравится».

На практике это выглядит скромно: папка с файлами промптов, где у каждого есть номер версии и дата. Таблица, в которой вы отмечаете, что изменили и что получили. Набор из трёх критериев, по которым оцениваете ответ. Ничего из этого не требует инструментов — только привычки. Но именно эти три привычки отделяют человека, который «нащупал работающий промпт», от человека, который понимает, почему он работает.

Что заменить в старом арсенале

«Пожалуйста», вежливость — когда работал: 2022—2023, слабые модели. сейчас: влияние не подтверждено. замена: убрать, не тратить токены.

«Ты — эксперт в X» — когда работал: 2022—2023. сейчас: иногда задаёт регистр. замена: конкретная роль + аудитория.

«Думай шаг за шагом» — когда работал: 2022—2024. сейчас: reasoning-моделям не нужно, иногда вредит. замена: не добавлять автоматически.

«Дай идеальный ответ» — когда работал: 2023. сейчас: не работает. замена: задать критерии успеха.

«Мега-промпт» на 400 слов — когда работал: 2023. сейчас: перегружает контекст. замена: собрать по компонентам под задачу.

Чек-лист: признаки промпта-заклинания

Проверьте свой промпт по семи пунктам. Если совпало три и больше — переписывайте.

1. Есть слова «идеальный», «лучший», «качественный» без критериев.

2. Не указан читатель результата.

3. Не задан формат вывода.

4. Есть «думай шаг за шагом» по привычке.

5. Промпт скопирован из соцсетей и не адаптирован.

6. Больше трёхсот слов, но меньше трёх конкретных фактов о задаче.

7. Непонятно, как проверить, что ответ хороший.

Практикум: аудит трёх своих промптов

Возьмите три запроса, которые вы реально писали модели за последнюю неделю. Оцените каждый по трём осям:

• **Цель.** Понятно ли, что именно нужно получить? (0 — нет, 1 — частично, 2 — да)

• **Контекст.** Указаны ли аудитория, ограничения, исходные данные? (0 / 1 / 2)

• **Формат.** Задан ли вид результата? (0 / 1 / 2)

Максимум — 6 баллов. Всё, что ниже 4, — кандидат на переписывание. Запишите результат: в конце книги вы вернётесь к этому замеру и сравните.

Один промпт — три модели

Возьмите классическое заклинание: «Ты — лучший эксперт. Думай шаг за шагом. Дай идеальный ответ». Прогоните его в GPT-5, Claude 4 и Gemini 3 с одной и той же задачей — например, «составь план письма клиенту».

Результат покажет разницу: reasoning-модели сами достроят шаги, которые раньше приходилось выпрашивать. Но цели, контекста и формата в этом промпте по-прежнему нет — и каждая модель заполнит пробелы по-своему. Кто-то напишет коротко, кто-то развёрнуто, кто-то не в том тоне. И вы не сможете сказать, какая версия «правильная», потому что критериев нет.

Чтобы понять, почему заклинания перестали работать, надо заглянуть внутрь — не в математику, а в то, как модель «видит» ваш текст. Этим займёмся в следующей главе.

Глава 2. Как «думает» LLM

В понедельник вы получили от модели отличный ответ. В среду задали тот же вопрос — и получили другой. Модель «сломалась»? Нет. Вы просто не знаете, что происходит между вашим текстом и ответом. Эта глава — про механику: как модель видит промпт. Не математика, а минимум, без которого все дальнейшие приёмы работают наугад.

Модель не отвечает — она достраивает

LLM не «понимает вопрос» и не «ищет ответ». Она предсказывает, каким текстом вероятнее всего продолжится ваш промпт. Ваш текст — начало документа. Модель дописывает то, что логично его продолжает, — по статистике, собранной на огромном корпусе текстов.

Это меняет взгляд на промпт. Вы не задаёте вопрос — вы задаёте начало документа, который модель завершит. Если начало похоже на «пожалуйста, помоги с маркетингом», модель допишет типичный ответ на типичную просьбу. Если начало похоже на «ты — аналитик, вот данные, вот аудитория, вот формат», модель допишет типичный ответ аналитика на конкретную задачу.

Простая проверка: прочитайте свой промпт и спросите — «что логично продолжает этот текст?». Если ответ «что угодно» — вы получите что угодно. Это не мистика, а следствие механики.

Токены: из чего модель видит текст

Модель не видит буквы и слова. Она видит токены — кусочки текста, примерно по 2—4 символа или по одному короткому слову. Английское слово — обычно один токен. Русское слово — часто два-три токена, потому что кириллица в большинстве токенизаторов представлена менее эффективно. Грубая оценка для русского: один токен — около 1—2 символов, но точная цифра зависит от модели и токенизатора.

Зачем это знать на практике.

Длина стоит. Каждый токен — это вычисление. Чем длиннее промпт, тем дольше ответ и тем больше вы платите, если работаете через API.

Длина имеет предел. Контекстное окно модели измеряется в токенах: 128 тысяч у большинства современных моделей, 200 тысяч у Claude, до миллиона у отдельных версий Gemini (на момент написания; уточняйте в документации вендора). Предел не «сколько слов влезет», а «сколько токенов модель удержит в голове одновременно».

Смешение языков не бесплатно. Промпт на русском с английскими терминами даёт смешанную токенизацию. Это не ошибка, но учитывайте при расчёте лимитов.

Контекстное окно: стол, а не бездонный колодец

Контекстное окно — это рабочий стол модели. Всё, что на нём лежит, модель «видит» при генерации ответа. Всё, что не влезло, — не видит вовсе.

Ошибка новичка — думать, что большое окно решает проблему длинного промпта. Не решает. В ряде исследований описан эффект «потерянной середины»: информация в начале и в конце длинного контекста запоминается лучше, чем в середине. Модель как бы «пробегает глазами» по краям и хуже удерживает центр.

Практический вывод — **карта контекстного окна**:

НАЧАЛО: роль, главная
СЕРЕДИНА: данные, контекст, документы
КОНЕЦ: формат, напоминание о главном

Что куда класть:

• **Начало:** роль модели, главная задача, критерии успеха.

• **Середина:** данные, которые модель должна использовать, — документы, примеры, факты.

• **Конец:** формат вывода, напоминание о самом важном ограничении.

Если инструкция «не выходи за 150 слов» стоит в начале длинного промпта, а между ней и ответом — три страницы данных, шанс, что модель её проигнорирует, выше. Повтор в конце работает не как «напоминание для человека», а как смещение фокуса в зону, где модель удерживает внимание лучше.

Температура и top-p: ручки, которые реально влияют

Когда вы нажимаете «Отправить», модель не выдаёт один и тот же ответ всегда. Она выбирает следующее слово вероятностно. Две настройки управляют этой вероятностью.

Temperature (температура) — «степень свободы» выбора. Низкая (0—0.3) — модель выбирает самые вероятные слова, ответ стабильный и предсказуемый. Высокая (0.7—1.0 и выше) — модель допускает менее вероятные слова, ответ разнообразнее, но и риск «уехать» выше.

Бытовая аналогия: температура — это как выбор блюда в меню. При 0 модель всегда берёт самое популярное. При 1 — иногда экспериментирует.

Top-p — «широта меню». Модель рассматривает только те слова, которые вместе набирают заданную вероятность. Top-p = 0.9 значит: «выбирай из слов, которые в сумме дают 90% вероятности». Top-p = 1.0 — «рассматривай всё».

Temperature — степень случайности выбора. Низкое значение: стабильно, предсказуемо. Высокое: разнообразно, креативно. Когда использовать: 0—0.3 для фактов, отчётов, кода; 0.7—1.0 для брейншторма.

Top-p — широта выбора слов. Низкое значение: узкий круг, безопасно. Высокое: широкий круг, рискованно. Когда использовать: 0.9—0.95 — рабочий диапазон по умолчанию.

Max tokens — максимальная длина ответа. Низкое значение: короткий ответ. Высокое: длинный ответ. Когда использовать: ставьте явно, чтобы не получить обрубок.

Практическое правило: для задач с проверяемым результатом (отчёт, резюме, код) — низкая температура. Для задач на идеи (брейншторм, варианты заголовков) — выше. Для большинства рабочих задач 0.2—0.5 — разумный старт.

Почему «чем длиннее промпт, тем лучше» — ложь

«Мега-промпты» на 500 слов обещают всеобъемлющий результат. На деле длинный промпт ломается по трём причинам.

Противоречия. Чем больше инструкций, тем выше шанс, что две из них конфликтуют. «Пиши подробно» и «уложись в 100 слов» — модель выберет что-то среднее и не выполнит ни то, ни другое.

Потеря середины. Чем длиннее промпт, тем больше информации оказывается в зоне, которую модель удерживает хуже.

Перегрузка внимания. Каждая инструкция «размывает» фокус на главной. Промпт из тридцати правил хуже, чем промпт из пяти, — потому что модель не знает, какое из тридцати важнее.

Правило простое: **промпт должен быть достаточно длинным, чтобы задать контракт, и достаточно коротким, чтобы не размывать его**. Если можно убрать слово без потери смысла — уберите. Если можно заменить абзац таблицей — замените.

Вот пример сжатия. Было: «Ты — опытный специалист по маркетингу. Пожалуйста, постарайся написать качественный текст для нашего сайта, чтобы он был понятным, интересным и убедительным, и при этом не слишком длинным, а также учитывал нашу аудиторию, которая состоит из владельцев малого бизнеса». Стало: «Перепиши текст для владельца малого

бизнеса. До 120 слов. Убери канцелярит. Первое предложение — главная выгода». Смысла больше, слов в четыре раза меньше.

Reasoning-модели: другой режим работы

Модели 2022—2023 годов отвечали сразу: получили промпт — выдали следующий токен. Модели 2025—2026 годов (GPT-5, Claude 4, Gemini 3 — версии на момент написания) умеют тратить «внутренние» токены на рассуждение перед ответом.

Что это меняет для промпта.

Классический CoT («думай шаг за шагом») часто избыточен. Модель и так думает. Явная инструкция может дублировать внутренний процесс или, хуже, навязывать ему чужую структуру.

Практика неоднородна. На одних задачах CoT помогает и reasoning-модели (особенно когда нужно показать ход рассуждения пользователю или проверить логику), на других — вредит, потому что модель тратит токены на «проговаривание» того, что уже сделала внутри. Универсального правила нет — проверяйте на своей задаче.

Практическая рекомендация. По умолчанию не добавляйте «думай шаг за шагом» в промпт для reasoning-модели. Если результат не устраивает — попробуйте добавить и сравните. Не наоборот: не начинайте с CoT, потому что «так делали в 2023».

Галлюцинация — не баг, а следствие пробела

Галлюцинация — это когда модель уверенно выдаёт неверный факт, несуществующий источник или вымышленную цитату. Принято думать, что это «баг» модели, который когда-нибудь исправят. Точнее смотреть иначе.

Модель достраивает документ. Если в промпте нет данных — она достраивает правдоподобно, по статистике. Это не ложь и не сбой: это ровно та работа, которую вы ей поручили. Пробел в спецификации модель заполняет сама.

Отсюда практический вывод: **лечить галлюцинацию промптом «не выдумывай» — почти бесполезно**. Модель не различает «выдуманное» и «правдоподобное» так, как это делаете вы. Работает другое: дать ей данные (контекст, документы, источники) и явно указать, что можно брать только из них. Или попросить помечать утверждения, в которых она не уверена.

Это возвращает нас к тезису из первой главы: размытая цель + неполный контекст = уверенная галлюцинация. Галлюцинация — симптом, а не болезнь. Болезнь — пробел в спецификации.

Тест на человечность

Есть простая практика, которую используют инженеры промптов. Прежде чем отправлять промпт модели, покажите его коллеге — без контекста задачи. Если коллега запутался, не понял, что от него хотят, или задал уточняющий вопрос — модель запутается тоже. Модель обрабатывает текст похожим образом: ей тоже нужен контекст, чтобы не достраивать наугад.

Пять вопросов, которые стоит задать коллеге (и себе):

1. Понятно ли, что именно нужно сделать?

2. Понятно ли, для кого результат?

3. Понятно ли, в каком виде его вернуть?

4. Понятны ли ограничения — что нельзя, что обязательно?

5. Хватает ли данных, чтобы выполнить задачу, не додумывая?

Если хотя бы на один вопрос ответ «нет» — модель тоже ответит «нет». Только молча.

Ошибка новичка. Получить плохой ответ и начать «объяснять модели получше» — добавлять абзацы, уточнения, эмоции. Модель не «недопоняла»: у неё не было контекста и

критериев. Лечится не длиной объяснения, а конкретикой: кто читатель, что считать хорошим результатом, в каком формате вернуть.

Практикум: два эксперимента

Эксперимент 1. Тест на человечность. Возьмите свой рабочий промпт. Покажите коллеге без пояснений. Попросите ответить на пять вопросов выше. Зафиксируйте, на каких вопросах коллега запнулся. Это карта пробелов в вашем промпте.

Эксперимент 2. Температура. Возьмите одну задачу — например, «придумай три заголовка для статьи о промптах». Прогоните её дважды: при temperature 0.2 и при temperature 0.9. Сравните: насколько ответы различаются между собой; есть ли в «горячем» варианте идеи, которых нет в «холодном»; есть ли в «горячем» варианте ошибки или странности.

Запишите вывод: при какой температуре вы будете решать эту задачу в следующий раз.

Один промпт — три модели

Возьмите задачу с проверяемым результатом: «составь список из пяти вопросов для интервью с кандидатом на позицию аналитика». Прогоните при temperature 0.2 и 0.9 в трёх моделях.

Что вы увидите: при 0.2 все три модели дадут похожие списки, различия — в нюансах; при 0.9 разброс вырастет — кто-то предложит нестандартные вопросы, кто-то странные; стабильность при низкой температуре у всех высокая, но не абсолютная.

Вывод не «какая модель лучше», а «какая температура подходит задаче». Для интервью-вопросов 0.2 разумнее: вам нужны предсказуемые, проверяемые формулировки, а не креатив.

Поняв, как модель видит текст, можно собирать промпт из деталей — как конструктор, а не как заклинание. Этим займёмся в следующей главе.

ЧАСТЬ II. Анатомия промпта

Глава 3. Структура, которая работает у всех трёх лабораторий

Возьмите два промпта. В первом — всё, что вы хотите сказать модели, свалив в один абзац. Во втором — то же самое, но разложенное по блокам: роль, задача, контекст, формат. Отправьте оба в одну модель. Первый даст что-то среднее и невнятное. Второй — предсказуемый результат, который можно проверить. Содержание одно, структура разная, качество — несопоставимое. Эта глава — про то, как собрать структуру, которая работает у всех трёх ведущих лабораторий: Anthropic, OpenAI и Google.

Почему структура важнее слов

В предыдущей главе мы разобрались, как модель видит текст: она достраивает документ по вероятности. Структура влияет на то, каким документ «кажется» модели. Абзац без разметки — это сплошной поток, в котором модели труднее выделить, что важнее. Размеченный промпт — это документ с явными зонами: инструкция, данные, формат.

Представьте, что вы даёте задание стажёру. Можно сказать: «Ну, посмотри там, сделай что-нибудь с отчётом, чтобы было понятно». А можно: «Ты — аналитик. Вот отчёт. Сделай из него выжимку на одну страницу для руководителя. Формат: три пункта, каждый — одно предложение». Во втором случае стажёр сделает предсказуемо. В первом — как получится.

Модель работает похоже. Структура промпта — это не украшение, а способ сделать задачу исполнимой.

Семь компонентов: консенсус трёх лабораторий

Anthropic, OpenAI и Google выпускают руководства по промпт-инжинирингу. Формулировки в них различаются, но набор компонентов, из которых должен состоять рабочий промпт, совпадает. Семь элементов, которые повторяются у всех трёх.

1. Роль или персона. Кто модель в этой задаче. Не «ты — лучший эксперт», а конкретно: «ты — редактор научно-популярных текстов», «ты — аналитик данных в маркетинге», «ты — методист, пишущий для студентов первого курса». Роль задаёт регистр и словарь. Она не делает модель умнее, но помогает выбрать тон.

2. Контекст задачи. Что важно знать до начала работы. Аудитория, исходные данные, предыдущие решения, ограничения. Контекст — это фундамент, на котором модель строит ответ. Без него она достраивает сама.

3. Чёткие инструкции. Что именно сделать. Одна задача — один глагол действия. «Составь», «перепиши», «сравни», «найди». Если глаголов несколько, укажите порядок. Размытое «поработай с текстом» модель поймёт как угодно.

4. Последовательные шаги. Если задача сложная — разложите её на шаги в самом промпте. «Сначала выдели ключевые тезисы. Потом сгруппируй их по темам. Потом предложи структуру статьи». Шаги работают как скелет рассуждения. Для reasoning-моделей (см. гл. 2) это не всегда обязательно, но для сложных задач — полезно.

5. Few-shot примеры (обучение на примерах). Один-два примера того, что считается хорошим результатом. Не инструкция «пиши хорошо», а образец: «Вот так выглядит хорошая выжимка: [пример]». Примеры задают стандарт лучше любых прилагательных. Подробнее разберём в главе 5.

6. Формат вывода. В каком виде вернуть результат. Таблица, список, JSON, письмо, три абзаца. Если формат не задан, модель выберет удобный ей. Это не всегда совпадает с вашими ожиданиями.

7. Ограничения. Что нельзя, что обязательно, какие рамки. «Не больше 150 слов», «без вводных фраз», «только факты из предоставленного текста», «не используй слово „клиент“ — говори „пользователь“». Ограничения работают как забор: без них модель может уйти в сторону.

Это семь компонентов, которые встречаются в руководствах всех трёх лабораторий. Не потому, что они договорились, а потому, что каждый из них решает конкретную проблему: неопределённость роли, пробел в контексте, размытость задачи, сложность, отсутствие стандарта, непредсказуемость формы, размытость границ.

Восьмой компонент: декомпозиция

Семь компонентов — база. Восьмой элемент — декомпозиция: разбиение сложной задачи на управляемые подзадачи. В исследованиях 2025 года декомпозиция показывала прирост точности до 57% на конкретных задачах; на ваших данных цифра может отличаться.

Что это значит на практике. «Напиши стратегию продвижения» — задача, которую модель не может выполнить хорошо за один шаг. Слишком много переменных: аудитория, каналы, бюджет, сроки, конкуренты, метрики. Разбейте на пять подзапросов:

1. Определи три целевые аудитории и их боли.

2. Предложи по два канала на каждую аудиторию.

3. Опиши по одному формату контента на канал.

4. Составь план на первый месяц по неделям.

5. Предложи три метрики для отслеживания.

Каждый подзапрос — маленькая задача с проверяемым результатом. Сумма пяти ответов будет лучше, чем один ответ на «напиши стратегию». Не потому, что модель стала умнее, а потому, что вы перестали просить её сделать невозможное за один шаг.

Декомпозиция — восьмой компонент, который не всегда нужен, но без которого сложные задачи не решаются. Используйте её, когда задача не помещается в один ответ.

Порядок имеет значение: эффект края

Компоненты — это ингредиенты. Порядок — рецепт. Он влияет на результат не меньше, чем состав.

В главе 2 мы говорили про эффект «потерянной середины»: информация в центре длинного контекста удерживается хуже. Это значит, что **ключевые инструкции нужно ставить в начало и в конец промпта**. Начало задаёт рамку, конец — напоминает о главном.

Практическая схема для большинства задач:

```
[НАЧАЛО]
Роль + главная задача + критерии успеха
[СЕРЕДИНА]
Контекст + данные + примеры
[КОНЕЦ]
Формат вывода + ключевое ограничение (повтор)
```

Если ограничение «не больше 150 слов» стоит только в начале длинного промпта, риск, что модель его проигнорирует, выше. Повтор в конце работает не как «напоминание для человека», а как смещение фокуса в зону, где модель удерживает внимание лучше.

Порядок компонентов — не догма. Для коротких промптов (до 100 слов) он почти не влияет. Для длинных (200+ слов) — влияет сильно. Правило простое: **чем длиннее промпт, тем важнее ставить главное по краям**.

Различия лабораторий: что работает везде

Все три лаборатории рекомендуют похожие компоненты, но разметка у них разная. Это важно знать, если вы работаете с несколькими моделями.

Claude (Anthropic) — по документации Anthropic, хорошо реагирует на XML-теги (теги вида <role> … </role>). Промпт вида <role> … </role> <task> … </task> <context> … </context> для Claude читается как структурированный документ. Теги помогают модели различать зоны, особенно в длинных промптах.

GPT (OpenAI) хорошо работает с markdown-разметкой. Заголовки ###, списки, жирный шрифт — модель воспринимает markdown как естественную структуру.

Gemini (Google) хорошо реагирует на естественный язык с явными разделителями. «Сначала роль. Потом задача. Потом контекст» — работает не хуже тегов.

Что делать, если вы пишете один промпт для всех трёх. Есть универсальный минимум, который работает везде:

- **Заголовки блоками.** Даже без разметки — просто «РОЛЬ:», «ЗАДАЧА:», «КОНТЕКСТ:» — модель видит границы.
- **Пустая строка между блоками.** Разделяет зоны визуально и для модели.
- **Явные переходы.** «Теперь задача:», «Данные:», «Формат:» — работают как маркеры.
- **Повтор ключевого в конце.** Универсально для всех моделей.

Если вы работаете с одной моделью — используйте её любимую разметку. Если с несколькими — используйте заголовки блоками без специфичной разметки. Работает везде, чуть менее эффективно, чем «родная» разметка, но предсказуемо.

Если вы новичок и ещё не выбрали модель, начните с заголовков блоками и явных переходов. Это не требует знания особенностей конкретной лаборатории и переносится между моделями без переписывания. Когда освоитесь — добавите «родную» разметку той модели, с которой работаете чаще.

Шаблон «Скелет промпта»

Собрать промпт из восьми компонентов — задача, которую не хочется решать каждый раз с нуля. Поэтому — шаблон. Одна страница, которая закрывает 80% задач.

РОЛЬ: [кто модель в этой задаче — конкретно]
ЗАДАЧА: [что сделать — один глагол действия]
АУДИТОРИЯ: [кто будет читать результат]
КОНТЕКСТ: [что важно знать: данные, ограничения, предыдущие решения]
ШАГИ: [если задача сложная — разложить на 2—5 шагов]
ПРИМЕР: [один образец хорошего результата — если есть]
ФОРМАТ: [как вернуть: список, таблица, письмо, N слов]
ОГРАНИЧЕНИЯ: [что нельзя, что обязательно]

Это скелет. Не все блоки нужны в каждом промпте. Для простых задач достаточно роли, задачи, формата и одного ограничения. Для сложных — все восемь.

Правило: **если блок можно убрать без потери смысла — уберите**. Скелет — не обязательный каркас, а чек-лист того, что стоит продумать. Промпт из четырёх блоков лучше, чем промпт из восьми, где половина — «для галочки».

Компонент, зачем и пример

Роль — зачем нужен: задаёт регистр и словарь. Пример: «Ты — редактор научно-популярных текстов»

Задача — зачем нужен: убирает размытость. Пример: «Перепиши этот абзац короче, сохранив смысл»

Аудитория — зачем нужен: задаёт уровень объяснения. Пример: «Для читателей без технического бэкграунда»

Контекст — зачем нужен: убирает пробелы, из которых рождается галлюцинация. Пример: «Это фрагмент главы, до него было про токены»

Шаги — зачем нужен: разбивает сложное на управляемое. Пример: «Сначала выдели тезисы, потом сгруппируй»

Пример — зачем нужен: задаёт стандарт лучше прилагательных. Пример: «Вот так выглядит хорошая выжимка: [текст]»

Формат — зачем нужен: убирает сюрпризы в выводе. Пример: «Таблица из трёх колонок, не больше 200 слов»

Ограничения — зачем нужен: ставит забор. Пример: «Только факты из текста, без домыслов»

Что делать, если промпт не помещается в скелет

Скелет — не религия. Для некоторых задач он избыточен. Для некоторых — недостаточен. Если задача не помещается в скелет, скорее всего, её надо разбить на две (декомпозиция, восьмой компонент).

Если задача помещается, но блоки «не работают» — проверьте порядок (эффект края) и длину (не размыта ли главная инструкция). Часто проблема не в том, что компонент отсутствует, а в том, что он стоит не там или сформулирован размыто.

Ошибка новичка. Пихнуть в промпт все восемь компонентов сразу — и получить перегруженную кашу вместо ясности. Восемь блоков по три предложения — это двадцать четыре предложения инструкций. Модель теряет фокус. Лучше четыре блока по одному предложению, чем восемь по три.

Практикум: собрать по скелету

Возьмите одну свою рабочую задачу. Соберите для неё промпт по «Скелету» — заполните все восемь блоков, даже если кажется, что некоторые не нужны.

Затем:

1. Сократите: уберите блоки, без которых можно обойтись.

2. Сравните результат с вашим текущим промптом на той же задаче.

3. Оцените по трём критериям: точность (насколько результат соответствует задаче), полнота (не упущено ли важное), формат (тот ли вид).

Запишите, какие блоки реально повлияли на результат, а какие оказались «для галочки». Это ваша карта для будущих промптов.

Один промпт — три модели

Возьмите один и тот же промпт, собранный по скелету, и оформите его двумя способами: с XML-тегами и с markdown-заголовками. Прогоните оба варианта в GPT, Claude и Gemini на одной задаче — например, «составь план статьи о промптах для новичков».

Что вы увидите: Claude чуть лучше держит структуру с XML-тегами; GPT — с markdown; Gemini — примерно одинаково, но лучше реагирует на естественные переходы («Теперь задача:»).

Разница невелика, но заметна на длинных промптах. На коротких (до 100 слов) разметка почти не влияет: модель справляется с любым вариантом. Вывод: для коротких промптов — не

переусложняйте. Для длинных — используйте разметку, родную для модели, с которой работаете.

Структура — это каркас. Дальше научимся наполнять его содержанием: целью, ограничениями и форматом вывода.

Глава 4. Цель + Ограничения + Форма: GRASP-подход

Три модели, один промпт: «Сделай красиво». Первая выдала длинный торжественный текст. Вторая — короткий сухой список. Третья — что-то среднее, но не в том тоне. Все три «сделали красиво» — каждая по-своему. Проблема не в моделях. Проблема в том, что «красиво» — это не задача, а пожелание. В этой главе — как превратить пожелание в управляемый контракт, который модель может выполнить, а вы — проверить.

Почему «сделай красиво» не работает

В главе 2 мы разобрались: модель достраивает документ по вероятности. Если в промпте нет критериев, она выберет то, что статистически вероятнее для похожих запросов. «Сделай красиво» в корпусе текстов встречается рядом с десятками разных результатов: торжественные речи, короткие слоганы, витиеватые описания, сухие списки. Модель выберет что-то из этого диапазона — и угадает или не угадает.

«Красиво» — это оценка, а не задача. У оценки нет критериев, пока вы их не задали. Модель не знает, что для вас «красиво»: коротко или подробно, строго или дружелюбно, для новичка или для эксперта. Она заполнит пробел своим представлением о прекрасном.

GRASP — это способ задать критерии до того, как модель начнёт работать. Пять компонентов, которые превращают пожелание в спецификацию.

GRASP: пять компонентов контракта

GRASP — это аббревиатура. Goal, Role, Audience, Structure, Parameters. Цель, роль, аудитория, структура, параметры. Пять элементов, которые закрывают то, что «Скелет промпта» из главы 3 оставлял на усмотрение автора.

Goal (цель). Что именно должно быть сделано. Один глагол действия и один результат. «Переписать абзац короче, сохранив смысл» — цель. «Поработать с текстом» — не цель, а направление.

Role (роль). Кто модель в этой задаче. Конкретно: «редактор научно-популярных текстов», «аналитик данных в маркетинге». Роль задаёт словарь и регистр. Она не делает модель умнее, но помогает выбрать тон, соответствующий задаче.

Audience (аудитория). Кто будет читать результат. Не «люди», а конкретно: «руководитель, у которого две минуты на чтение», «студент первого курса без технического бэкграунда», «клиент, который уже знаком с продуктом». Аудитория задаёт уровень объяснения и глубину.

Structure (структура). В каком виде вернуть результат. Список, таблица, письмо, три абзаца, JSON. Структура — это выходной контракт: если её нет, модель выберет удобный ей формат.

Parameters (параметры). Ограничения и рамки. Длина, тон, что нельзя, что обязательно. Параметры — это забор вокруг задачи: без них модель может уйти в сторону.

Пять компонентов — не догма, а минимальный набор, который закрывает большинство рабочих задач. В отличие от «Скелета промпта» из главы 3, GRASP — это не чек-лист блоков, а способ думать о задаче. Скелет отвечает на вопрос «что написать в промпте». GRASP — на вопрос «что я на самом деле хочу получить».

Goal: как отличить цель от пожелания

Цель — это то, что можно проверить. Пожелание — то, что нельзя.

«Сделай качественно» — пожелание. Как проверить «качественно»? Никак. «Уложись в 150 слов, сохрани три ключевых тезиса, убери вводные конструкции» — цель. Можно проверить: посчитать слова, найти тезисы, проверить, остались ли вводные.

Простой тест: **можете ли вы, получив результат, сказать «да» или «нет» без объяснений?** Если можете — это цель. Если начинаете «ну, в целом неплохо, но…» — это пожелание.

Практика: перед тем как писать промпт, сформулируйте цель одним предложением. Если не получается — задача ещё не продумана. Разбейте её (декомпозиция, гл. 3) или уточните, что именно вам нужно.

Role и Audience: два разных фокуса

Роль и аудитория — не одно и то же, хотя их часто путают.

Роль — это позиция, с которой модель смотрит на задачу. «Ты — редактор» — модель будет править текст как редактор: искать длинноты, канцелярит, несоответствия. «Ты — маркетолог» — модель будет искать, как текст работает на продажу: цепляет ли, убеждает ли, есть ли призыв к действию.

Аудитория — это тот, на кого работает результат. «Для руководителя» — коротко, по делу, с выводами вперёд. «Для клиента» — с объяснением выгод, без технических деталей. «Для коллег» — с терминами, без разжёвывания.

Одна и та же задача с разными ролями и аудиториями даёт разные результаты. «Перепиши письмо» — что переписать? Если роль — редактор, а аудитория — клиент: убрать канцелярит, сделать дружелюбнее. Если роль — юрист, а аудитория — коллеги: уточнить формулировки, убрать двусмысленности. Один и тот же текст, разные результаты.

Практическое правило: **роль задаёт, как модель думает; аудитория задаёт, для кого результат**. Обе нужны, если задача сложнее «перескажи текст».

Structure: формат как контроль

Формат вывода — это не украшение, а инструмент контроля. Если формат задан, вы можете проверить, соблюдён ли он. Если не задан — модель выберет сама, и вы будете спорить с результатом на вкус.

Три уровня контроля формата.

Свободный формат. «Напиши текст» — модель выберет сама. Подходит для брейншторма, черновиков, идей. Не подходит для рабочих документов.

Структурированный формат. «Три абзаца: проблема, решение, результат» — модель следует структуре. Подходит для большинства рабочих задач. Вы можете проверить, есть ли три абзаца и соответствуют ли они заявленной структуре.

Жёсткий контракт. JSON, таблица с заданными колонками, XML-теги. Модель должна вернуть результат в точно заданной форме — но не всегда соблюдает: проверяйте. Подходит для задач, где результат идёт в другую систему или где нужна машинная проверка.

Чем жёстче формат, тем меньше свободы у модели и тем предсказуемее результат. Чем свободнее формат, тем больше вариативности и тем сложнее проверить. Выбирайте по задаче: для идей — свободный, для рабочих документов — структурированный, для интеграций — жёсткий.

Parameters: ограничения, которые работают

Ограничения — самая недооценённая часть промпта. Их либо не пишут, либо пишут так, что модель их игнорирует.

Есть простое правило: **«не делай X» работает хуже, чем «делай Y»**. Модель часто ухватывает только ключевое слово из запрета. «Не пиши длинно» для модели звучит как «пиши длинно» с отрицанием — и она может ухватить только «пиши длинно». «Уложись в 150 слов» — конкретная инструкция, которую можно выполнить.

Сравните:

Не пиши длинно → уложись в 150 слов.

Не используй канцелярит → пиши простыми предложениями, одно действие — один глагол.

Не будь скучным → начни с конкретного примера или вопроса.

Не выдумывай → используй только факты из текста ниже; если факта нет — напиши «нет данных».

Не отклоняйся от темы → каждый абзац должен отвечать на вопрос: [тема].

Не пиши общими словами → каждое утверждение подкрепи примером или цифрой.

Запрет оставляет модели пространство для интерпретации. Позитивная инструкция задаёт направление. Модель лучше следует тому, что можно сделать, чем тому, чего нельзя.

Второе правило ограничений: **чем конкретнее, тем лучше**. «Коротко» — размыто. «До 100 слов» — конкретно. «Дружелюбно» — размыто. «Без формального „Вы“, с обращением по имени» — конкретно.

Третье правило: **не больше трёх ключевых ограничений на промпт**. Если ограничений больше, модель начинает путаться, какое важнее. Если нужно больше — разбейте задачу на две.

Шаблон GRASP

Собрать промпт по GRASP — это ответить на пять вопросов. Шаблон выглядит так:

GOAL (цель): [что сделать — один глагол, один результат]

ROLE (роль): [кто модель в этой задаче — конкретно]

AUDIENCE (аудитория): [кто будет читать результат — конкретно]

STRUCTURE (структура): [в каком виде вернуть: список, таблица, письмо, N абзацев]

PARAMETERS (параметры): [2—3 ключевых ограничения, позитивно]

Пример заполнения для маркетинговой задачи:

GOAL: Переписать описание продукта короче и понятнее.

ROLE: Ты — копирайтер с опытом в B2B (бизнес для бизнеса) продуктах.

AUDIENCE: Руководитель отдела закупок, у которого 30 секунд на чтение.

STRUCTURE: Три предложения: что это, для кого, что даёт.

PARAMETERS: До 60 слов. Без прилагательных-оценок («лучший», «уникальный»). Каждое утверждение — проверяемый факт.

GRASP — это не формальность. Заполняя пять блоков, вы отвечаете себе на вопрос, чего хотите. Часто на этом этапе выясняется, что задача сформулирована не до конца — и это хорошо: лучше понять это до промпта, чем после ответа модели.

Три кейса: как GRASP работает в разных задачах

Кейс 1. Маркетинговый бриф.

Было: «Напиши текст для лендинга нового продукта».

Что не так: не указан продукт, аудитория, формат, тон, длина. Модель напишет что-то среднее — «инновационное решение для современного бизнеса» — и это будет бесполезно.

Стало по GRASP:

GOAL: Написать текст для лендинга нового тарифа CRM (система управления клиентами).

ROLE: Ты — копирайтер с опытом в SaaS (облачный сервис по подписке) продуктах.

AUDIENCE: Владелец малого бизнеса, 5—20 сотрудников, уже

пользуется другой CRM и рассматривает переход.
STRUCTURE: Три блока: заголовок (до 8 слов), подзаголовок
(до 20 слов), три пункта выгод (по одному предложению).
PARAMETERS: Тон — деловой, без пафоса. Не больше 100 слов
всего. Каждая выгода — конкретная (не «удобно», а «импорт
базы за 10 минут»).

Кейс 2. Аналитический отчёт.

Было: «Сделай анализ продаж за квартал».

Что не так: не указаны данные, метрики, аудитория, формат выводов. Модель может написать общие слова о «динамике рынка», которые не относятся к вашим данным.

Стало по GRASP:
GOAL: Выделить три ключевых инсайта из данных о продажах
за Q3.
ROLE: Ты — аналитик данных в отделе продаж.
AUDIENCE: Коммерческий директор, который принимает решения
по бюджету на Q4.
STRUCTURE: Три пункта, каждый — одно предложение с инсайтом
и одно предложение с обоснованием из данных.
PARAMETERS: Только данные из таблицы ниже. Если данных
недостаточно для вывода — напиши «недостаточно данных».
Без общих слов о рынке. Каждый инсайт — с цифрой.

Кейс 3. Учебный план.

Было: «Составь план занятия по промптам».

Что не так: не указана аудитория, длительность, уровень, формат. Модель может составить план на 8 часов для экспертов или на 20 минут для новичков — угадает или не угадает.

Стало по GRASP:
GOAL: Составить план занятия по основам промптинга.
ROLE: Ты — методист, пишешь для взрослой аудитории.

Конец ознакомительного фрагмента.

Текст предоставлен ООО «Литрес».

Прочитайте эту книгу целиком, купив полную легальную версию на Литрес.

Безопасно оплатить книгу можно банковской картой Visa, MasterCard, Maestro, со счета мобильного телефона, с платежного терминала, в салоне МТС или Связной, через PayPal, WebMoney, Яндекс.Деньги, QIWI Кошелек, бонусными картами или другим удобным Вам способом.