

Одна задача обработки массива

Найдём сумму чётных элементов и посчитаем их
количество за один проход по массиву.

Зачем нужны задачи на обработку массива

- Массивы помогают хранить набор однотипных данных — от оценок до расходов. Обработка массива позволяет получать итоги, отбирать элементы по условию и сравнивать значения. На уроках это закрепляет работу с `for` и `if`.



Что такое массив

- Массив — это упорядоченный набор элементов с общим именем. У каждого элемента есть номер, или индекс, по которому к нему обращаются в программе. Обычно первый индекс равен 0.
- Чтобы работать с массивом, важно понимать, сколько в нём элементов. От этого зависит, как далеко пройдёт цикл обработки. Без точного знания длины легко ошибиться в количестве проверок.
- Ещё один принцип — одинаковый тип данных у элементов массива. Например, в одном массиве хранят числа одного вида: целые, вещественные или, если нужно, одинаковые по смыслу значения. Это делает работу предсказуемой.

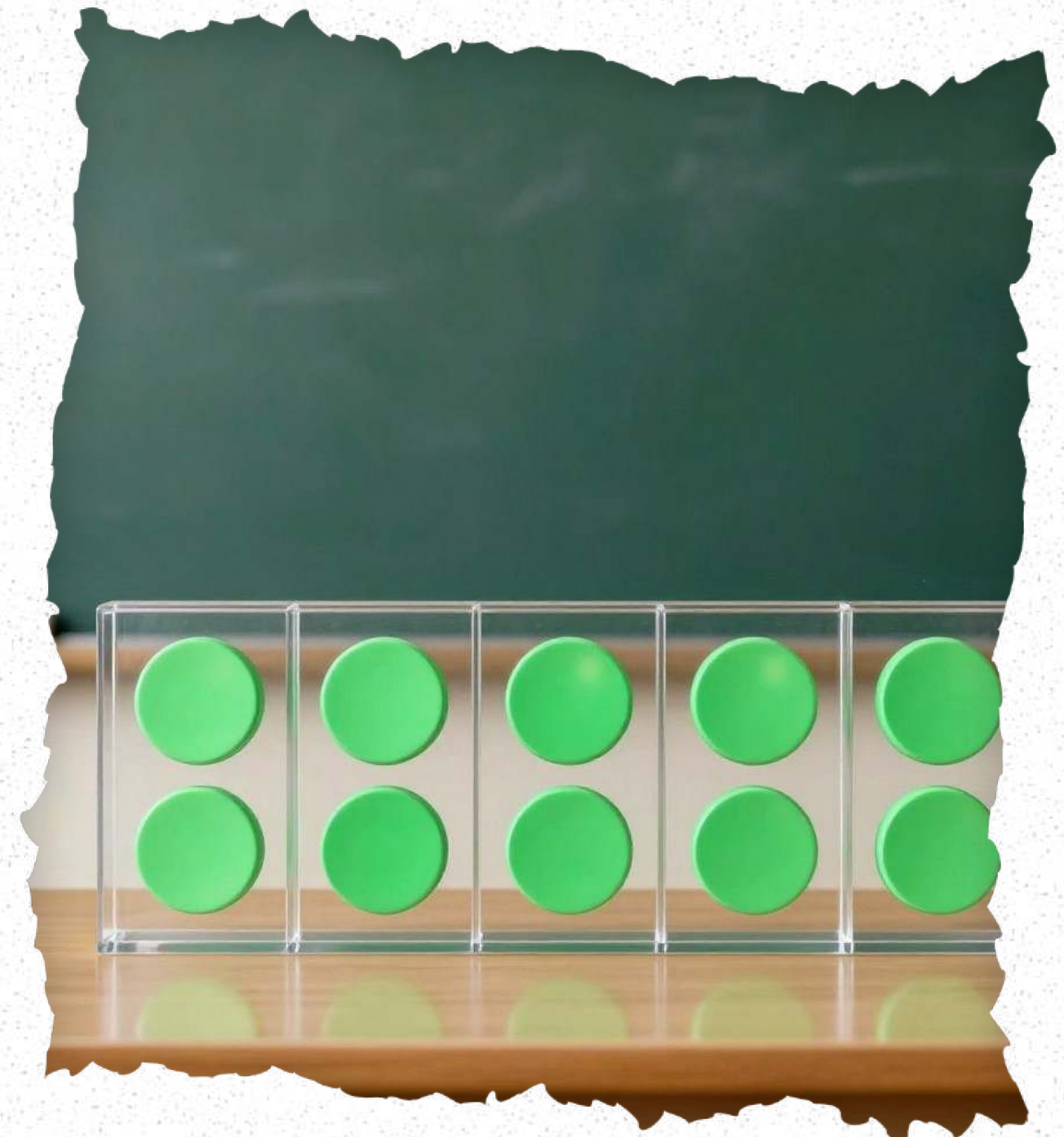
Пример массива целых чисел

Данные и индексы

Рассмотрим массив: 5, 12, 7, 8, 3, 10. В нём 6 элементов, а индексы идут от 0 до 5. Так программа «видит» каждый элемент по его позиции.

Отбор чётных чисел

Чётные в этом наборе — 12, 8 и 10. На иллюстрации выделяются ячейки массива и соответствующие им индексы, чтобы было ясно: условие применяется к каждому элементу по очереди.



Формулировка задачи

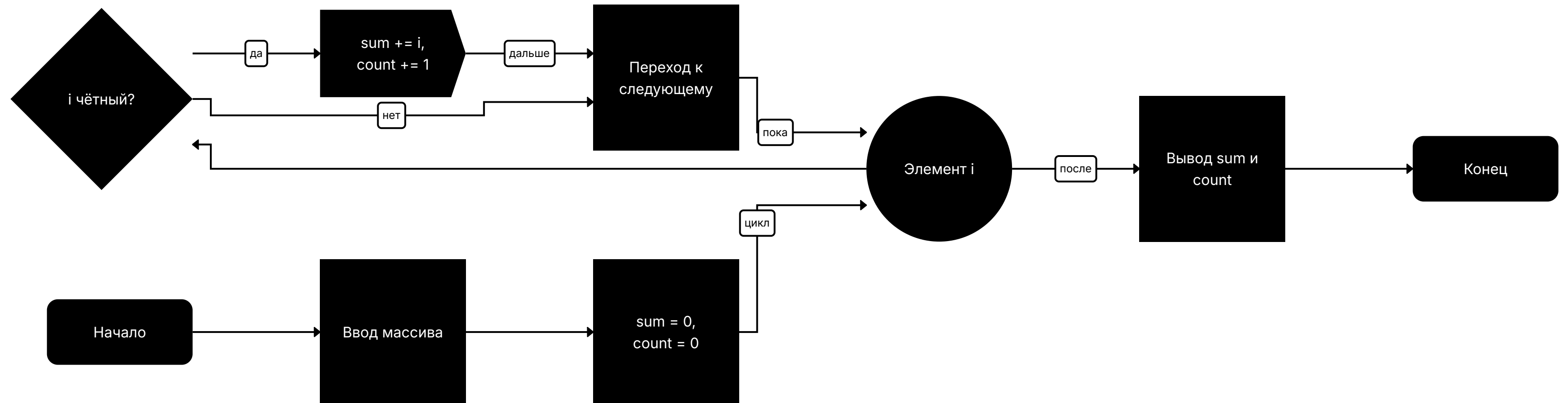
Дан массив целых чисел. Нужно пройти по нему и найти сумму всех чётных элементов. То есть сложить только те значения, которые делятся на 2 без остатка.

Параллельно требуется посчитать, сколько таких чётных элементов оказалось в массиве. В результате будет два числа: итоговая сумма и количество подходящих элементов.

Если в массиве нет ни одного чётного числа, ответ всё равно определён: сумма будет равна 0, а количество также станет 0. Это важно для корректности программы в любых данных.

Алгоритм: сумма и количество чётных

Обработка массива целых чисел по очереди с условием чётности



Как меняются сумма и счётчик на примере

У нечётных чисел вклад в сумму равен нулю, а счётчик не меняется. Только чётные элементы добавляют значение и увеличивают число найденных.

Для массива 5, 12, 7, 8, 3, 10 чётные — 12, 8, 10. Их сумма 30, а количество равно 3, потому что остальные элементы условие не выполняют.

Пример из условия: 5, 12, 7, 8, 3, 10

Пошаговая обработка массива

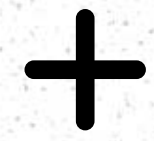
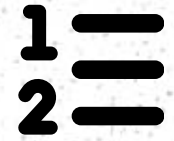
Таблица показывает решение наглядно: по одному проходу по элементам обновляются сумма и счётчик. После последнего элемента получаются $sum = 30$ и $count = 3$.

Каждый элемент массива проверяется ровно один раз, поэтому итог не зависит от того, как выглядят числа, а зависит только от того, какие из них чётные.

Элемент	Чётный?	Вклад в сумму	Счётчик после шага
5	нет	0	0
12	да	12	1
7	нет	0	1
8	да	8	2
3	нет	0	2
10	да	10	3

Последовательность: 5, 12, 7, 8, 3, 10

Переменные, которые нужны в программе

							
	Массив В программе хранится сам массив целых чисел. По нему выполняется цикл: элементы берутся по индексу по очереди, чтобы ни один не пропустить. Массив задаётся на входе.		Переменная суммы Сумма накапливает вклад чётных чисел. Перед началом работы ей присваивают 0, а затем при каждой удачной проверке к ней добавляют очередной чётный элемент.		Переменная счётчика Счётчик показывает количество найденных чётных элементов. Он тоже начинается с 0 и увеличивается на 1 только тогда, когда текущий элемент проходит условие чётности.		Текущий элемент Текущий элемент берётся из массива на каждом шаге цикла. Именно его значение проверяют на чётность. Благодаря этой переменной логика одинаково применяется ко всем элементам по очереди.

Проверка чётности числа

Чётное число делится на 2 без остатка. Поэтому в программе проверку удобно записывать через остаток от деления: число $\% 2 == 0$ означает, что остатка нет, и число чётное.

Оператор $\%$ в выражении возвращает остаток от деления. Если остаток равен 0, условие выполняется, и число можно учитывать в сумме и счётчике как чётное.

Это правило работает для всех целых чисел, в том числе для отрицательных. Например, -4 и -12 тоже делятся на 2 без остатка, значит они чётные.

Сравнение чётных и нечётных элементов

Для массива 5, 12, 7, 8, 3, 10 чётных столько же, сколько нечётных: по 3. Это объясняет, почему итог определяется только теми, кто проходит условие.

Когда доля чётных меняется, меняются и сумма, и количество. Но порядок элементов не влияет: каждый элемент либо добавляет вклад, либо нет.

Данные: массив 5, 12, 7, 8, 3, 10 (пример из урока).

Псевдокод решения

Инициализация

Задаём сумму = 0 и счётчик = 0. Эти переменные будут накапливать результат, пока мы просматриваем массив.

Цикл по массиву

Для каждого элемента массива по очереди выполняем проверку на чётность. Текущим значением становится элемент с очередным индексом.

Обновление при чётности

Если элемент чётный, то добавляем его к сумме и увеличиваем счётчик на 1. Если нечётный, изменений нет.

Вывод результата

После завершения цикла выводим оба значения: итоговую сумму чётных элементов и их количество.



Проверка алгоритма на разных массивах

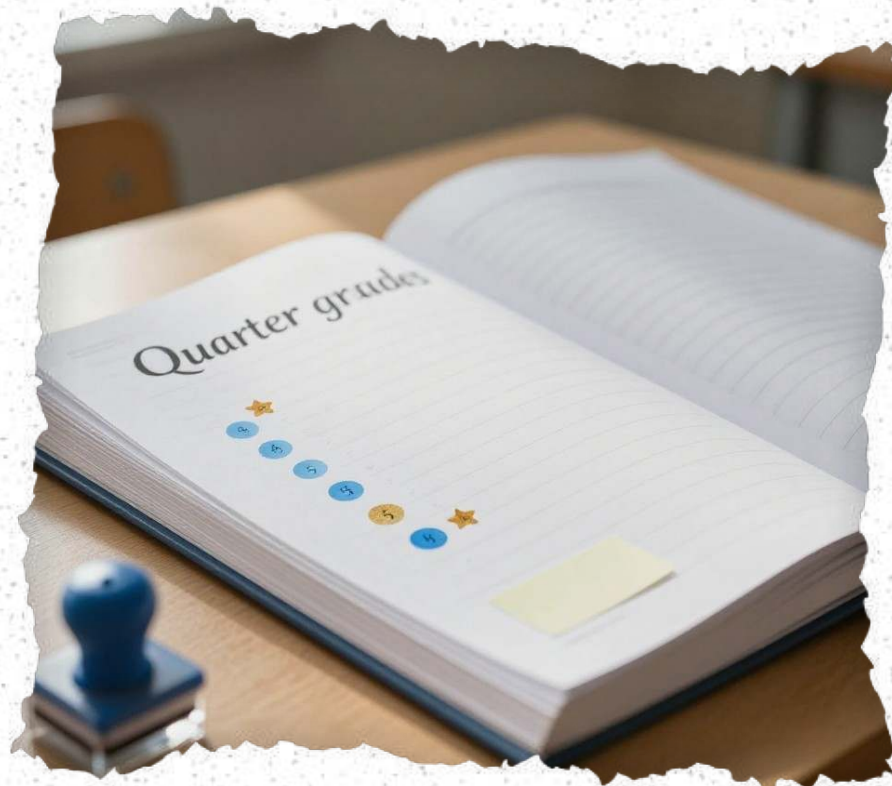
Случай	Сумма чётных	Кол-во чётных
Все чётные	(сумма всех элементов)	n
Все нечётные	0	0
Смешанный	(сумма чётных)	(число чётных)

Результат логического анализа условий алгоритма (без привязки к конкретным числам).

Сравним три ситуации: только чётные, только нечётные и смешанный массив. Итоги показывают корректность и независимость от порядка элементов.

Алгоритм верно работает для любых целых чисел: если чётных нет — сумма и количество равны нулю, иначе учитываются только элементы, прошедшие проверку.

Связь с реальными данными: короткий итог из длинного списка



Оценки за четверть

В классном журнале десятки чисел: отметки за контрольные. Если нужно посчитать, сколько «четвёрок» и «пятёрок» или сложить подходящие баллы, алгоритм действует как фильтр и копилка. Из списка получается понятная цифра без ручного пересчёта.



Шаги и активность

Представьте неделю тренировок: число шагов за каждый день записано в ряд. Допустим, нас интересуют дни, где шагов достаточно — например, значения, удовлетворяющие условию. Просмотр проходит один раз, а итог выводит сумму и количество таких дней.

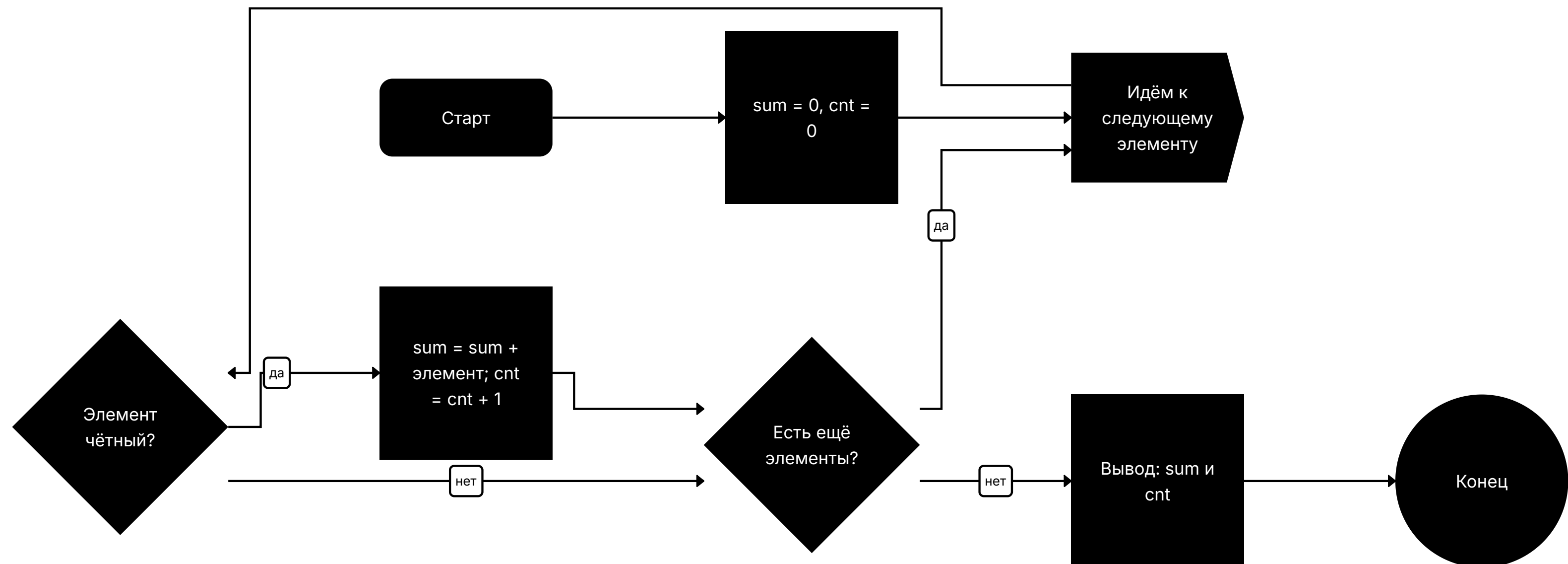


Покупки в магазине

В корзине много товаров, и каждый ценник — отдельное число. Если требуется посчитать общую стоимость товаров с нужным признаком (например, чётные цены) и узнать, сколько их было, массив превращается в два ответа. Это помогает быстро увидеть результат и не потеряться в деталях.

Цикл обработки без пропусков

Блок-схема показывает последовательный просмотр элементов массива и проверку условия чётности для каждого из них.



Сколько ресурсов требует алгоритм

● Алгоритм проходит по массиву ровно один раз, поэтому время работы растёт вместе с числом элементов. Для n чисел выполняется примерно n проверок чётности.

● Память используется экономно: не нужно хранить дополнительные списки. Достаточно нескольких переменных — суммы, счётчика и текущего элемента во время обхода.

● Поэтому алгоритм масштабируется предсказуемо: добавили больше чисел — получили больше проверок, но структура программы остаётся такой же простой и понятной.

Рост числа проверок при увеличении массива

На графике видно линейное увеличение: сколько элементов добавляется, столько же становится проверок на чётность. Это и делает поведение алгоритма предсказуемым.

Линейный рост говорит о том, что время работы напрямую зависит от длины массива. Дополнительные операции появляются только когда число чётное, но закономерность остаётся той же.

Данные: оценки числа проверок как n (для массивов длиной 5, 10, 20 и 50).

Типичные ошибки при решении

Самая частая ошибка — не обнулить сумму и счётчик до начала цикла. Тогда старые значения «прилипают» к новому ответу, и даже правильные проверки дают неправильную итоговую сумму.

Другая проблема — проверять только часть массива и пропускать элементы. Например, остановиться раньше или не пройти до последнего индекса. В таком случае количество чётных будет меньше, чем должно быть.

Иногда путают смысл переменных: записывают сумму в счётчик или наоборот. Нужно помнить: сумма — это накопление значений, а счётчик — только количество элементов, которые удовлетворили условию.

Также приводит к ошибкам неверный индекс или выход за границы массива. Если цикл идёт мимо первого или последнего элемента, алгоритм либо пропустит число, либо завершится с проблемой.

Контрольный пример ответа

Для массива 2, 4, 7, 9, 12 чётные элементы — 2, 4 и 12. Их сумма равна 18, количество равно 3. Проверьте, совпадает ли результат вашей программы.

Количество чётных равно 3, потому что «да» встречается три раза. Сумма получается как $2+4+12=18$ — это удобная контрольная точка для поиска ошибок.

Элемент	Чётный?
2	да
4	да
7	нет
9	нет
12	да

Проверка по условию «делится на 2 без остатка».

Итоги работы с массивом

Одна задача обработки массива сводится к последовательному просмотру элементов, проверке условия и обновлению двух переменных. В задаче про чётные числа итогом становятся сумма и количество подходящих элементов.