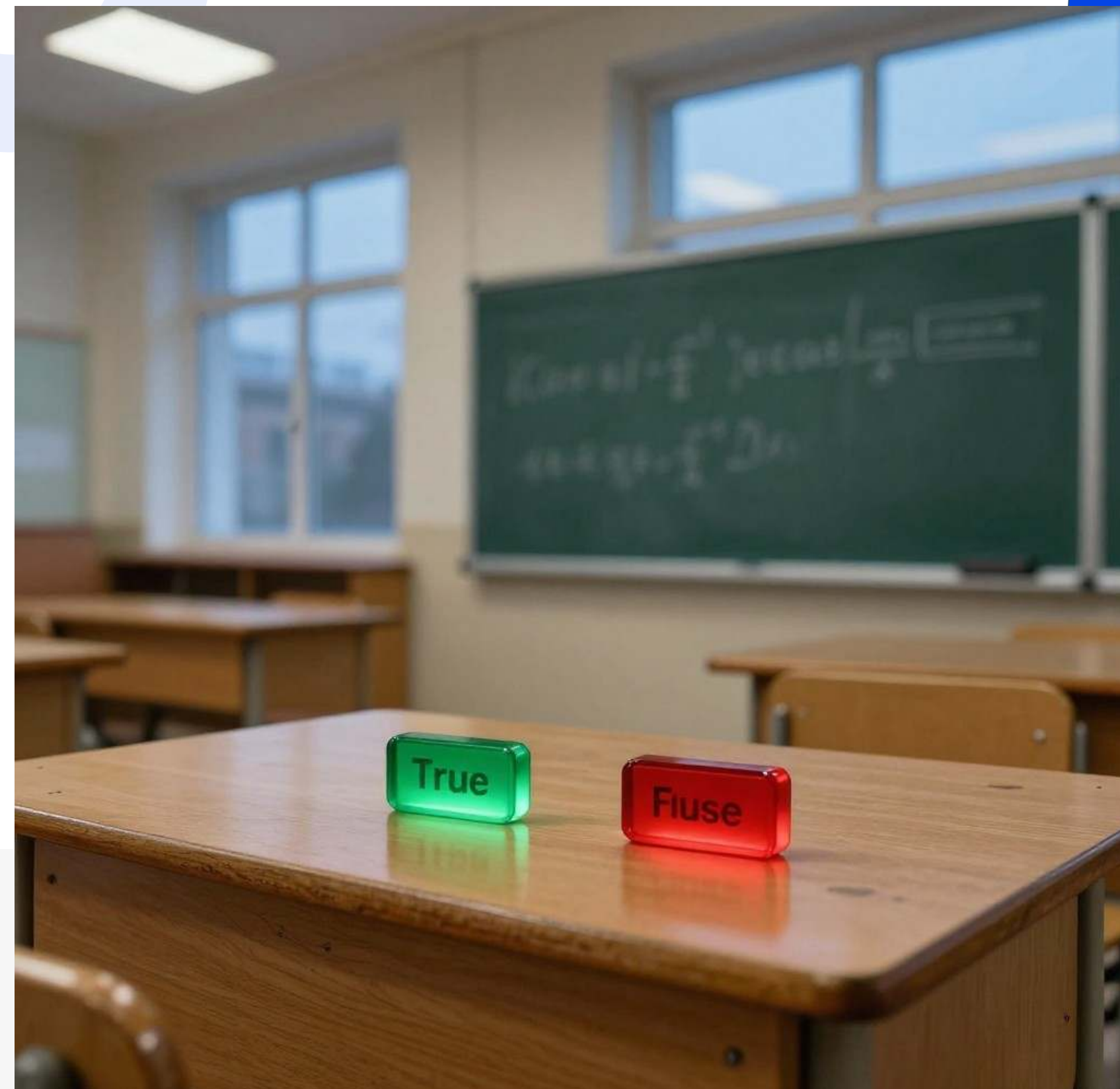


Логические величины, операции и выражения

Истина и ложь в информатике: операции И, ИЛИ, НЕ, исключающее ИЛИ, скобки и приоритет.



Зачем школьнику нужна логика в информатике



Логические выражения управляют ветвлениями, циклами и проверками условий. Булева алгебра названа в честь Джорджа Буля: он описал правила работы с истиной и ложью. В цифровой технике эти правила воплощаются в логических элементах, а в сервисах — в фильтрах поиска и авторизации.

Логическая величина: что именно проверяем



Истина и ложь

Любое логическое утверждение отвечает только на один вопрос: верно или неверно. В задачах это часто представляют как 1 и 0, не смешивая другие числовые смыслы.



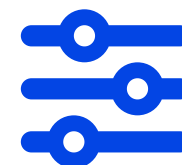
Логическая переменная

Переменная типа `boolean` фиксирует результат проверки. Она отвечает на смысловое условие полностью: выполнено ли требование в задаче или в программе, а не просто «частично».



Примеры из школьных задач

«Число больше нуля», «файл существует», «пароль введен верно», «ученик выполнил условие». Такие проверки превращают текст задачи в формальные правила.



Двоичное представление в коде

Когда условие превращают в программу, логические проверки удобно хранить и комбинировать. Именно поэтому логические величины лежат в основе условий, фильтров и решений алгоритмов.



Основные логические операции

Три базовые операции задают способ объединять и изменять условия. Их обозначения совпадают в математике, в схемах и в программировании.

Ключевой вывод: операция определяет «строгость» условия. И требует одновременного выполнения, ИЛИ допускает альтернативу, а НЕ переворачивает смысл.

Операция	Когда истинно	Запись
И	Когда истинны оба	AND / and
ИЛИ	Когда истинно хотя бы одно	OR / or
НЕ	Когда меняется на противоположное	NOT / not / !



Таблица истинности операции И

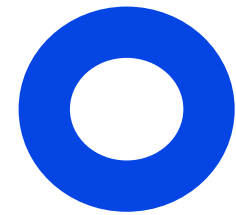
При операции И результат становится истинным только тогда, когда оба условия истинны одновременно. Любая «ошибка» в одном операнде сразу обнуляет итог.

Диаграмма подчёркивает: И — самая требовательная базовая операция. Она подходит, когда критерии должны совпасть строго все.

Формализованная таблица истинности для булевых операций (без использования внешних данных)



Операция ИЛИ и исключающее ИЛИ



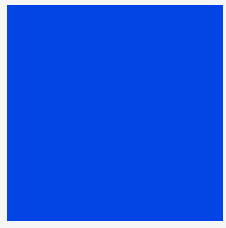
Операция ИЛИ даёт истину, если выполнено хотя бы одно из двух условий. Это удобно для «разрешено по первому критерию или по второму», когда достаточно одного основания.



Различие между ИЛИ и исключающим ИЛИ влияет на смысл алгоритма и тестов. Поэтому при составлении условий важно выбрать операцию, соответствующую задаче: «один из многих» или «ровно один».



Исключающее ИЛИ (XOR) истинно только при различии значений операндов: одно условие истинно, другое — ложно. Если оба совпадают (оба истина или оба ложь), результат становится ложью.



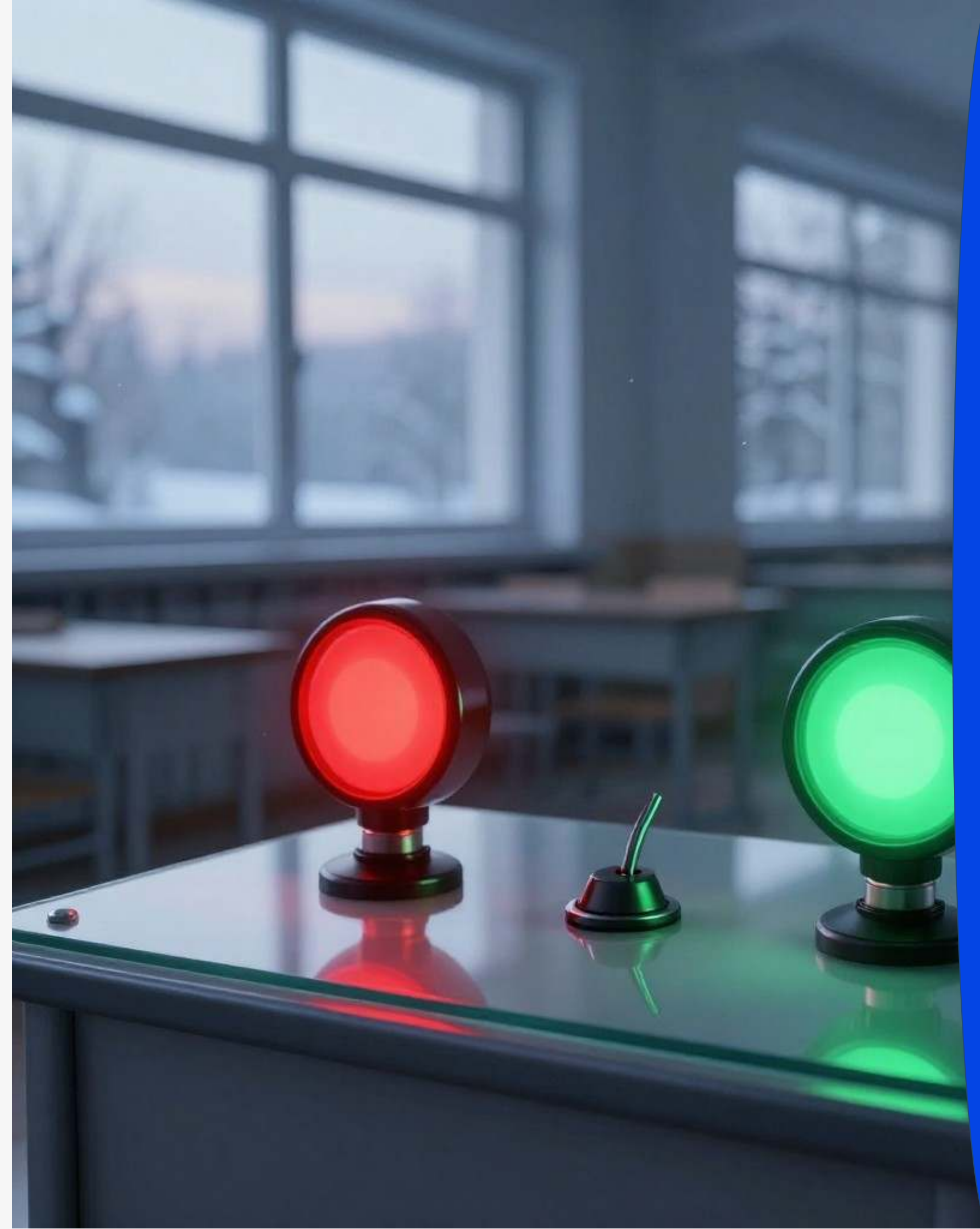
Операция НЕ и инверсия условия

Как работает отрицание

Операция НЕ меняет состояние: истина превращается в ложь, а ложь — в истину. В программировании её часто записывают как `not`, ! или как отрицание условия в тексте проверки.

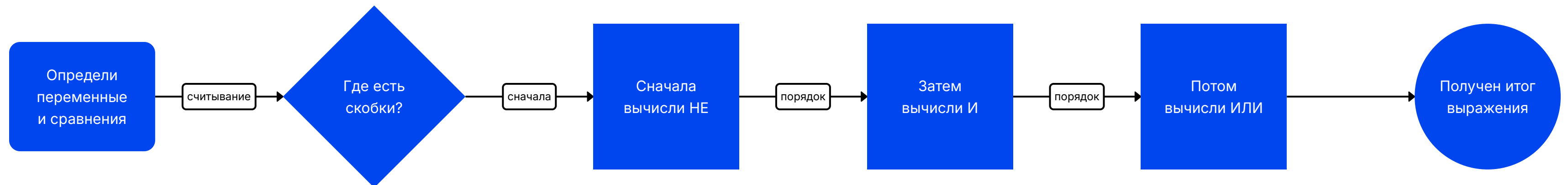
Как это выглядит в задачах и интерфейсах

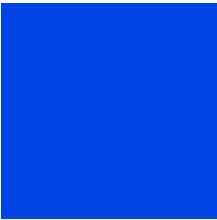
Примеры формулировок: «не равно», «не выполнено», «не найдено», «не является пустым». Визуально отрицание часто изображают переключателем или знаком, который показывает переход между двумя противоположными значениями.



Порядок вычисления логического выражения

Чтобы верно оценить условие, сначала вычисляют простые части, затем применяют приоритет операций и только потом учитывают итоговые скобки.





Сложные логические выражения

01

Сложное выражение соединяет несколько простых условий через операции и скобки. Например, $(A > 5) \text{ И } (B < 10)$: итог зависит от истинности обоих сравнений одновременно.

02

Чем больше условий, тем важнее структура записи. В длинных формулах появляются несколько уровней скобок и отрицаний, и каждое из них задаёт собственную «логику связывания» критериев.

03

Истинность всей формулы определяется не только значениями переменных, но и тем, как они организованы в выражении. Поэтому два похожих на вид записи могут дать разные результаты.



Приоритет логических операций

При одинаковых А и В выражения могут дать разные итоги, если скобки заданы иначе. Причина — смена того, какие части считаются раньше.

На практике это означает: приоритет НЕ $\rightarrow$ И $\rightarrow$ ИЛИ нельзя игнорировать. Скобки делают смысл записи однозначным и уменьшают вероятность ошибки в условии.

Правила приоритета и порядок вычисления логических операций (учебный материал по информатике)

Таблица истинности как способ проверки

1. Таблица истинности перечисляет все возможные наборы значений логических переменных и результат выражения для каждого набора. Для двух переменных получается 4 строки, а для трёх — 8.

2. Проверка с таблицей не требует догадок: каждое сочетание «истина/ложь» рассматривается явно. Поэтому ошибку в сложном условии можно найти сразу, ещё до написания программы.

3. Такой подход помогает учиться логике: вы видите, какая часть выражения «ответственна» за изменение результата. Постепенно становится понятным смысл И, ИЛИ, НЕ и исключающего ИЛИ.

4. В информатике таблицы истинности используют при составлении программ и проверке логических схем. Они служат универсальным способом подтвердить корректность условий и связей между сигналами.

Сравнение базовых логических операций

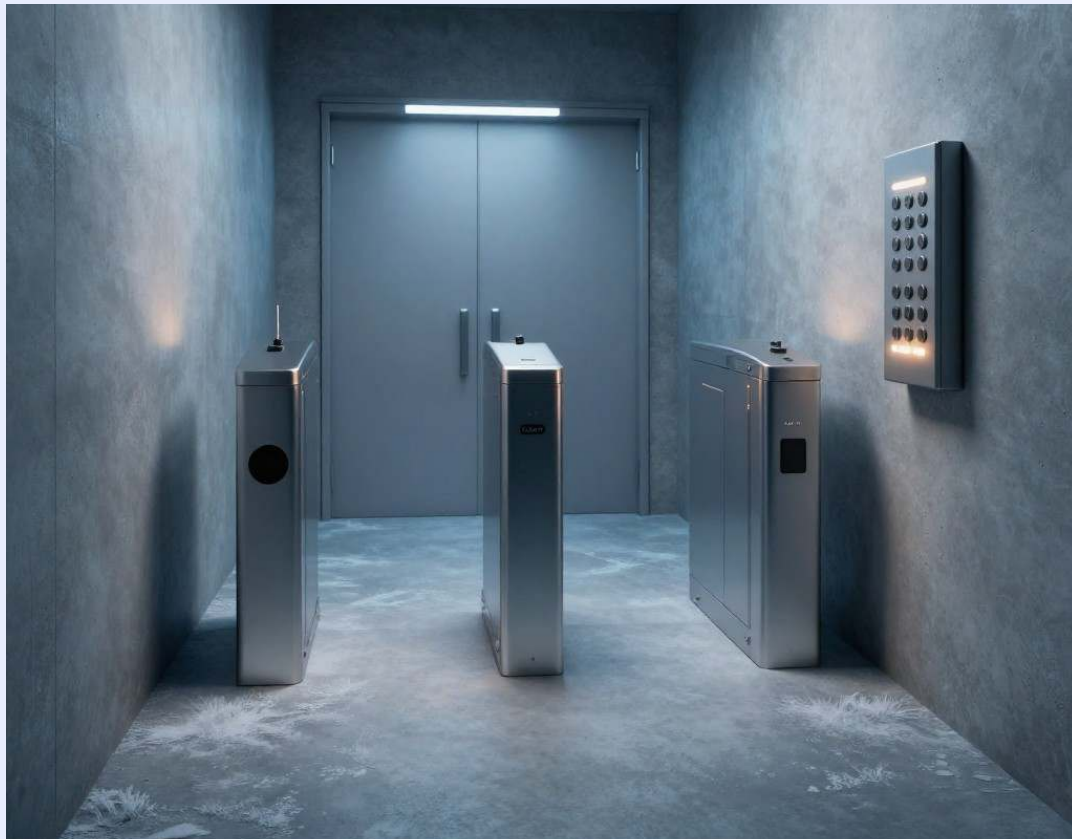
Операция	Когда истинна	Пример применения
И	Истинна, когда оба условия истинны	«Выполнены А и В»
ИЛИ	Истинна, когда истинно хотя бы одно	«Подходит А или В»
НЕ	Истинна, когда условие ложно	«Не выполнено»
Исключающее ИЛИ	Истинна, когда значения различны	«Выбрать ровно один вариант»

И, ИЛИ, НЕ и исключающее ИЛИ удобно сравнивать через условия истинности: что именно должно получиться «истиной» и где это соответствует смыслу задачи.

Разные операции моделируют разный смысл требований: «все условия» (И), «достаточно одного» (ИЛИ), «обратное» (НЕ) и «ровно один» (исключающее ИЛИ). Поэтому выбор операции нельзя делать наугад.

Учебные материалы по булевой алгебре
(обобщённые правила операций И/ИЛИ/НЕ).

Логика в повседневных интерфейсах: короткие истории



Вход на сайт: проверка условий за кулисами

При входе на сайт логика решает, можно ли продолжить: пароль верный, пользователь существует, данные введены корректно. Если хотя бы одно условие не проходит, действие блокируется. Так абстрактные «истина/ложь» превращаются в реальные решения интерфейса.



Фильтры в интернет-магазине: «подходит ли товар»

Когда вы выбираете фильтры, система собирает логическое выражение из ваших условий: цена, категория, наличие, бренд. Операции И/ИЛИ определяют, какие товары считаются подходящими. Поэтому один чекбокс может расширить выбор, а другой — сузить его.



Чекбоксы и тесты: «если — то» в миниатюре

В электронных тестах и формах условия прячутся в логике ответов: выбрана ли опция, выполнено ли требование, совпадает ли результат с ожидаемым. Чекбоксы и переключатели обычно соответствуют простым логическим переменным. А итог зависит от комбинаций этих признаков.



Как меняется результат выражения $A \wedge B$ ИЛИ НЕ C

Здесь C действует как «переключатель»: при $C = 0$ отрицание делает выражение истинным независимо от A и B .

График показывает закономерность: когда C истинно, результат зависит только от одновременного выполнения A и B ; когда C ложно — выражение всегда истинно.

Расчёт по правилам булевой алгебры для A, B, C как логических переменных (0/1), без внешних статистических данных.

Законы логики

1

Законы коммутативности, ассоциативности и дистрибутивности позволяют менять структуру выражения без изменения смысла. Для И и ИЛИ порядок операндов обычно не влияет на результат.

2

Коммутативность означает, что $A \vee B$ и $B \vee A$ дают одинаковую истинность, так же как $A \wedge B$ и $B \wedge A$. Это полезно, когда формулы нужно привести к более удобному виду.

3

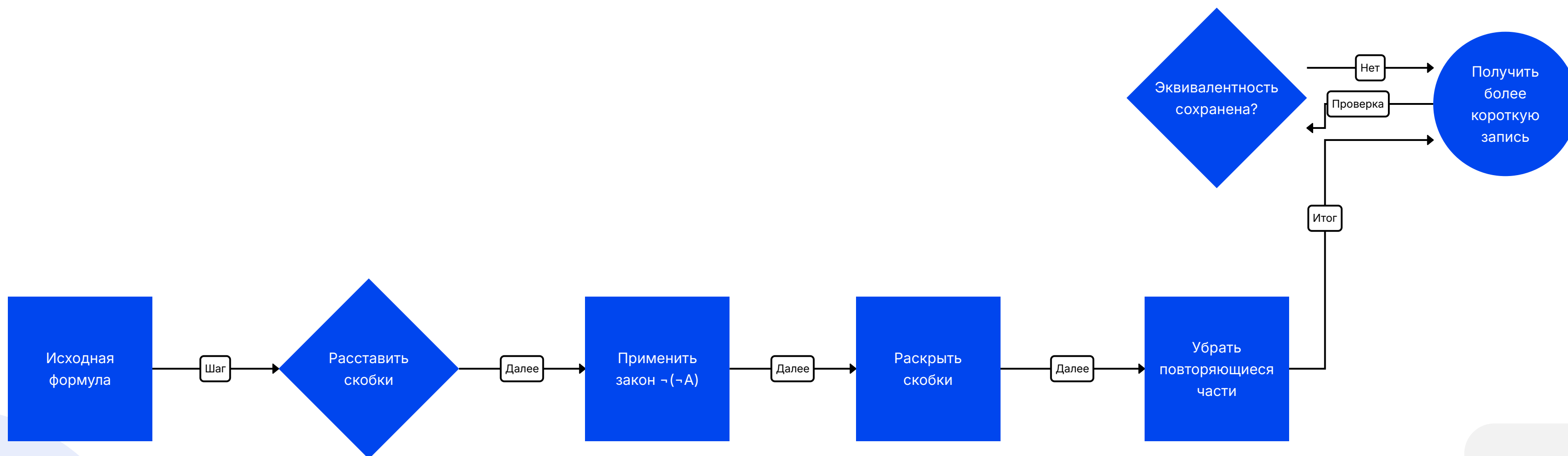
Ассоциативность позволяет иначе группировать одинаковые операции: $(A \vee B) \vee C$ и $A \vee (B \vee C)$ эквивалентны. Благодаря этому выражение можно «собирать» или «разбирать» по шагам.

4

Дистрибутивность связывает И и ИЛИ, помогая раскрывать скобки и упрощать условия. Наконец, закон двойного отрицания возвращает исходное значение: $\neg(\neg A)$ равно A .

Упрощение логического выражения

Преобразование эквивалентно исходному: сохраняется истинность при любых наборах переменных.



Типичные ошибки при работе с логикой



Путаница И и ИЛИ

Частая проблема — перепутать операции: где требуются оба условия, используется ИЛИ, и наоборот. Проверка таблиц истинности помогает сразу увидеть, почему результат становится неожиданным.



Пропуск скобок

Без скобок приоритет операций может изменить смысл выражения. В итоге формула читается иначе, чем вы задумали. Скобки позволяют задать порядок вычисления однозначно.



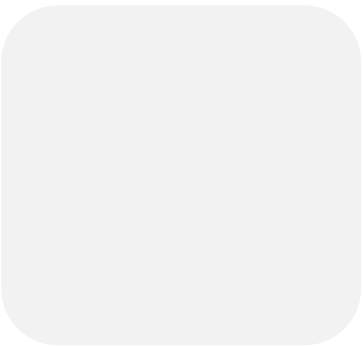
Неверное понимание отрицания

Ошибки возникают, когда неверно читают НЕ: что именно оно меняет — всё выражение или только один фрагмент. В сложных условиях важно внимательно расставлять отрицания и скобки.



Смешение сравнения и присваивания

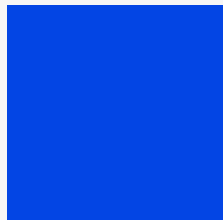
Иногда в коде путают сравнение и присваивание, из-за чего условие превращается в другую инструкцию. Это нарушает логику ветвления и даёт неправильное поведение программы.



Где логические операции встречаются чаще всего

Диаграмма показывает не «точные проценты», а относительную частоту применения: больше всего логика нужна там, где постоянно принимаются решения по данным.

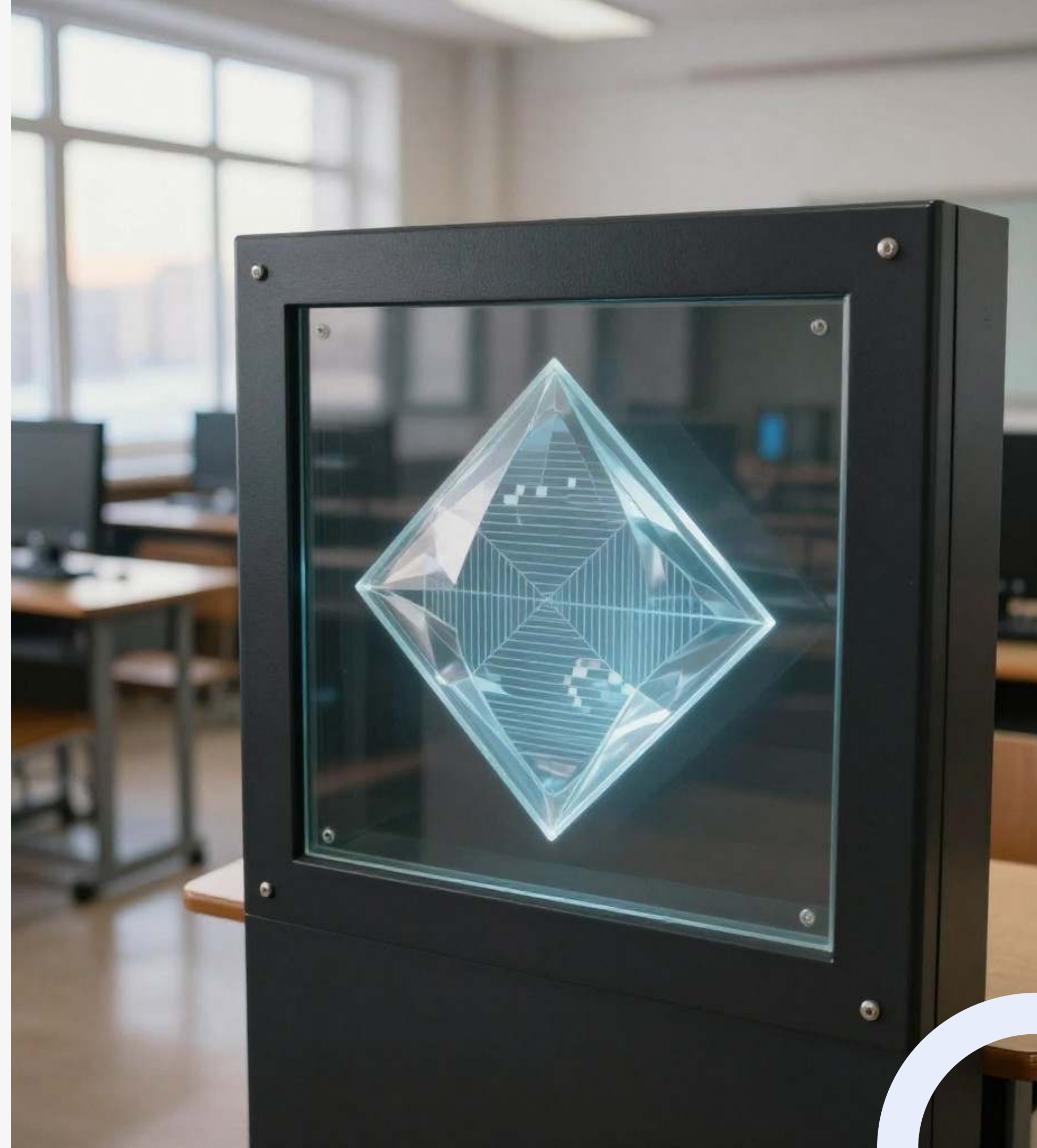
Логические операции пронизывают цифровые сервисы: они обеспечивают выбор ветвей, проверку корректности и контроль доступа. Чем больше решений в системе, тем больше логики внутри.



От логических выражений к алгоритмам

Логические проверки задают ветвления в алгоритмах и определяют, какие действия выполняются дальше. Когда условие истинно, происходит один переход, когда ложно — другой, так ограничивается доступ к ресурсам.

В блок-схемах это оформляют ромбами с вариантами «да» и «нет». Если вы видите ромб, значит внутри записано логическое выражение, и программа принимает решение по его значению.





Булева логика — база решений в информатике

Логические величины, операции и выражения лежат в основе условий в информатике, программировании и цифровых устройствах. Понимание И, ИЛИ, НЕ, исключающего ИЛИ, таблиц истинности и приоритета вычисления помогает анализировать код и составлять алгоритмы без ошибок.