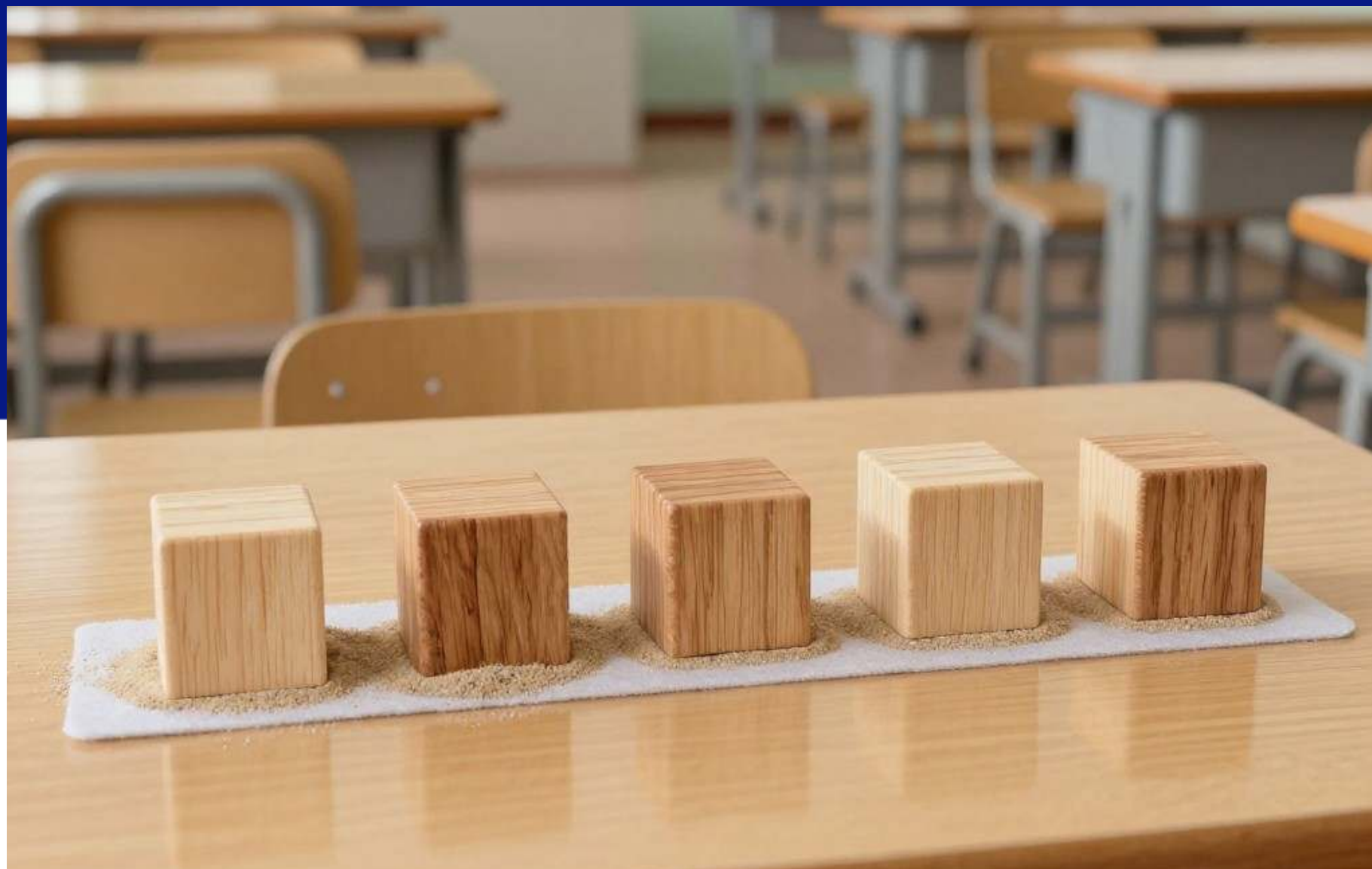


Основные алгоритмические конструкции



Алгоритм — это чёткие шаги. Разберём последовательность, ветвление и цикл на знакомых примерах.

Зачем изучают алгоритмические конструкции в 7 классе

Эти конструкции лежат в основе многих программ и сервисов. Последовательность задаёт порядок шагов, ветвление выбирает вариант, цикл помогает повторять действия. Благодаря им проще записывать алгоритмы, находить ошибки и создавать простые программы.



Последовательность: команды выполняются по порядку

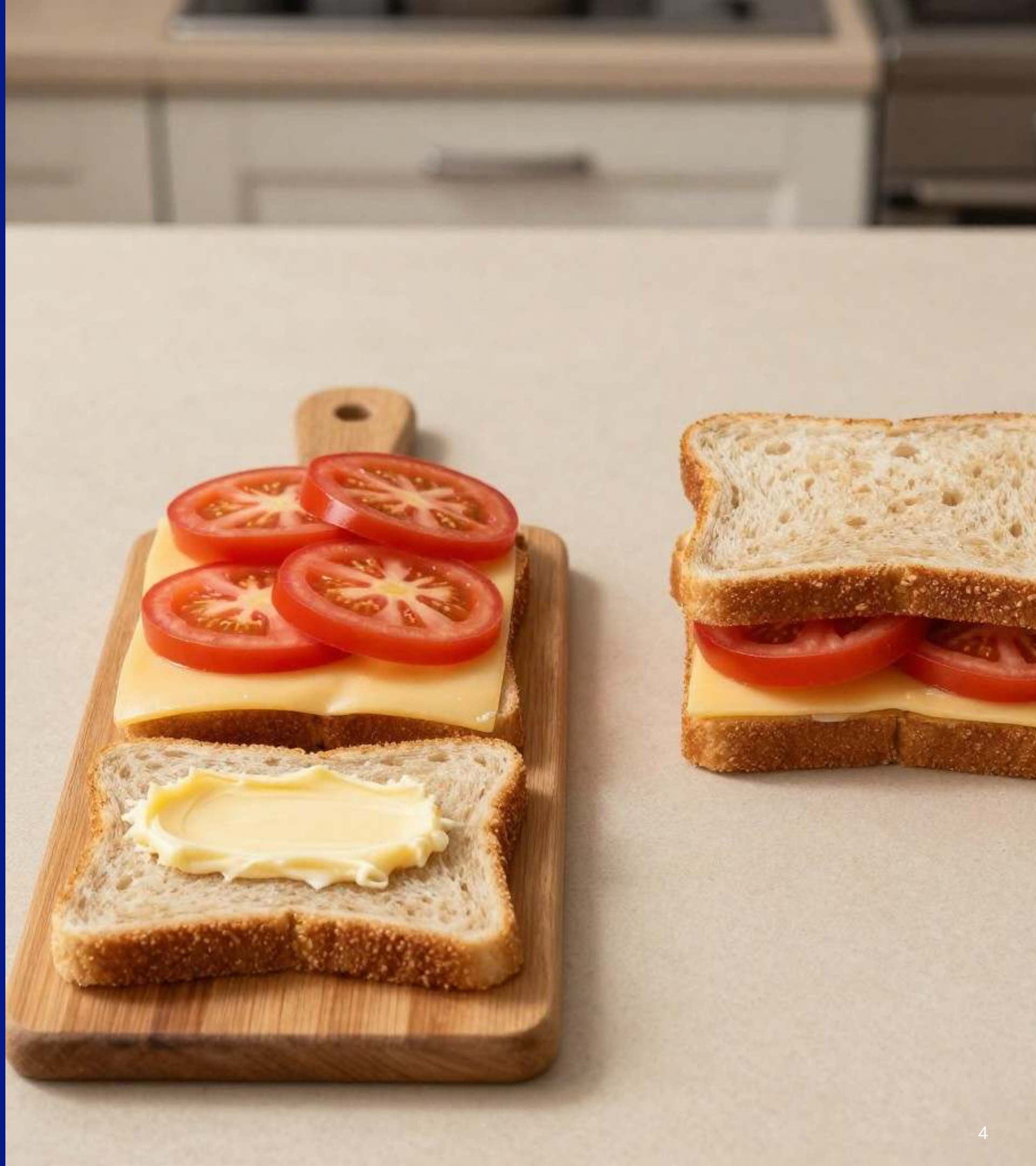
Последовательность — это выполнение команд одна за другой, без пропусков и возвратов. Каждый следующий шаг опирается на предыдущий, поэтому порядок важен для всего результата.

Например: включить компьютер, открыть программу, ввести текст и сохранить файл. Если сделать шаги в другом порядке, файл может не сохраниться или текст окажется неверным.

Такую логику легко проверить на практике: если программа «не понимает» действие, значит, шаг был выполнен не вовремя. Поэтому последовательность — самая простая и понятная конструкция.

Последовательность в знакомых действиях

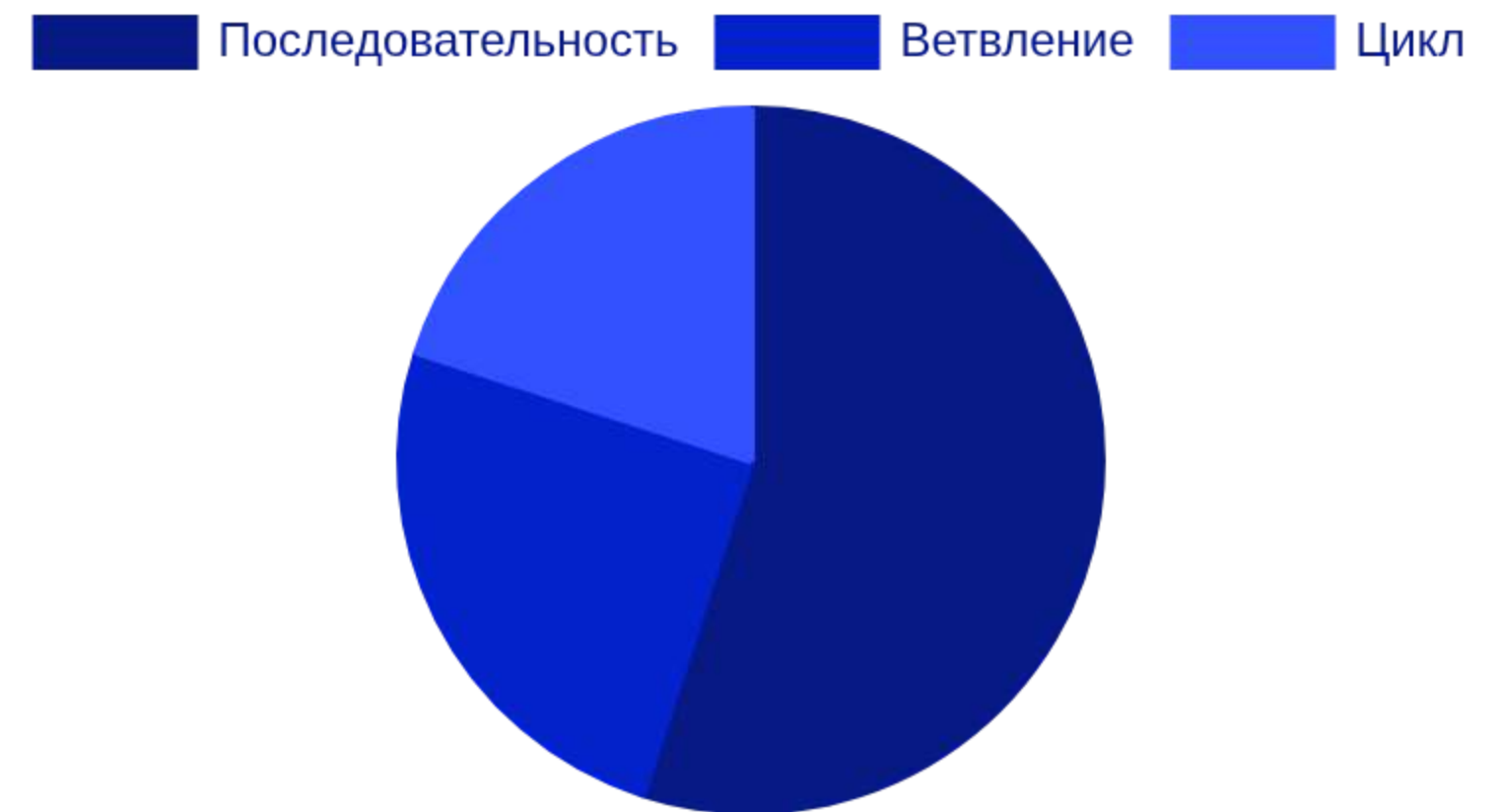
1. Последовательность хорошо заметна в приготовлении бутерброда: возьмите хлеб, добавьте начинку, накройте вторым куском и положите на тарелку. Если пропустить шаг, процесс будет «сломаться».
2. Похожий порядок есть и в сборке школьного рюкзака: сначала положить учебник, затем тетрадь, потом пенал и сменную обувь. В правильной очередности собираться быстрее и меньше забываешь.



Три конструкции алгоритмов и их роль

В одних алгоритмах чаще нужен порядок, в других — выбор по условию, а в третьих — повторение одних и тех же действий.

Чем больше в задаче повторов, тем заметнее роль цикла. Если возможны разные варианты действий, появляется ветвление.



Обобщённая схема для обучения (оценка долей не основана на статистике).

Ветвление: выбор по условию

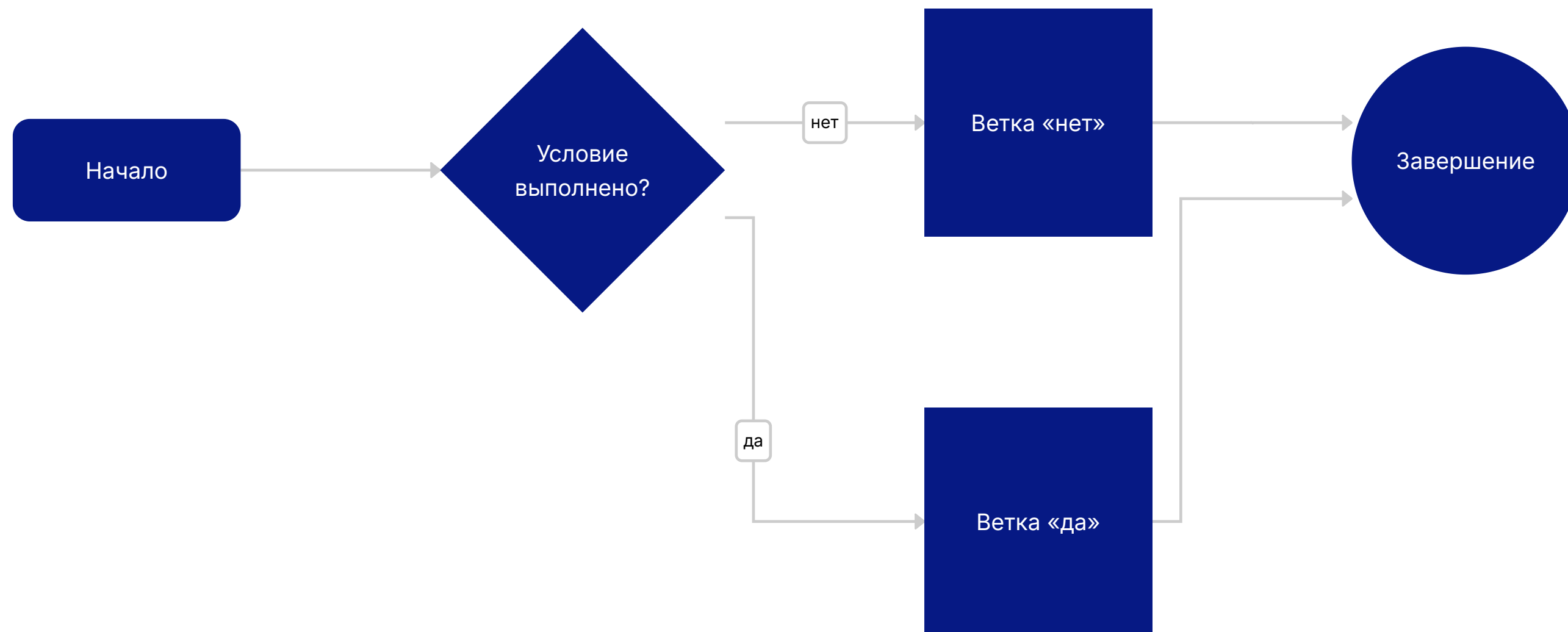
Ветвление используют, когда алгоритм должен выбрать один из нескольких путей. Сначала проверяется условие, а потом выполняется подходящая последовательность команд.

Например: если идёт дождь, взять зонт; если ответ верный, поставить балл; если пароль совпал, открыть доступ. В каждом случае результат зависит от ответа «да/нет».

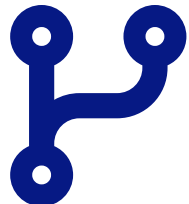
Такой подход помогает создавать понятные правила в программах: от проверок в тестах до входа в приложения. Без ветвления алгоритм не сможет реагировать на разные ситуации.

Как работает ветвление

Схема процесса: проверяем условие и идём по разным веткам в зависимости от ответа.



Сложные условия и вложенные проверки

	Составное условие Иногда нужно учесть сразу несколько признаков: например, «число больше 0 и меньше 10». Тогда в одном решении используются слова «и» и «или».
	Вложенное ветвление Вложенность появляется, когда внутри одной проверки выполняется ещё одна. Так можно точнее описать правила, например: сначала проверяем группу, затем конкретный вариант.
	Фильтры и поиск В тестах и фильтрах поиск часто строится на нескольких проверках. Алгоритм сначала отбирает по условию, а затем при необходимости уточняет результат.
	Правила доступа При входе в системы проверяются права пользователя. Сначала выясняют базовое условие, а затем по дополнительной проверке открывают нужную возможность.

Сравнение последовательности, ветвления и цикла

Конструкция	Назначение	Пример из школьной жизни	Главное свойство
Последовательность	Действия по порядку	Открыть тетрадь и записать дату	Шаги идут строго один за другим
Ветвление	Выбор по условию	Взять зонт при дожде	Действия зависят от проверки
Цикл	Повторение действий	Решить 10 одинаковых примеров	Одно действие выполняется много раз

Учебное обобщение по теме алгоритмических конструкций (без внешних данных).

В таблице показано, чем отличаются конструкции и какие задачи решает каждая из них.

Последовательность отвечает за «что дальше», ветвление — за «что выбрать», а цикл — за «сколько раз повторить». Вместе они покрывают большинство школьных задач.

Цикл: повторение одинаковых действий

Цикл нужен, когда одно и то же действие выполняется несколько раз. Он помогает считать, рисовать узоры, проверять варианты и обрабатывать список элементов.

Вместо десяти одинаковых команд записывается один цикл, а затем программа выполняет повтор автоматически. За счёт этого алгоритм становится короче и легче для чтения.

Цикл особенно удобен, если повторов много или они повторяются по одинаковому правилу. Тогда вместо длинной «простыни» шагов появляется понятный повторяющийся блок.

Когда цикл особенно полезен

Если число повторов небольшое, разница в записи почти не заметна. Но с ростом количества шагов удобнее выделить повторяющееся в цикл и не копировать команды вручную.

По мере увеличения числа повторений ценность цикла становится очевиднее: алгоритм короче, легче читать и меньше шансов ошибиться при переписывании одинаковых действий.



Подготовлено на основе примеров из учебного курса (повторы: 3, 5, 10, 100), без внешних данных.

Два вида циклов в алгоритмах



Цикл с параметром

Цикл с параметром выполняется заданное число раз: заранее известно, сколько шагов нужно. Например, при рисовании «7 квадратов» повторение можно записать одной конструкцией, а не семью одинаковыми командами.

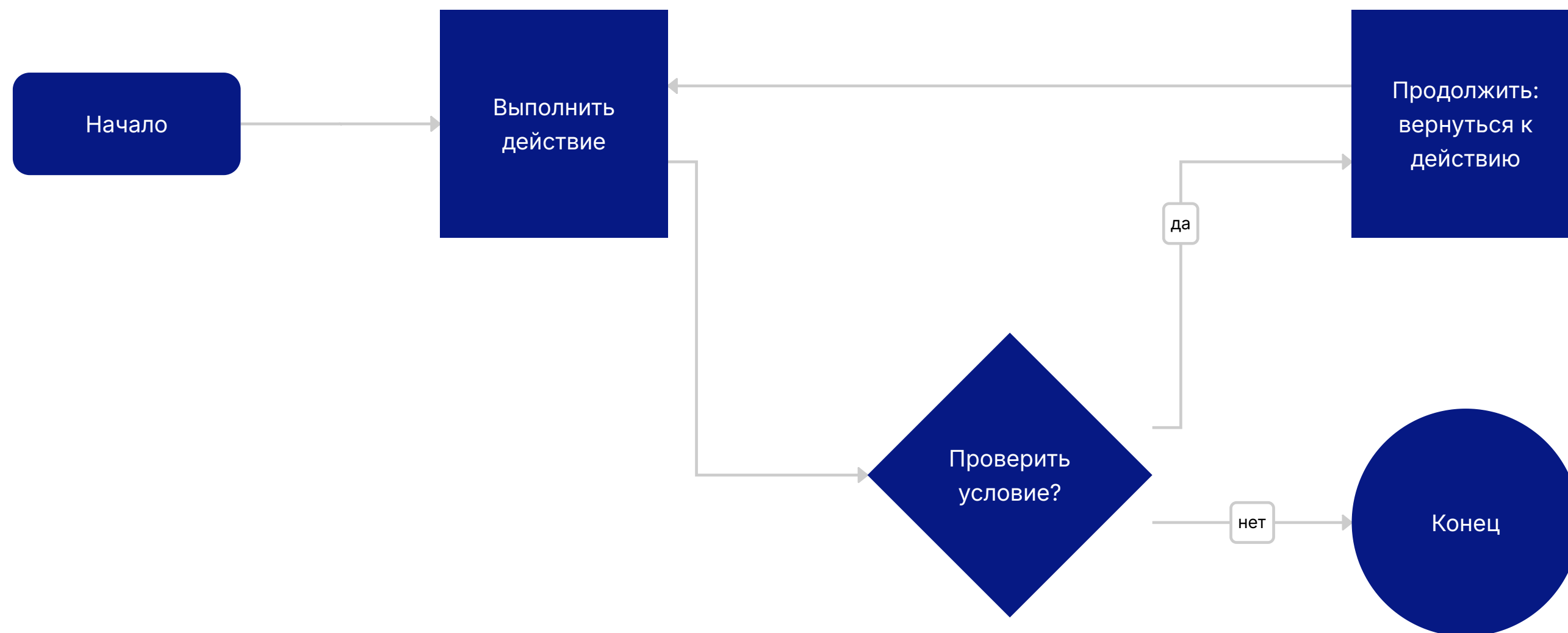


Цикл с условием

Цикл с условием работает, пока условие остаётся истинным. Пример: повторять ввод пароля до тех пор, пока пароль не станет верным. Алгоритм завершится только при достижении нужного результата.

Повторение с проверкой: «попробуй — проверь — продолжай»

Источник логики: алгоритмическая конструкция «цикл» с условием продолжения (повтор назад при «да»).



Алгоритм как набор блоков

Собираем программу по частям

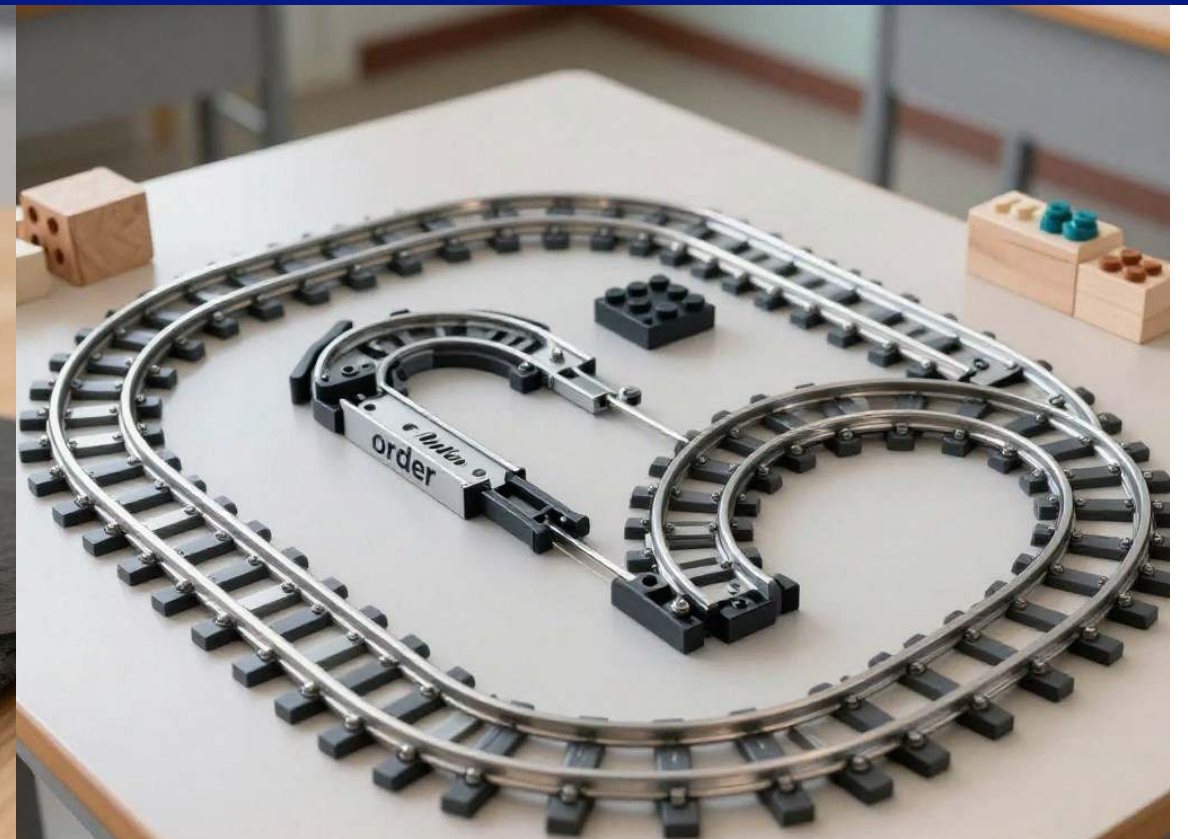
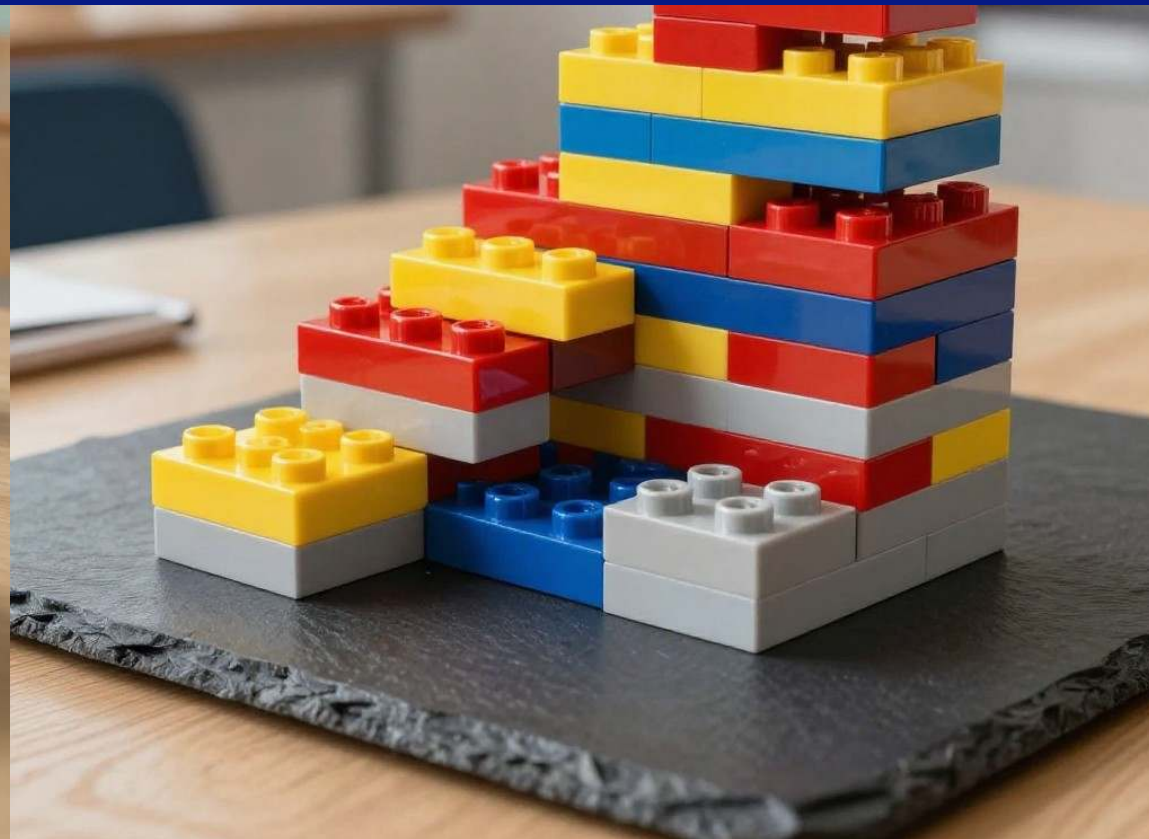
Представьте, что программа — это цепочка деталей: последовательность задаёт порядок, ветвление выбирает путь, цикл повторяет нужные шаги. Когда части собраны правильно, получается один целостный алгоритм, который легко проверять и улучшать.

Конструктор LEGO для логики

Как в конструкторах: отдельные блоки соединяются и образуют модель, так и в алгоритмах разные конструкции «вкладываются» друг в друга. Отдельный блок проще понять, чем всю программу сразу, поэтому сборка идёт поэтапно.

Компоненты держат порядок и смысл

Одна и та же цель достигается разными способами, но правила сборки остаются: сначала определяем, что идёт по порядку, где нужен выбор, и где требуется повторение. Тогда алгоритм становится понятным и для автора, и для тех, кто его читает.



Вложенные алгоритмы

Вложенность означает, что внутри одной конструкции находится другая. Например, внутри цикла может быть ветвление: сначала повторяем шаг, а затем выбираем действие по условию, и так каждый раз.

Точно так же ветвление может включать цикл: при выполнении условия начинается повторяющееся действие. Такая структура полезна в заданиях, где выбор запускает дальнейшую серию проверок или действий.

В играх вложенные конструкции помогают управлять персонажем и логикой событий: иногда нужно многократно выполнять одинаковые попытки, а иногда — после каждой попытки проверять условия успеха.

Меню программ и таблицы тоже часто используют вложенность: выбор пункта определяет, какие повторяющиеся шаги выполняются дальше. Благодаря этому алгоритм остаётся структурным, а не превращается в набор случайных команд.

Типичные ошибки в алгоритмах

Ошибка	Пример	Результат
Пропуск шага	Сохранить файл до ввода данных	Получается пустой или старый файл
Неверное условие	«Если число ≥ 0 » вместо « > 0 »	Выбирается не та ветка, ответ смещается
Бесконечный цикл	Условие всегда истинно	Программа не завершает работу

Обобщение типичных учебных ошибок при записи алгоритмов.

Алгоритмы легко «сломать» одной неправильной строкой: пропуск шага, неверное условие или бесконечное повторение приводят к неправильному результату или зависанию программы.

Большинство проблем связано с логикой переходов: внимательно проверяйте порядок, точность условий и то, что цикл имеет понятный путь к завершению.

Сколько команд экономит цикл

Цикл описывается компактно: одна команда повторения заменяет множество одинаковых действий. Поэтому при росте числа повторов экономия выражается сильнее, чем в задачах на 2–3 шага.

Разница между «копированием команд» и «одной командой цикла» растёт вместе с количеством повторов: при 20 действиях обычная запись получается в двадцать раз длиннее.



Примерная оценка по числам повторов из описания слайд-урока (1, 5, 10, 20), без внешних данных.

Алгоритмические конструкции вокруг нас

Маршрут: по шагам

Навигатор строит маршрут как последовательность действий: получить старт и цель, определить варианты, показать очередность поворотов и действий.

Выбор при пробках

Когда условия меняются, используется ветвление: при загрузженности дорог навигатор выбирает альтернативный путь и пересобирает маршрут по новому условию.

Пересчёт и повторение

Алгоритм может многократно пересчитывать параметры маршрута: поиск, оценка, сравнение вариантов повторяются, пока не будет достаточно точный результат.

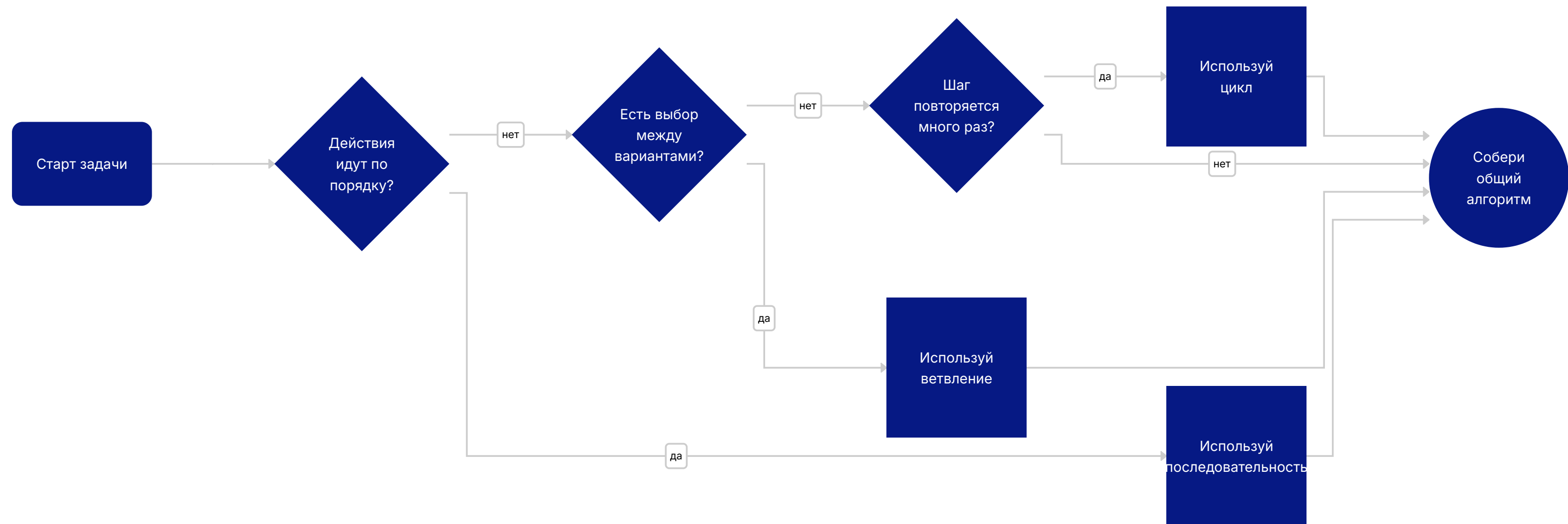
Игры и фильтры

В играх правила победы проверяются условиями, а попытки часто повторяются циклом. В поиске и фильтрах также используются проверки и повторяемые шаги по элементам.



Как выбрать нужную конструкцию

Источник логики: критерии выбора между последовательностью, ветвлением и циклом.



Три конструкции — и алгоритм становится понятным

Последовательность задаёт порядок действий, ветвление выбирает один из путей, цикл повторяет команды. Эти конструкции помогают описывать задачи разной сложности, читать блок-схемы и находить ошибки в программах.