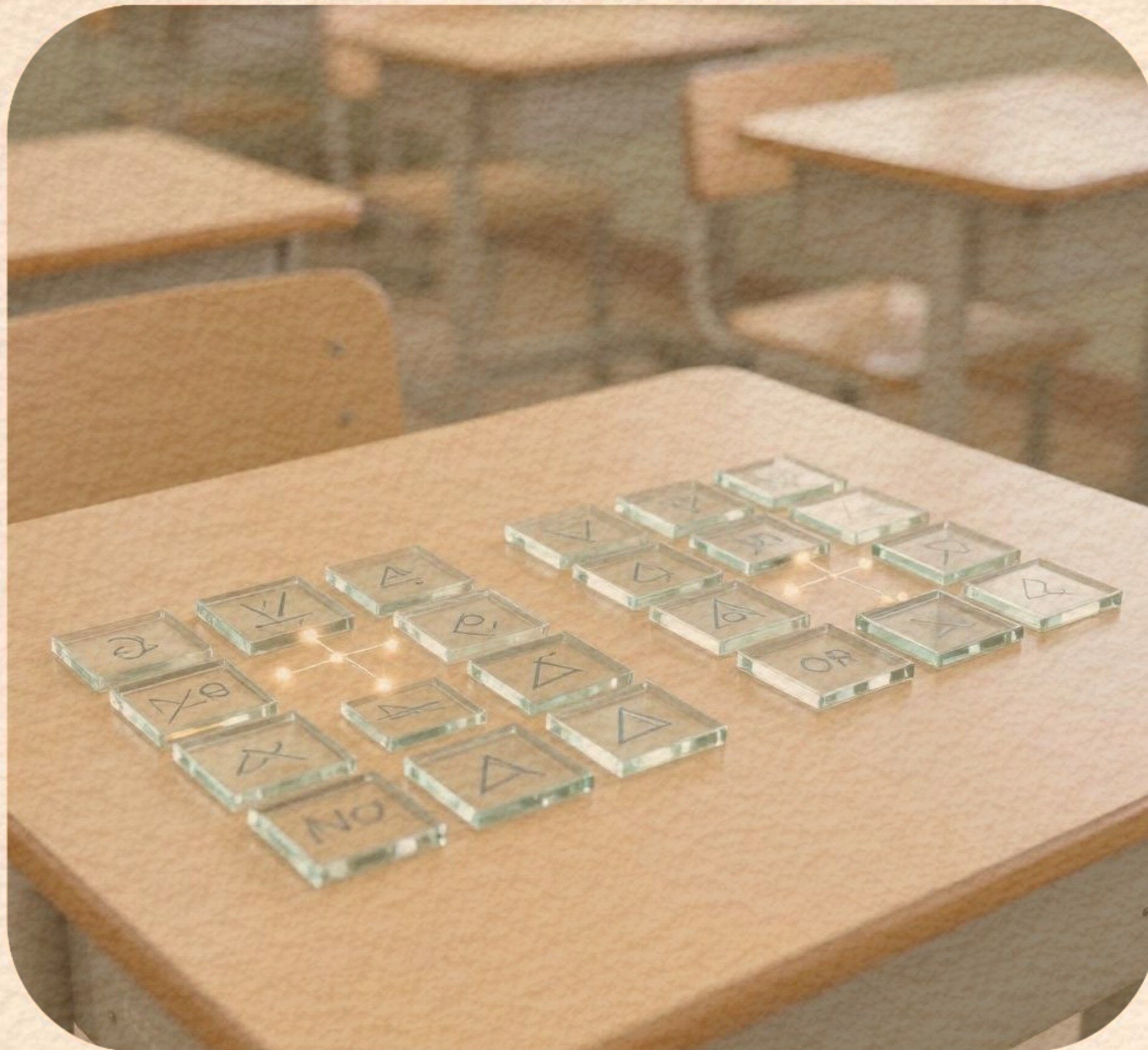


# Преобразование логических выражений

Логические выражения задают условия в программах и помогают анализировать алгоритмы и цифровые схемы.

# Зачем изучать преобразование логических выражений в 10 классе



Логические выражения встречаются в ветвлениях, циклах и проверках условий. Сложная формула хуже читается, повышает риск ошибки и усложняет проверку. Преобразования сохраняют смысл, сокращают запись и опираются на приоритет, де Моргана, поглощение, распределение и таблицы истинности.

# Логические операции и их обозначения



## НЕ (отрицание)

Отрицание  $\neg A$  (или  $!A$ ) меняет значение на противоположное: если  $A$  истинно, то  $\neg A$  ложно, и наоборот. В выражениях это помогает явно учесть «не выполняется».



## И (конъюнкция)

Конъюнкция  $A \wedge B$  (или  $A \&\& B$ ,  $A \& B$ ) истинна только тогда, когда истинны оба операнда. Если хотя бы один из них ложен, всё выражение становится ложным.



## ИЛИ (дизъюнкция)

Дизъюнкция  $A \vee B$  (или  $A || B$ ,  $A | B$ ) истинна при истинности хотя бы одного операнда. Когда оба условия ложны, результат тоже ложен — это ключевой смысл выбора вариантов.

# Приоритет логических операций

Сначала выполняются операции внутри скобок, затем НЕ, после – И, и в последнюю очередь ИЛИ. Поэтому без скобок результат может измениться.

Неправильный приоритет меняет значение условия. Скобки нужны, чтобы зафиксировать однозначную структуру логики.

| Порядок | Операция | Обозначения        |
|---------|----------|--------------------|
| 1       | Скобки   | ( ... )            |
| 2       | НЕ       | $\neg A$ , !A      |
| 3       | И        | $A \square B$ , && |
| 4       | ИЛИ      | $A \sqcup B$ ,     |

*Школьный курс алгебры логики и правила записи логических выражений.*

# Таблица истинности как способ проверки

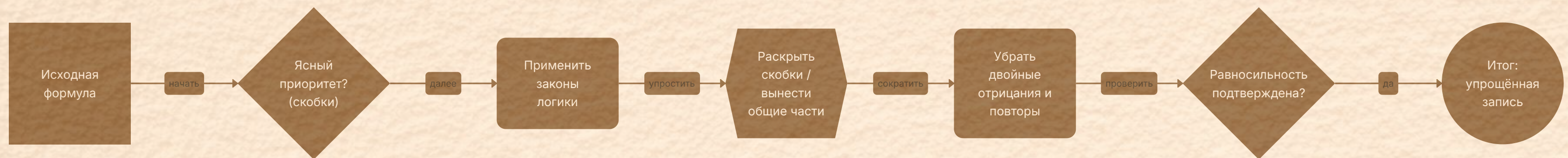


Таблица истинности показывает результат для всех наборов значений переменных. Для двух переменных получается 4 строки, для трёх — 8, для четырёх — 16, поэтому проверка становится полной и однозначной.

Если две формулы дают одинаковые значения в каждой строке, их можно считать равносильными. На практике удобно сверять выражения с НЕ, И и ИЛИ по подчеркнутым строкам результата вычисления.

# Алгоритм преобразования логического выражения

Пошаговая схема для сохранения смысла и контроля равносильности.



Закон  $\neg\neg A = A$  действует для любых переменных и составных выражений. Он помогает убрать лишние отрицания и сделать формулу короче без изменения смысла.

*Алгебра логики (школьный курс).*

## 2

Две операции отрицания взаимно компенсируют друг друга.

# Законы де Моргана

Законы де Моргана описывают, как внешнее отрицание распределяется по скобкам:  $\neg(A \vee B)$  превращается в  $\neg A \wedge \neg B$ , а  $\neg(A \wedge B) \rightarrow \neg A \vee \neg B$ . Меняется и операция внутри.

Такое преобразование особенно полезно, когда в выражении есть отрицание «снаружи». Развернув его по правилам, вы получаете набор отрицаний внутри, и дальнейшее упрощение становится проще и системнее.

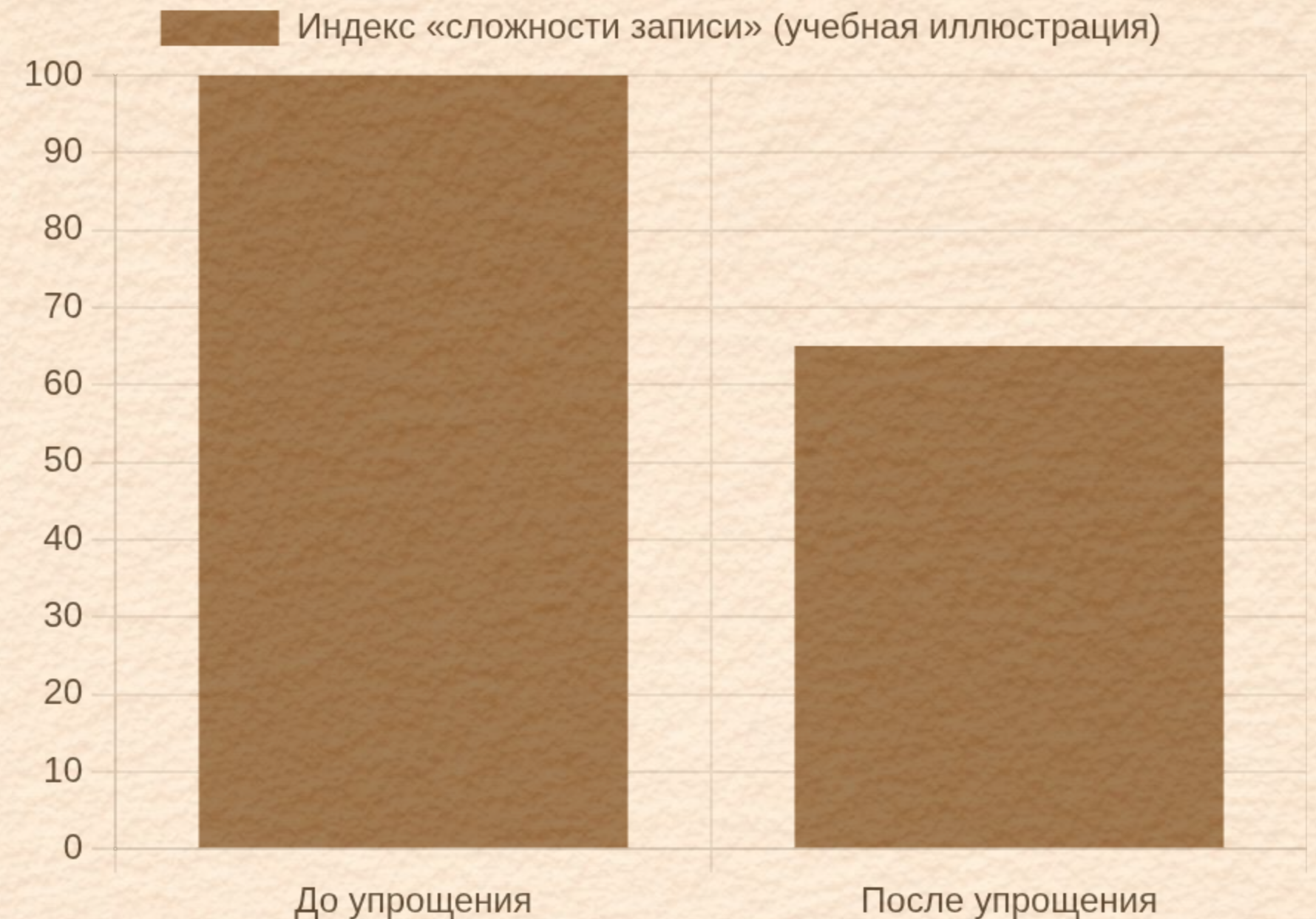
В задачах де Моргана обычно комбинируют с законом двойного отрицания: после переноса отрицания внутрь часто появляется  $\neg\neg$ , которое сразу можно сократить. Это уменьшает длину записи и ускоряет вывод.

# Как меняется структура формулы после упрощения

После корректных преобразований обычно сокращается число операций И и ИЛИ, уменьшается количество вложенных скобок и уходит часть повторяющихся фрагментов.

Тенденция едина: при равносильности смысл сохраняется, а формула требует меньше вычислительных шагов и символов, поэтому её проще анализировать и проверять.

*Учебная визуализация на основе типичных результатов упрощения по законам алгебры логики (качественный характер данных).*



# Коммутативность и ассоциативность

Для операции И верно  $A \vee B = B \vee A$ , а для ИЛИ —  $A \wedge B = B \wedge A$ . Это позволяет переставлять операнды местами, сохраняя истинность всего выражения.

В преобразованиях эти свойства помогают «навести порядок»: сначала удобно сгруппировать одинаковые части, затем применить другие законы — поглощение, распределение или идемпотентность.

Ассоциативность утверждает, что группировка одинакова по смыслу:  $(A \vee B) \vee C = A \vee (B \vee C)$  и  $(A \wedge B) \wedge C = A \wedge (B \wedge C)$ . Значит, скобки можно переносить между однородными операциями.

Практический смысл коммутативности и ассоциативности — меньше ошибок при чтении. Когда формула приведена к аккуратной структуре, её проще проверить и объяснить другим.

# Распределительные законы

01.

**Старт: когда нужно раскрыть скобки**

Если в выражении встречается  $A (B \vee C)$  или  $A (B \wedge C)$ , распределительные законы позволяют «разнести» операцию и убрать внешние скобки. Смысл при этом сохраняется.

02.

**Шаг: переход между формами записи**

Пример:  $A (B \vee C) = (A \vee B) (A \vee C)$ . Аналогично  $A (B \wedge C) = (A \wedge B) (A \wedge C)$ . Меняется структура, но результат остаётся тем же при любых значениях переменных.

03.

**Финиш: первый шаг к упрощению**

На практике именно после распределения появляется возможность сокращать формулу законами поглощения и идемпотентности. Иногда распределение специально делают «в обе стороны», чтобы выбрать удобный вид.

# Примеры равносильных преобразований

| Исходное выражение      | Закон             | Итоговая запись                    |
|-------------------------|-------------------|------------------------------------|
| $\neg\neg A$            | Двойное отрицание | $A$                                |
| $\neg(A \sqcap B)$      | де Моргана        | $\neg A \sqcup \neg B$             |
| $A \sqcap (B \sqcup C)$ | распределение     | $(A \sqcap B) \sqcap (A \sqcap C)$ |
| $A \sqcap (A \sqcup B)$ | поглощение        | $A$                                |

Сравните исходные выражения и их преобразованные формы. В каждой строке указан применённый закон, а в последней – подтверждение равносильности через совпадение значений по таблице истинности.

Разные цепочки преобразований приводят к одной и той же равносильной записи: достаточно, чтобы на каждом шаге применялось корректное тождество, а итог совпадал по таблице истинности.

*Школьный курс алгебры логики и правила равносильных преобразований.*

# Поглощение и идемпотентность



## Почему появляются повторяющиеся фрагменты

Когда в выражении встречаются повторяющиеся части, они часто «засоряют» запись и делают её длиннее, чем нужно. Закон поглощения устраняет такие конструкции:  $A (A \vee B) = A$  и  $A (A \wedge B) = A$ , сохраняя смысл.

## Как законы сокращают формулу

Идемпотентность показывает, что одиночный повтор не меняет логическое значение:  $A \wedge A = A$  и  $A \vee A = A$ . В задачах это помогает убирать дубль-конъюнкции или дубль-дизъюнкции и быстрее приходить к компактной форме.

# ДНФ и КНФ как удобные формы записи

Дизъюнктивная нормальная форма (ДНФ) — это дизъюнкции конъюнкций, а конъюнктивная нормальная форма (КНФ) — конъюнкции дизъюнкций. Такие формы упрощают разбор логики как «случаи» или «ограничения».

ДНФ особенно удобна, когда нужно перечислить все подходящие ситуации: «либо выполняется первый набор условий, либо второй...». КНФ удобнее, когда важно задать условия, которые должны выполняться одновременно для допустимых наборов.

В школьных задачах переход к ДНФ или КНФ помогает систематизировать выражение, выявить структуру и подготовить его к дальнейшему анализу. Кроме того, эти формы хорошо подходят для алгоритмической проверки и сопоставления с таблицами истинности.

# Преобразование к ДНФ или КНФ

Логический маршрут: сохраняем равносильность на каждом переходе

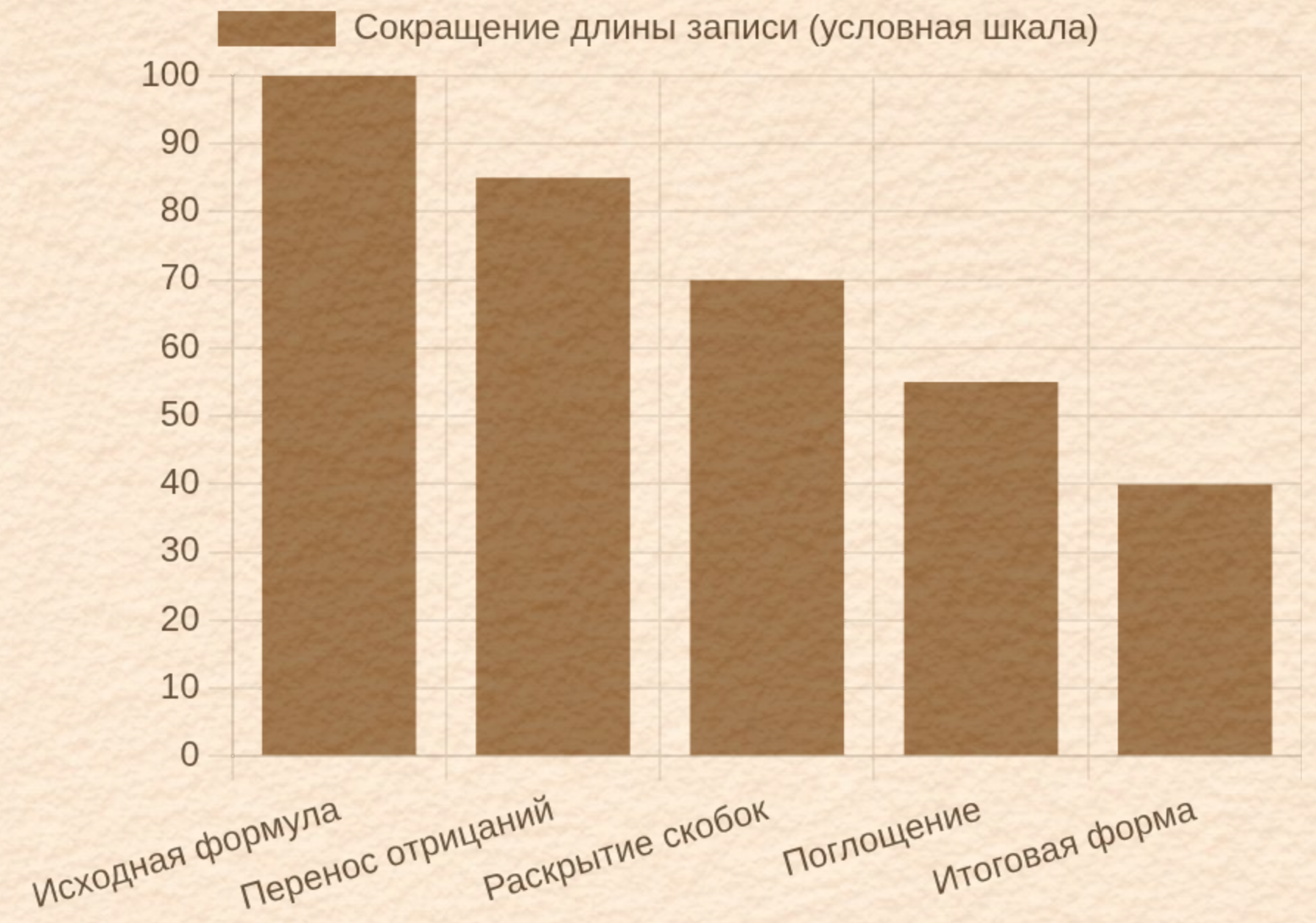


# Изменение длины выражения на этапах преобразования

На практике относительное уменьшение длины чаще всего заметнее после де Моргана и устранения повторяющихся фрагментов. Числа условные, показана общая тенденция.

Чем точнее вы доводите структуру до удобного вида, тем больше возможностей для сокращений. Равносильность сохраняется, но запись становится короче и понятнее.

*Схематическая иллюстрация по этапам преобразования (без привязки к конкретной формуле).*



# Типичные ошибки при преобразовании



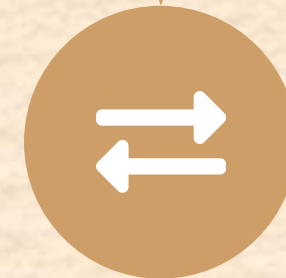
## Ошибка приоритета

Чаще всего путают порядок вычисления: сначала скобки, затем НЕ, потом И, и только после этого ИЛИ. Без скобок легко получить другой смысл выражения.



## Пропуск или неверное раскрытие скобок

Если забыть скобки или раскрыть их не по закону, выражение перестаёт быть равносильным. Особенно опасно, когда отрицание стоит перед группой.



## Неправильное применение законов де Моргана

Частая ловушка — поменять местами операции при переносе отрицания, но сделать это несимметрично:  
 $\neg(A \vee B)$  и  $\neg(A \wedge B)$  преобразуются в разные операции и требуют строгого соответствия.



## Нет проверки по таблице истинности

Даже одна неверная замена ломает равносильность. Поэтому итог желательно подтвердить совпадением значений по таблице истинности для всех наборов переменных.

# Применение в информатике

01.

Преобразование логических выражений используется, когда программа выбирает ветку выполнения: условие в if и составные проверки строятся из НЕ, И и ИЛИ. Формула определяет результат на конкретных данных.

02.

В циклах и фильтрах данных логика играет роль «критерия отбора»: нужно аккуратно записать условия, чтобы ничего не пропустить. Упрощение помогает яснее увидеть, какие именно случаи считаются подходящими.

03.

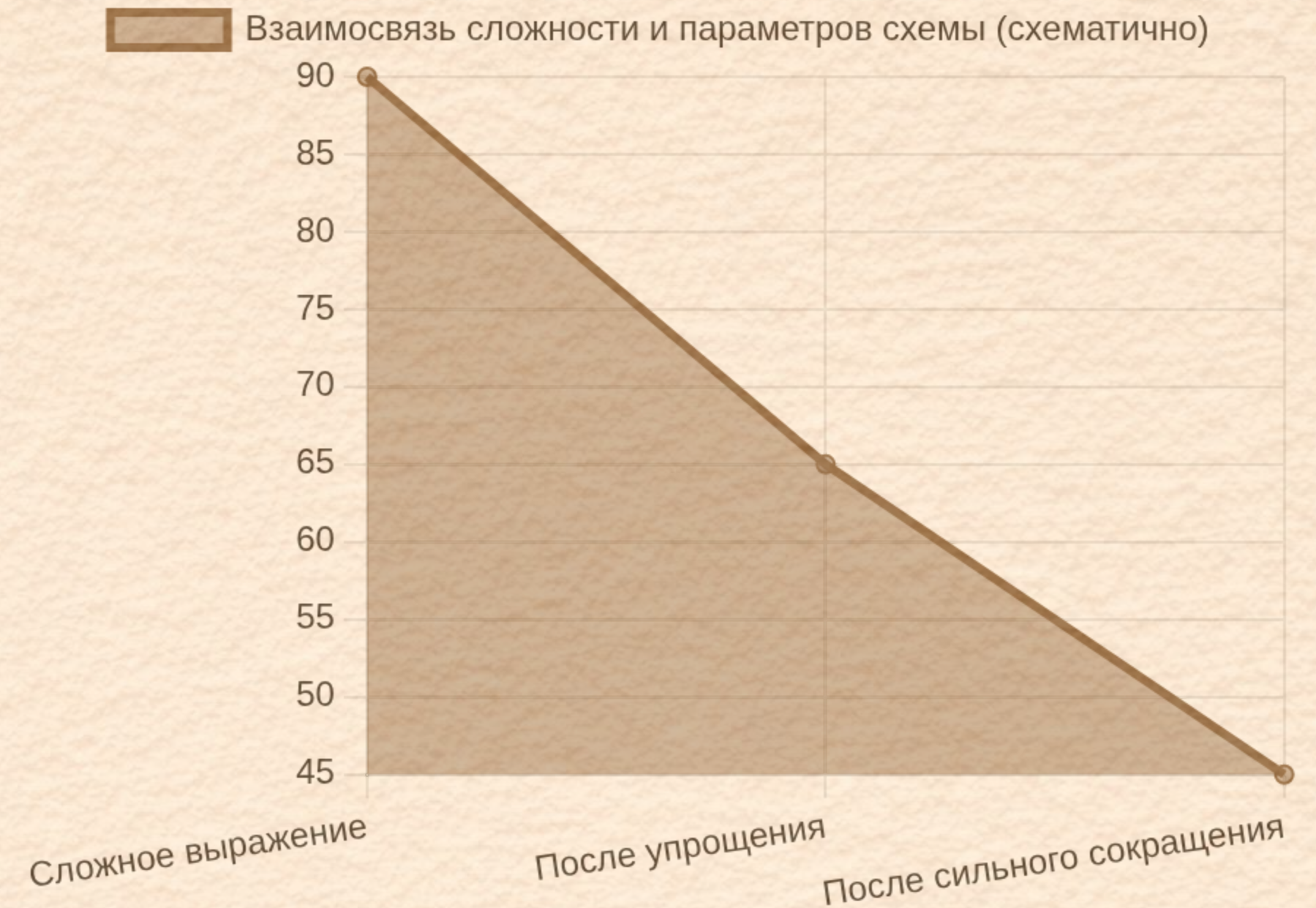
В программировании операторы !, && и || соответствуют логическим операциям, а итоговая проверка подчиняется тем же правилам, что и в алгебре логики. Меньшая по объёму запись обычно проще читать и отлаживать.

# Почему упрощение важно для цифровых схем

В простых схемах требуется меньше вентиляей, уменьшается число соединений и сокращается путь сигнала. В результате ниже риск реализации ошибок и легче поддерживать корректность логики.

Упрощение выражения напрямую связано с экономией ресурсов: сокращаются элементы и соединения, а значит, растёт надёжность и уменьшается задержка. Это делает подход практичным.

*Обобщённая зависимость из логического проектирования цифровых устройств (схема без конкретных измерений).*



# Логика становится понятнее после преобразований

Преобразование логических выражений опирается на законы алгебры логики и проверяется таблицами истинности. Приоритет операций, двойное отрицание, де Морган, распределение, поглощение и идемпотентность помогают получить равносильную, более удобную формулу для программ и цифровых схем.