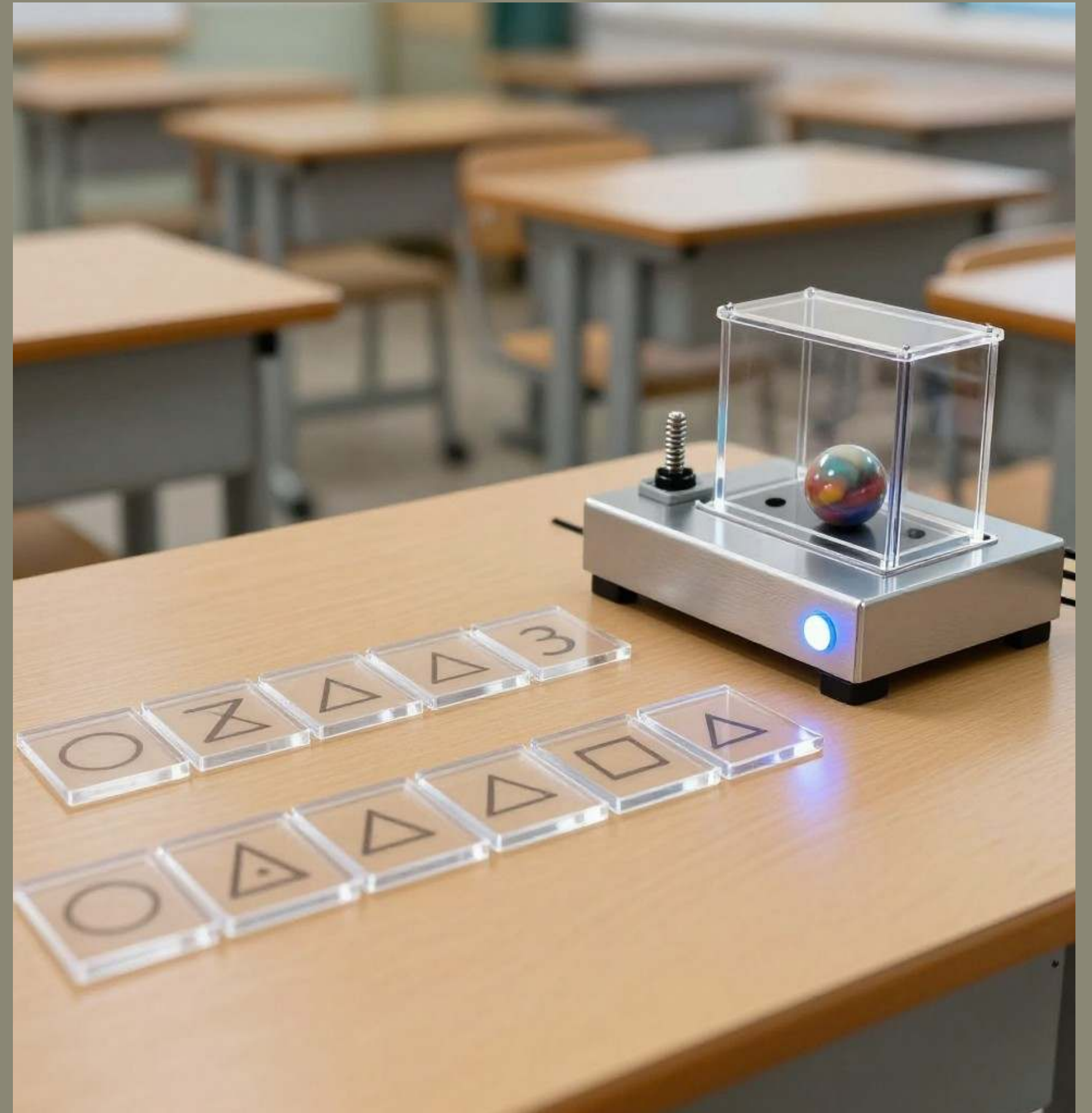


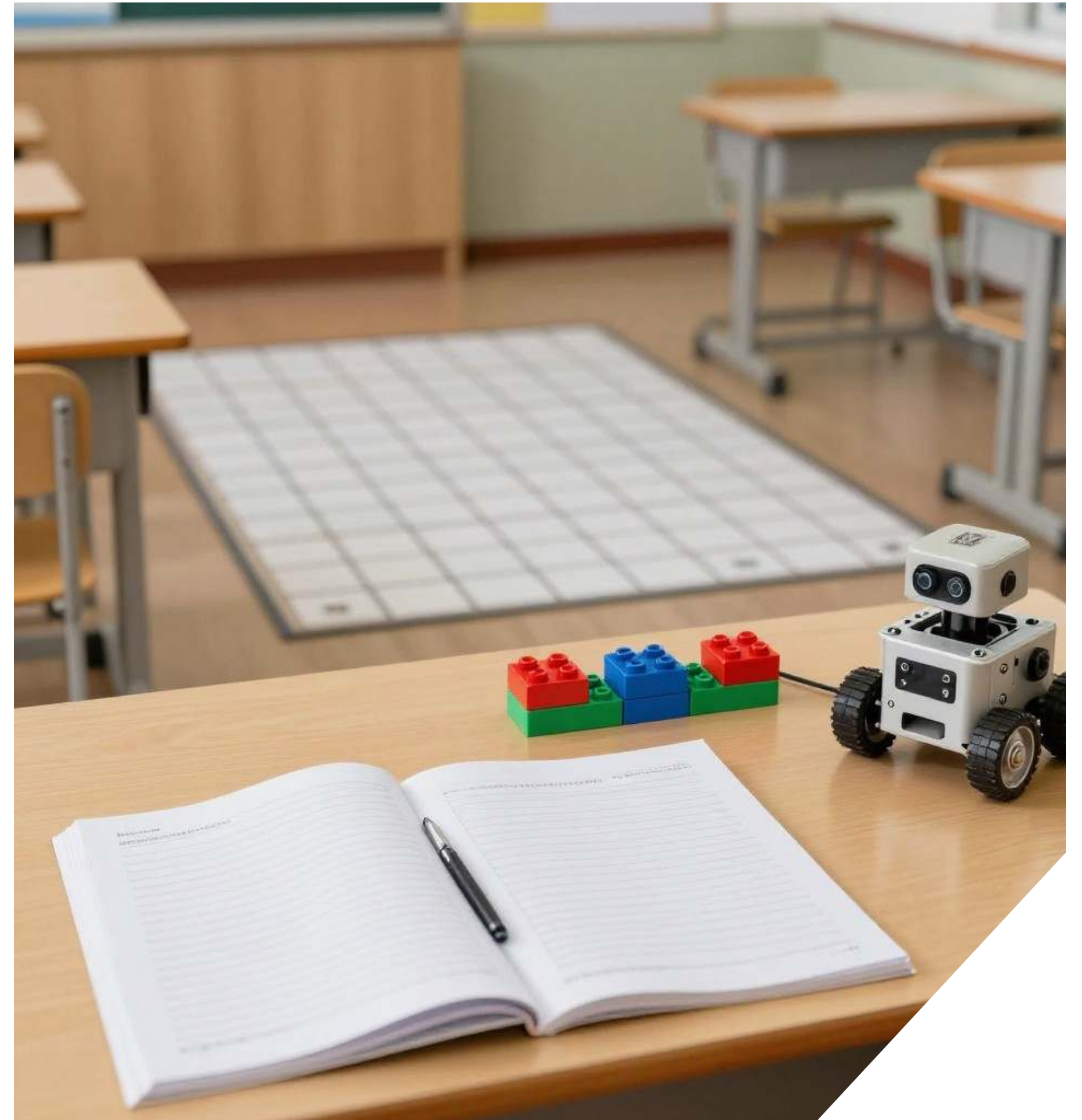
Алгоритм и его формальное исполнение

Разберём, что такое алгоритм и как
строгое выполнение по записи
приводит к результату.



> Зачем это нужно в 9 классе

Алгоритмы лежат в основе программ, игр, поисковиков, навигаторов и роботов. В школе вы узнаете, как переводить задачу в команды для компьютера. Формальное исполнение объясняет, почему одна и та же запись даёт одинаковый результат. Это основа точного алгоритмического мышления.



> Что такое алгоритм

Алгоритм — это конечная последовательность команд, которая приводит к нужному результату. У него есть входные данные, затем выполняется набор действий, и в конце появляется выходной результат.

Даже в учёбе есть алгоритмы: правило решения квадратного уравнения по шагам подсказывает, что делать сначала, что — дальше, и как получить ответ.

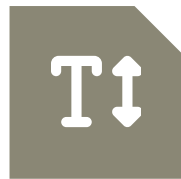
В повседневной жизни мы постоянно действуем по алгоритмам: рецепт задаёт шаги приготовления, маршрут от дома до школы — путь, а инструкция по сборке мебели — порядок действий.

> Основные свойства алгоритма



Дискретность

Выполнение идёт по отдельным шагам: алгоритм разбивает задачу на действия, которые можно выполнить одно за другим. Это помогает понимать и проверять каждый этап решения.



Понятность

Каждая команда алгоритма должна быть понятной исполнителю. Нельзя оставлять вопросы вроде «что именно сделать?» — иначе разные люди могут выполнить по-разному.



Определённость

Для одной и той же ситуации команды должны выполняться одинаково. Когда формулировка не допускает разных толкований, результат становится предсказуемым.



Результативность

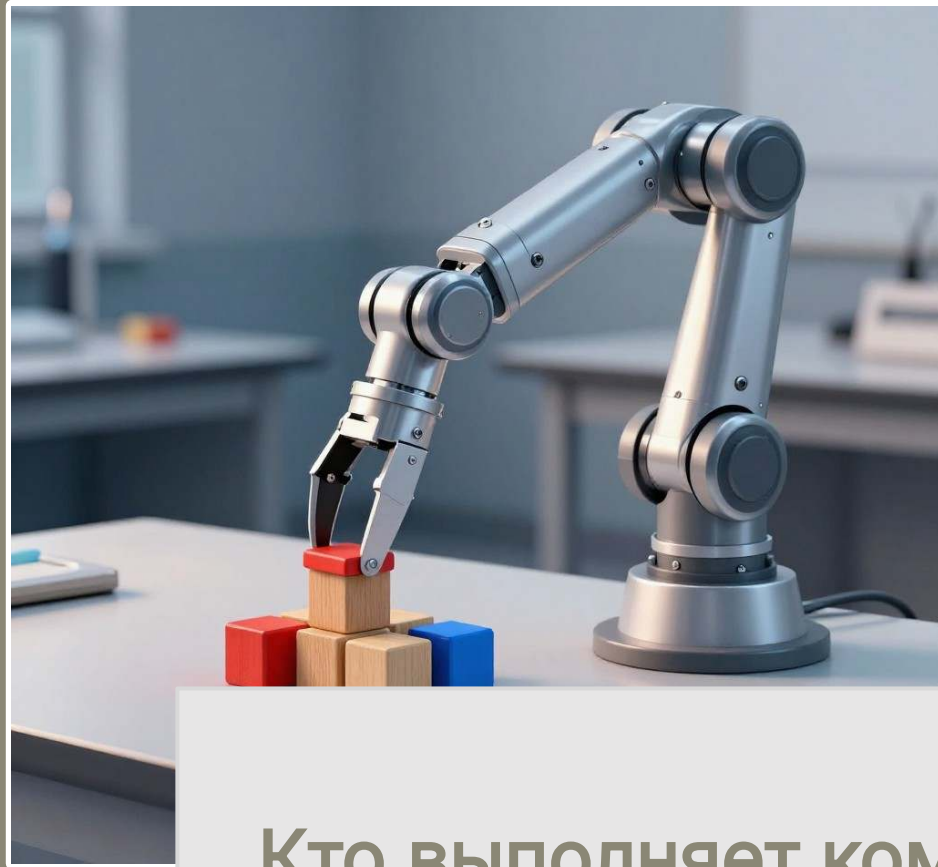
Алгоритм должен приводить к ответу или завершённому действию. Если после действий непонятно, получен ли итог, значит, алгоритм составлен недостаточно точно.



Конечность

Алгоритм завершается за конечное число шагов. То есть выполнение не превращается в бесконечный повтор без результата.

> Исполнитель и система команд



Кто выполняет команды

Исполнитель — это объект, который выполняет команды алгоритма. Это может быть человек, робот, компьютерная программа или учебный робот на уроке информатики. Важно, что исполнитель умеет делать только то, что предусмотрено его возможностями.

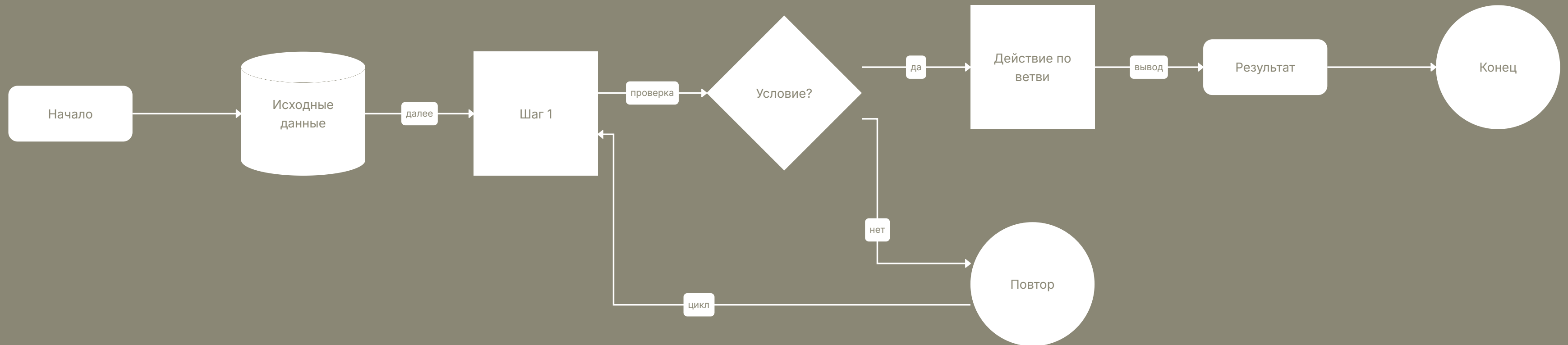


Почему важна «свойственная» система команд

У каждого исполнителя есть система команд — набор действий, которые он способен выполнять. Для точного результата важны состояние исполнителя, условия среды и соответствие команд его умениям.

> Как выполняется алгоритм

Схема отражает формальное выполнение: порядок шагов определяется записью и условиями.



> Формальная запись алгоритма

Формальная запись использует точные формулировки, нумерованные шаги и понятные структуры. Вместо общих слов применяются команды, которые нельзя прочитать «по-разному» в зависимости от настроения исполнителя.

Обычно алгоритм описывают словами, но такая версия может быть недостаточно однозначной. В формальной записи каждое действие имеет ясный смысл и привязано к конкретному месту в последовательности.

Формальность делает алгоритм проверяемым: его можно выполнить вручную или на компьютере. Если шаги записаны точно, результат будет получаться одинаково у разных исполнителей.

> Способы представления алгоритмов

Форма записи	Пример	Где используется
Словесное описание	инструкция	краткие правила
Блок-схема	фигуры и стрелки	ветвления и циклы
Псевдокод	как в программировании	учебные рассуждения
Программа	код на языке	реальное выполнение

Одна задача может быть показана разными способами, но смысл действий сохраняется.

Выбор способа зависит от цели: наглядность удобна для схем, точность — для программы, простота — для псевдокода.

> Где алгоритмы встречаются чаще всего



Наибольшие доли в таких учебных примерах приходится на поиск и навигацию: там постоянно используются фильтры, сортировка, подбор маршрута и рекомендации.

Диаграмма помогает увидеть закономерность: алгоритмические действия повторяются в сервисах, где нужно выбирать путь, данные или варианты ответа.

Учебный пример для демонстрации (данные не являются официальной статистикой)

> Линеинные алгоритмы

Линейный алгоритм состоит из последовательных команд: действий много, но они выполняются один раз и без ветвлений. Переходов по условиям нет, поэтому порядок строго задан.

Чтобы представить такой алгоритм, удобно взять пример из учёбы: открыть тетрадь, записать дату, решить три примера, проверить ответ, закрыть тетрадь. Здесь нет решений «в зависимости от ситуации».

Линейные алгоритмы хорошо подходят для первых задач и для тренировки аккуратности. Если поменять порядок шагов, результат может измениться, поэтому они помогают понять важность точного следования записи.

> Алгоритмы с ветвлением



Условие и два пути

Ветвление проверяет условие и выбирает дальнейший маршрут выполнения. При истинности выполняется одна группа команд, при ложности — другая. Так алгоритм подстраивается под ситуацию.



Пример из повседневной жизни

Если температура ниже 0 °С, алгоритм предлагает надеть тёплую одежду. Если условие не выполняется, выбор меняется. Одна и та же запись помогает действовать по-разному.

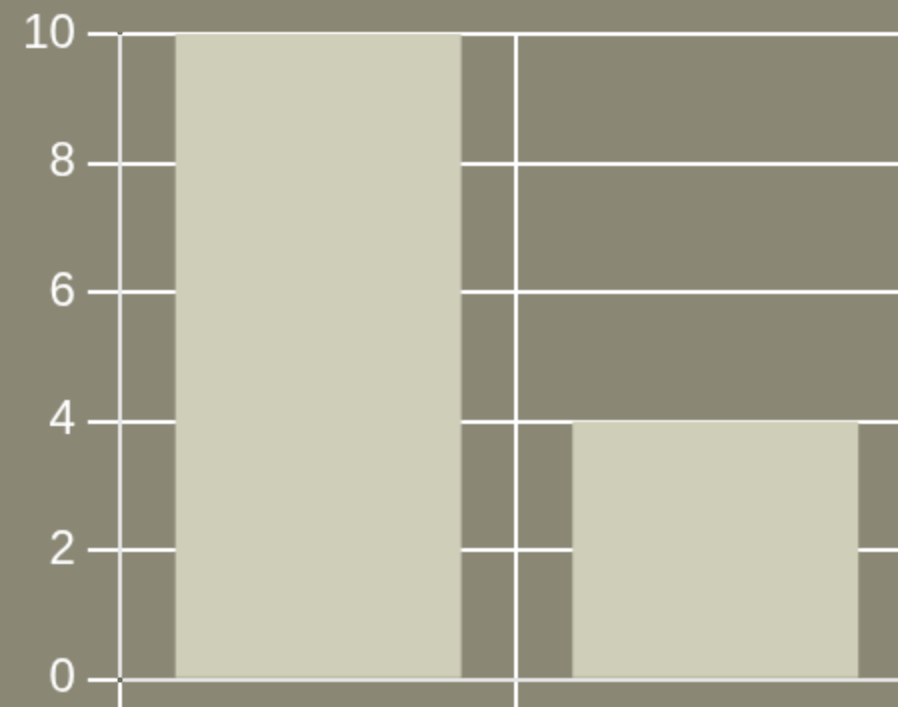


Проверка для чисел

Когда нужно различать чётные и нечётные значения, условие проверяет делимость на 2. Если делится без остатка — выводится «чётное», иначе — «нечётное». Результат зависит от входных данных.

> Зачем нужен цикл в алгоритме

Сравнение длины записи: линейный и циклический варианты (учебная иллюстрация)



Линейный (10 повторов отдельными командами)
Циклический (запись через повтор)

Цикл нужен, когда одинаковое действие повторяется много раз. Вместо длинной цепочки команд используется конструкция повторения с понятным условием окончания.

По сравнению с линейной записью цикл сокращает объём текста алгоритма и делает его устойчивее: меньше мест, где можно ошибиться при наборе.

Учебная модель по примеру «10 одинаковых упражнений», без реальных измерений; иллюстрация для понимания идеи цикла.

> Блок-схема как способ записи алгоритма

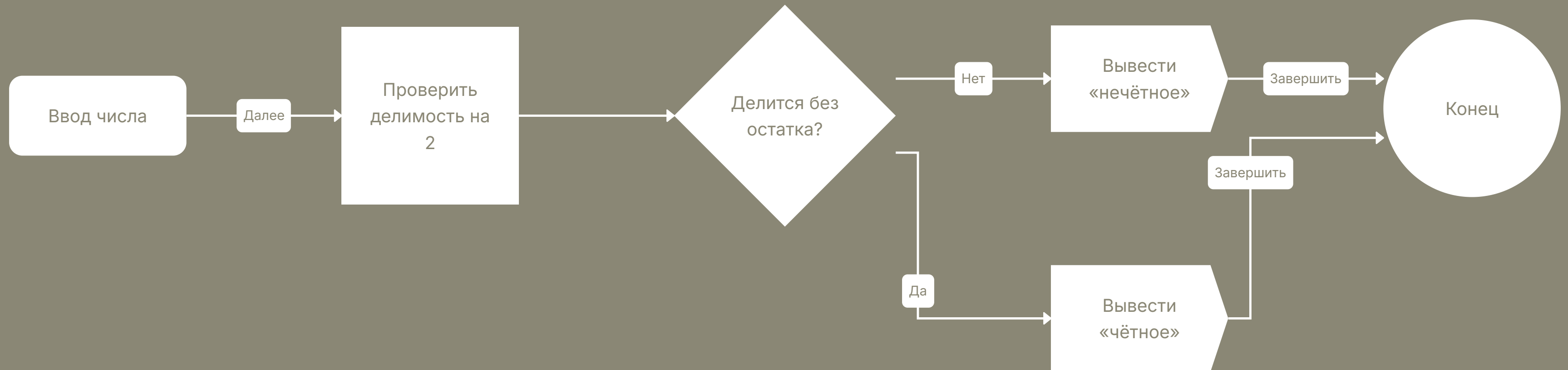


Блок-схема помогает увидеть структуру алгоритма целиком: начало и конец отмечают овалами, действия — прямоугольниками, проверки условий — ромбами. Стрелки показывают, куда

Такой формат особенно удобен для ветвлений и циклов: по одному взгляду понятно, какие ветви возможны и в каком порядке выполняются шаги. Подходящая иллюстрация — схема «условие → две ветви».

> Проверка числа на чётность

Ввод числа → проверка делимости на 2 → вывод результата → завершение.



> Ошибки при формальном исполнении

1

Пропуск шага разрушает порядок действий: алгоритм выполняет задачу не полностью, и результат становится непредсказуемым. Даже один забытый пункт может привести к неправильному выводу или отсутствию ответа.

2

Неверный порядок команд встречается чаще всего, когда шаги «меняют местами». Формальное исполнение строго следует записи, поэтому перестановка действий меняет промежуточные данные и итог.

3

Неоднозначная формулировка вроде «сделать чуть позже» делает команду непроверяемой. Исполнитель не должен догадываться: каждое действие обязано быть описано так, чтобы смысл был одинаковым.

4

Ошибки в условиях перехода тоже критичны: неверно сформулированное условие отправляет выполнение в неправильную ветку. В результате логика алгоритма ломается, а исправить бывает сложнее, чем предотвратить проблему.

> Проверка корректности алгоритма

Проверка типичных значений

Далее прогоняют алгоритм на обычных тестах. Смотрят, совпадает ли полученный результат с ожидаемым, и фиксируют, где возможны расхождения.

Исправление и повторная проверка

Если хотя бы один тест даёт ошибку, алгоритм исправляют. После изменений выполняют повторные проверки тем же набором тестов, чтобы убедиться в исправлении.

Подготовка тестов

Сначала выбирают набор допустимых входных данных. В него включают обычные примеры и крайние случаи, чтобы проверить поведение алгоритма в разных условиях.

Проверка крайних случаев

Затем проверяют ситуации на границах: минимальные или нулевые значения, а также значения «с особенностями» (например, отрицательные, если они допустимы).

> Алгоритм и программа: в чём разница

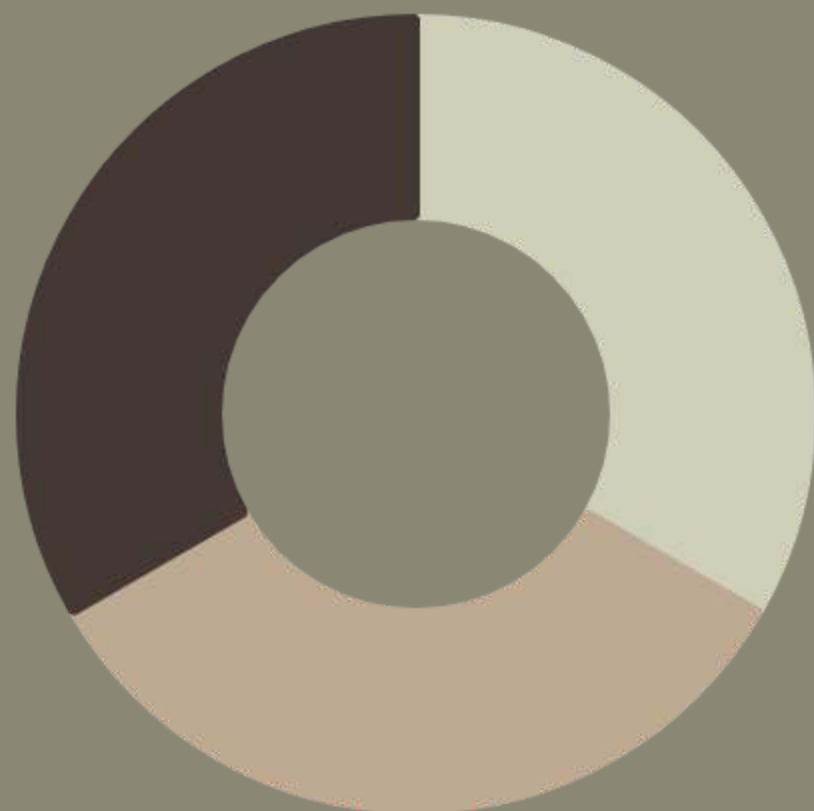
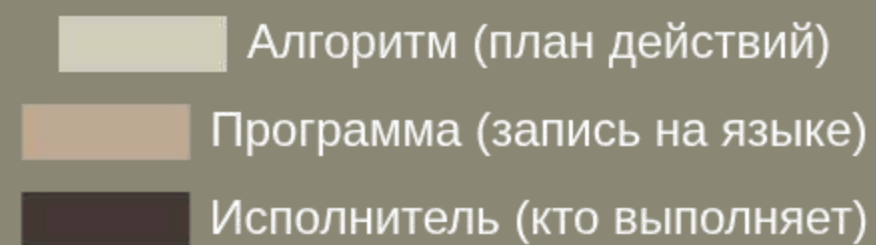


Диаграмма показывает, что алгоритм описывает логику решения, а программа — это его строго записанная версия на языке. Исполнитель выполняет команды, используя свои возможности.

Общее: алгоритм превращается в программу, но только исполнитель реально выполняет действия. Если изменить запись или набор команд, поведение может отличаться, даже при том же замысле.

Учебная схема из школьного курса информатики; концептуальные доли равны, так как речь о различиях, а не о статистике.

> Формальное исполнение в жизни и учёбе



Инструкция к
технике без

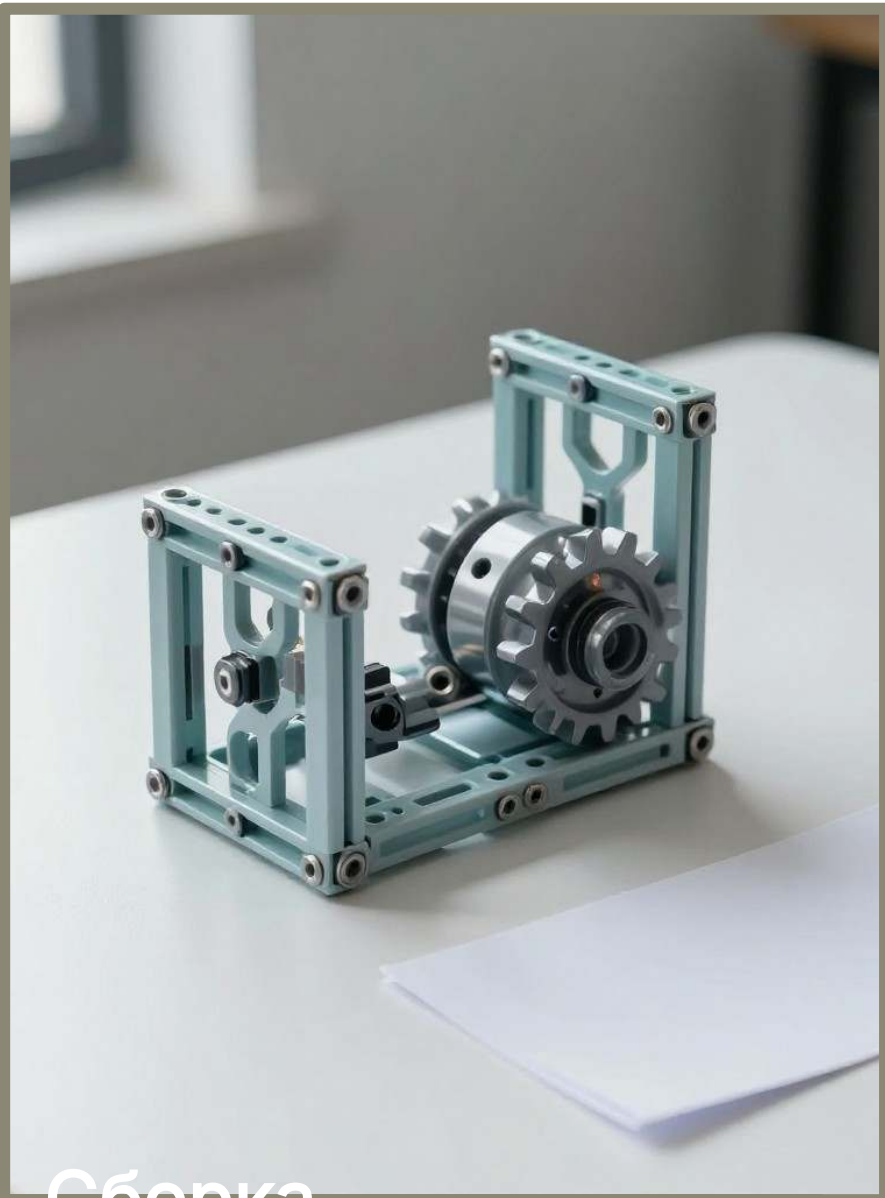
«догадок»

Кладём инструкцию «подключите кабель» и «нажмите кнопку», шаги нельзя менять местами. Производитель предполагает точное выполнение, чтобы устройство заработало безопасно. По такой логике работает и алгоритм: команда должна быть однозначной.



Пошаговый опыт на
лабораторной

В школьной лабораторной работу проводят по процедуре: подготовить реактивы, выполнить действие, записать наблюдения. Если пропустить измерение или перепутать порядок, результаты теряются. Это похоже на формальное исполнение алгоритма.



Сборка
конструктора по

схеме

Но на самом деле детали «не складываются». По карточке сборки видно, что и куда нужно поставить, и в каком порядке затягивать крепления. Следование записи уменьшает ошибки и помогает быстрее получить нужный результат.

> Алгоритм для повседневной задачи

Шаг	Действие	Результат
1	Проверить заряд телефона	Устройство готово к использованию
2	Открыть карту или навигатор	Получен маршрут
3	Выбрать нужный путь	Найдено направление
4	Дойти до места	Цель достигнута
5	Подтвердить прибытие	Задача завершена

Последовательность шагов делает действия предсказуемыми: вы меньше забываете, быстрее ориентируетесь и заранее понимаете, что должно получиться на каждом этапе.

Алгоритмизация снижает риск ошибок: каждое действие привязано к результату. Поэтому проще заметить, где что-то пошло не так, и быстрее исправить.

Пример из повседневной жизни с опорой на идеи алгоритмизации из школьного курса информатики.

> Точность, которая приводит к правильному результату

Алгоритм — это точный план действий, а формальное исполнение — выполнение плана без отклонений от записи. Свойства, способы записи, ветвления и циклы делают решения понятными и проверяемыми.