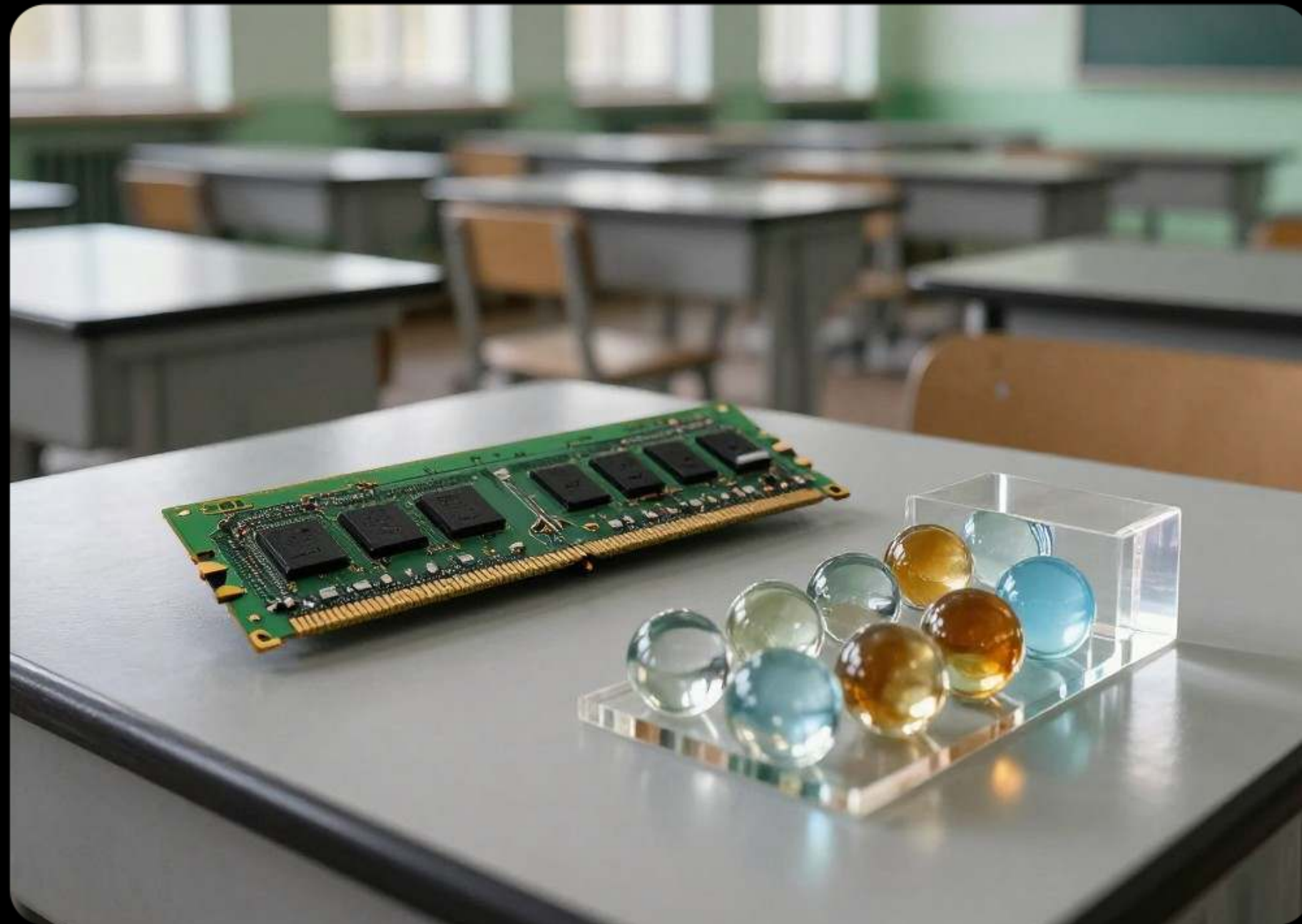




# Хранение + вещественных чисел в памяти компьютера

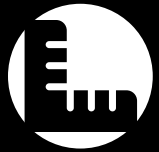
Вещественные числа — двоичные приближения по стандарту IEEE 754: знак, порядок, мантисса.

# Почему это важно в школьной информатике



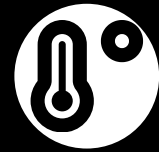
Вещественные числа встречаются в программах почти везде: время, координаты, температура, скорость, расстояния и результаты вычислений. В школьном курсе это связывает двоичную запись, кодирование данных и точность алгоритмов.

# Где применяются вещественные числа



## Измерения и размеры

Длины, массы и другие величины, которые меняются плавно, естественно задавать вещественными числами. Так компьютер хранит «почти как в жизни», но с ограниченной точностью, задаваемой форматом.



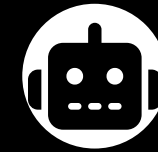
## Температура и физические величины

В экспериментах и моделировании температуры, давления и параметров среды удобно описывать непрерывно. Машинное представление позволяет работать с данными автоматически, хотя точное совпадение с формулами недостижимо.



## Координаты на карте и в трёхмерии

Координаты объектов на плоскости и в пространстве — типичные вещественные значения. В географии и навигации важно соотношение диапазона и точности, чтобы различать близкие точки.



## Робототехника и управление

Скорость, траектория и углы поворота требуют вещественных чисел для расчётов управления. При выборе формата учитывают, насколько допустима ошибка, и как она повлияет на устойчивость алгоритма.



## Компьютерная графика

В графике координаты, размеры и параметры освещения задаются числами с «плавающей точкой». Чем точнее формат, тем стабильнее рендеринг, но выше требования к памяти и вычислениям.

# Десятичная дробь в двоичной системе получается приближённой

- **Почему 0,1 в памяти не равна «0,1»**

Число 0,1 в десятичной записи кажется «простым», но в основании 2 оно превращается в бесконечную периодическую дробь. Поэтому память не может сохранить точную запись и хранит ближайшее представимое двоичное значение.



- **Ближайшее представимое значение заменяет точную дробь**

Аналогичная ситуация возникает с 0,2, 0,3 и многими другими дробями: их двоичное представление тоже не заканчивается. В результате машинное число может отличаться от того, что мы ожидаем увидеть в учебнике.

# Основные форматы IEEE 754

| Формат  | Биты                            | Точность                              |
|---------|---------------------------------|---------------------------------------|
| float32 | 1 знак, 8 порядок, 23 мантисса  | $\approx 7$ значащих цифр             |
| float64 | 1 знак, 11 порядок, 52 мантисса | $\approx 15\text{--}16$ значащих цифр |

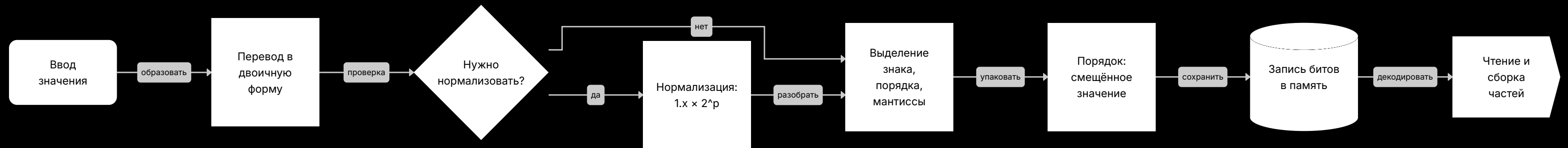
Сравнение форматов float32 и float64 по структуре числа и оценке точности значащих цифр.

Увеличение числа битов (прежде всего мантиссы) повышает точность и диапазон, но увеличивает объём данных.

IEEE 754

# От десятичного числа к двоичному представлению

Логическая последовательность кодирования по структуре IEEE 754



# Три части числа с плавающей точкой



Бит знака фиксирует, положительное число или отрицательное. Это позволяет хранить и сравнивать значения по единому «каркасу» без отдельной логики для знака в алгоритмах.

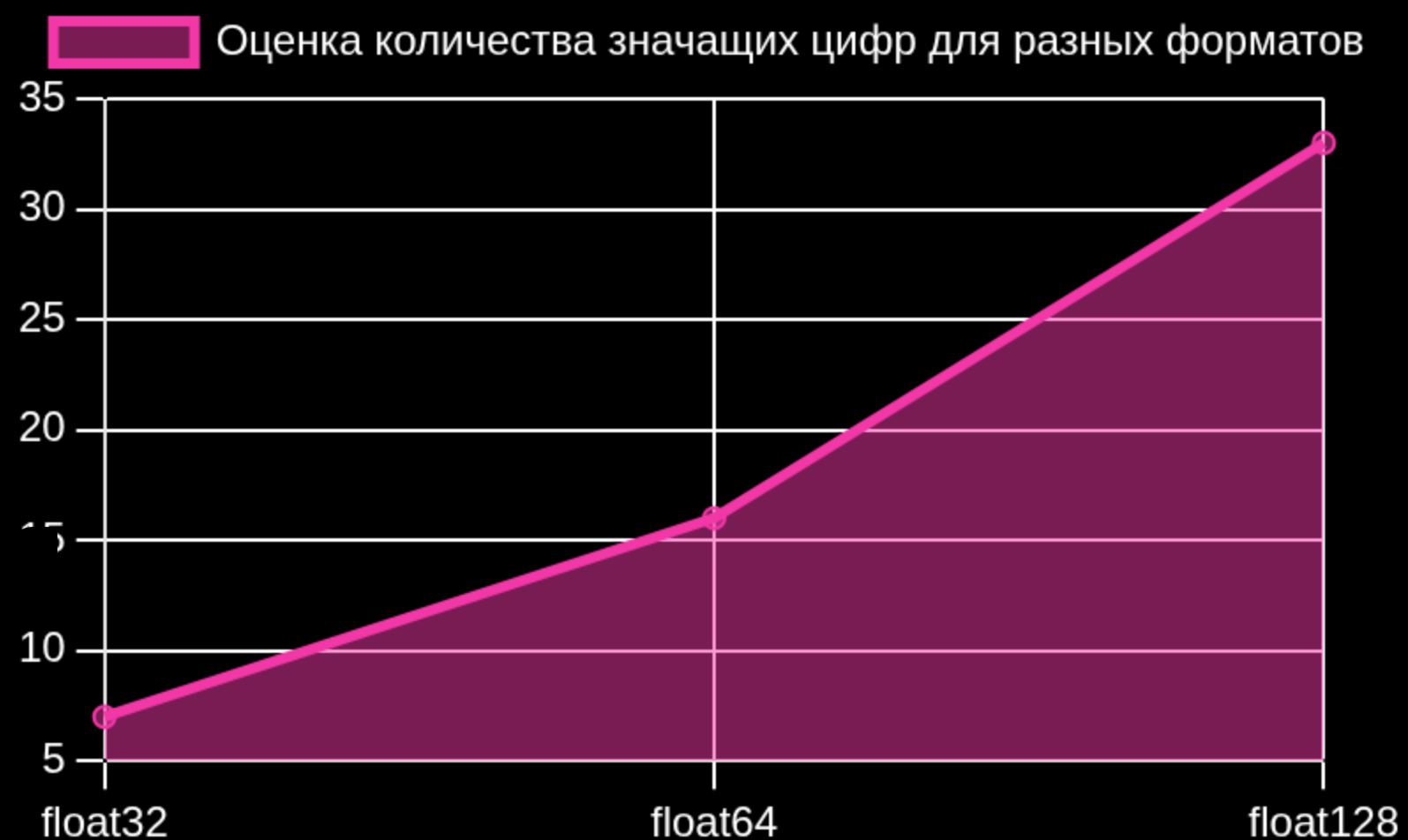


Порядок определяет степень двойки и задаётся в смещённом виде. Такое хранение упрощает сравнение чисел и их обработку в аппаратуре и программах.



Мантисса содержит значащие биты и несёт основную долю точности. Вместе три части дают возможность представлять очень большие и очень малые числа компактно.

# Сколько значащих цифр дают разные форматы



- Переход к более широкому формату увеличивает число разрядов мантиссы, поэтому «мелкие отличия» реже теряются при округлении.
- График показывает прямую связь точности с размером мантиссы: больше битов — надёжнее вычисления, но выше стоимость хранения.

---

IEEE 754 (оценочные значения значащих цифр)

# Нормализация и скрытый старший бит



- Нормализация приводит число к виду, где перед двоичной точкой стоит ровно одна значащая единица. Для ненулевых нормализованных значений эта единица известна заранее.

- Поскольку ведущая единица не хранится явно, она называется скрытым битом. Экономия одного бита позволяет выделить больше разрядов под мантиссу, увеличивая точность.

- Такой подход сохраняет тот же диапазон степеней, но делает представление числа «плотнее». В результате возрастает количество информации о дробной части при том же размере формата.

# Диапазон чисел в популярных форматах

- Сравнение приблизительных диапазонов и объёма памяти для одного значения в float32 и float64.
- Более широкий диапазон достигается за счёт большего числа битов порядка, что требует большего объёма памяти.

---

IEEE 754

| Формат  | Размер  | Диапазон (прибл.)                                          |
|---------|---------|------------------------------------------------------------|
| float32 | 4 байта | $\approx 1,18 \times 10^{-38} \dots 3,40 \times 10^{38}$   |
| float64 | 8 байт  | $\approx 2,23 \times 10^{-308} \dots 1,80 \times 10^{308}$ |

# Почему возникают ошибки округления

01.

Ошибки округления появляются уже на этапе перевода десятичной дроби в двоичную форму, если число не имеет конечной записи в системе счисления с основанием 2.

02.

Даже если мы выбираем ближайшее представимое значение, оно отличается от исходного. Дополнительная погрешность возникает из-за ограниченной длины мантиссы, которая не хранит бесконечные цифры.

03.

Когда вычисления повторяются многократно — в циклах или при суммировании большого массива — небольшие расхождения не исчезают. Они постепенно накапливаются и могут изменить итоговую величину.

04.

В информатике это важно учитывать при проверках, построении алгоритмов и сравнении результатов: «почти равно» часто означает «по определённому допуску», а не «строго равно».

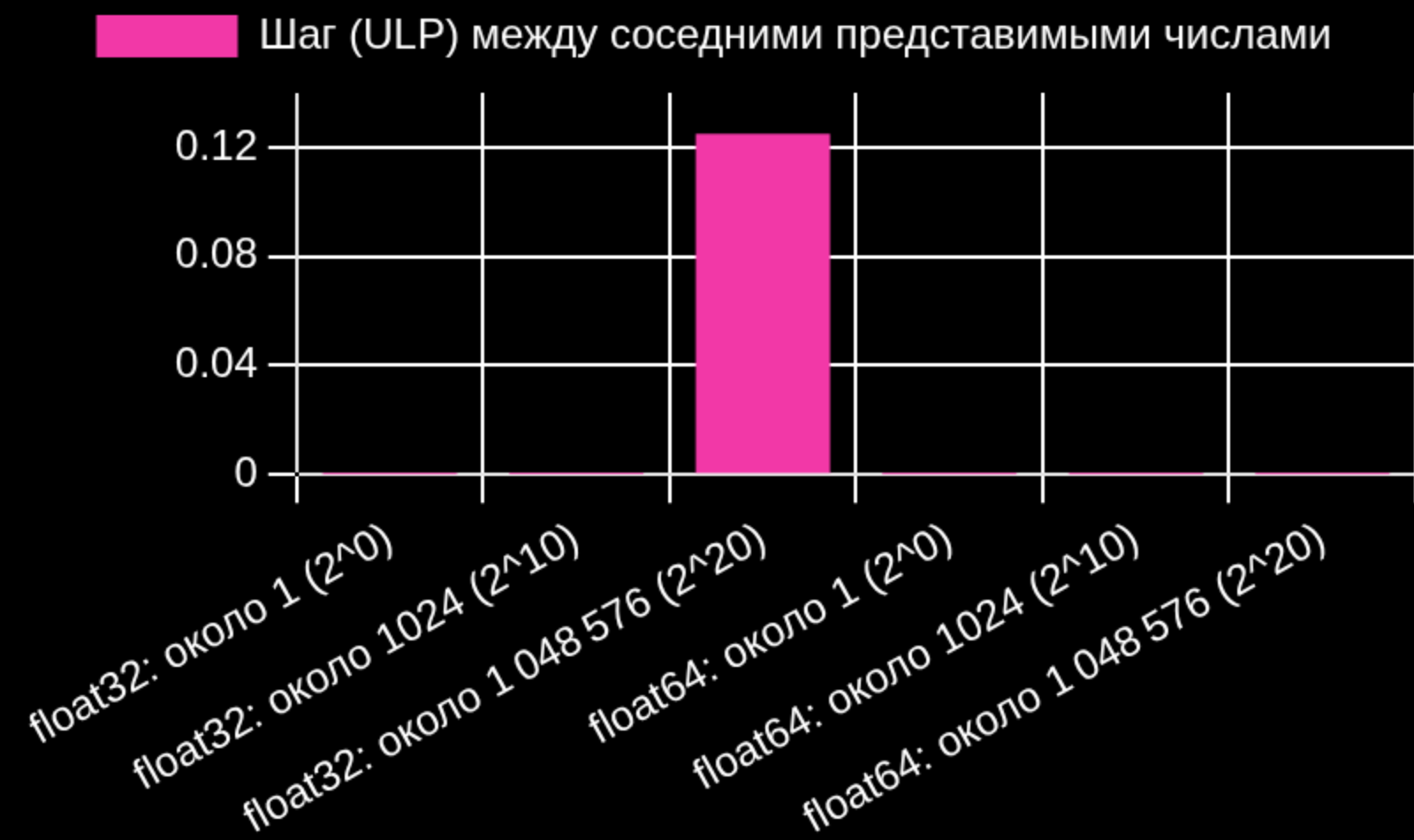
# Пример $0,1 + 0,2 \neq 0,3$ в машинной арифметике

В памяти числа 0,1 и 0,2 представлены как ближайшие двоичные приближения, потому что в основании 2 эти значения имеют бесконечные периодические дроби. Поэтому они уже «чуть другие», чем в учебнике.

Их сумма в float64 может дать 0,30000000000000004. Этот пример показывает, что строгие сравнения вещественных чисел с помощью равенства бывают ненадёжны; полезно различать точное ожидание, машинное приближение и результат расчёта.



# Шаг между соседними числами зависит от их величины



На разных масштабах одна и та же

- «единица точности» становится другой: шаг между представимыми значениями растёт с увеличением экспоненты числа.

График объясняет, почему при очень больших значениях мелкие изменения могут не

- отражаться в результате: вычисление просто «перескакивает» через доступные представимые числа.

---

Справочные оценки по IEEE 754: шаг между соседними нормализованными значениями (ULP) равен  $2^{(e - \text{mantissa\_bits})}$  для соответствующей экспоненты.

# Особые значения IEEE 754



## +0 и -0

В стандарте различаются +0 и -0. В большинстве привычных сравнений они ведут себя одинаково, но в отдельных операциях и при анализе знака результат может отличаться.



## Бесконечности

Если вычисление выходит за допустимый диапазон, результат может стать  $+\infty$  или  $-\infty$ . Это сигнал «переполнения» и дальнейшие операции уже распространяют особое значение по правилам IEEE 754.



## NaN (не число)

NaN используют для обозначения неопределённости. Например, при вычислении  $0/0$  или при попытке извлечь квадратный корень из отрицательного числа в вещественной арифметике.



## Поведение в программах

Наличие специальных значений означает, что ветвления и проверки должны учитывать случаи, когда число не похоже на обычное. Иначе алгоритм может дать неожиданные результаты без явной ошибки.

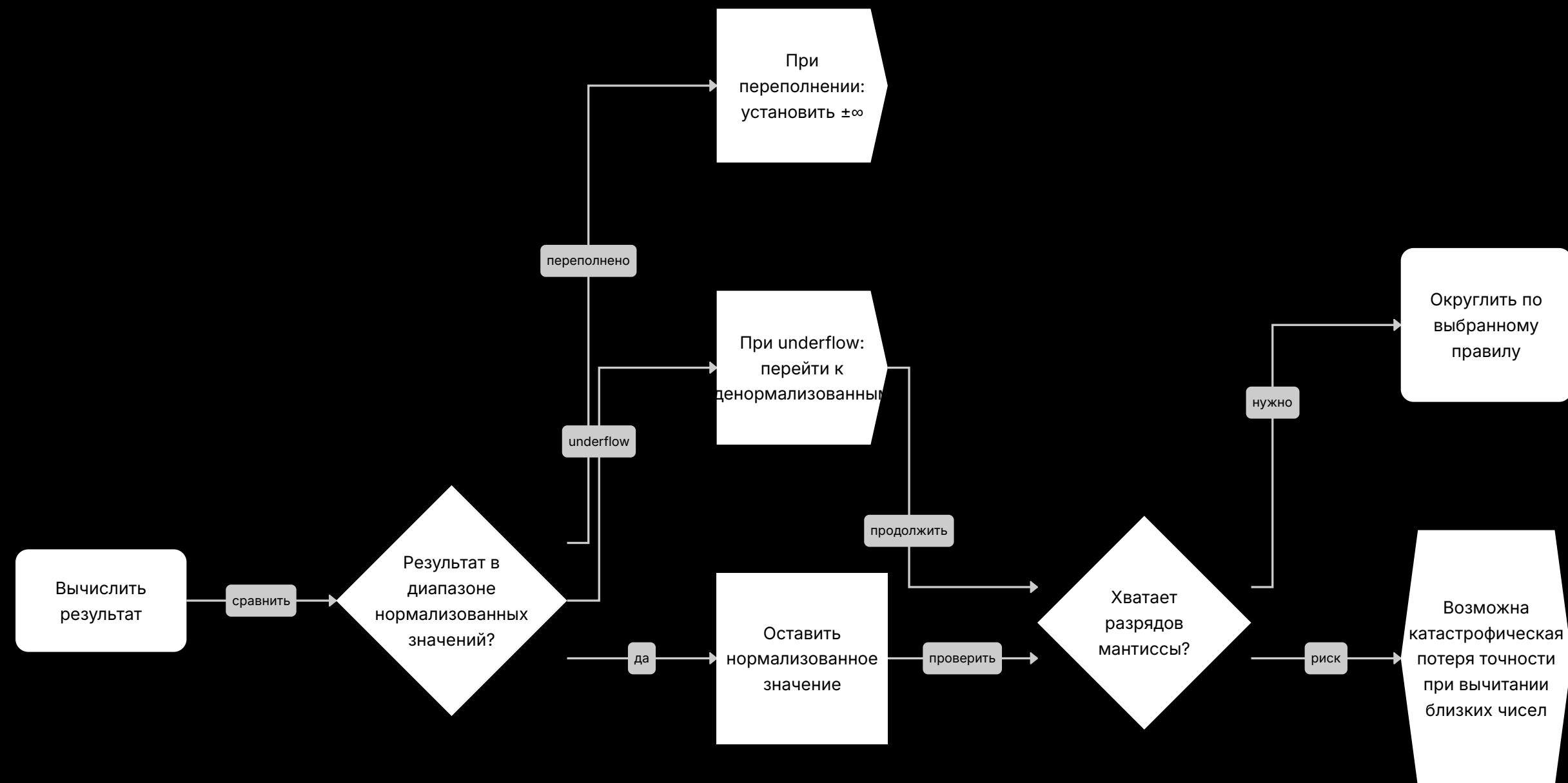


## Проверки вместо «равно»

При работе с вещественными числами надёжнее использовать допуски и специальные проверки (например, на NaN или бесконечности), чем полагаться только на строгие условия равенства.

# Что происходит при переполнении и потере точности

Схема отражает типичные этапы обработки в IEEE 754: проверка диапазона, округление мантиссы и переход к специальным состояниям при ошибках представления.



# Сравнение точности вычислений в простых примерах

| Операция                     | Ожидание        | float32                     | float64                              |
|------------------------------|-----------------|-----------------------------|--------------------------------------|
| 0,1 + 0,2                    | 0,3             | 0,3000000119                | 0,30000000000000000                  |
| Проверка на точное равенство | да              | не совпадает                | не совпадает                         |
| Вычитание близких чисел      | точное значение | относительная ошибка растёт | относительная ошибка растёт (меньше) |

Таблица показывает, что машинное представление и округление приводят к отличию фактического результата от «математически ожидаемого». float64 обычно точнее, но погрешность не исчезает полностью.

Число может быть «почти равно», но не равно строго. float64 уменьшает погрешность за счёт большей мантиссы, однако сравнения всё равно требуют допусков.

Учебные демонстрации стандартных примеров IEEE 754 (различия из-за двоичных приближений и округления).

# Как уменьшать ошибки при работе с дробями

01.

Для денежных расчётов лучше использовать целые числа в минимальных единицах — копейках, центах и т. п. Тогда вы избегаете двоичной арифметики и получаете точный результат на уровне единиц учёта.

02.

В численных методах применяют сравнение с допуском: проверяют, что разница меньше заранее выбранной величины, а не равенство «в точности». Это снижает риск ложных срабатываний.

03.

Устойчивость повышают аккуратным порядком действий и группировкой слагаемых: сначала суммируют близкие по величине числа, а также используют специальные приёмы суммирования, уменьшающие накопление округлительных ошибок.

# Где вещественные числа особенно важны



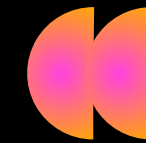
Форматы с плавающей точкой лежат в основе компьютерной графики и игр, где нужно хранить координаты, углы и длины. Небольшие погрешности часто незаметны визуально, но важна скорость обработки.



В инженерных расчётах и научном моделировании вещественные числа помогают описывать реальные процессы: температуры, давления, траектории и измерения. Там цена ошибки выше, поэтому чаще выбирают float64 или повышают точность алгоритма.



В машинном обучении и обработке сигналов данные проходят через большое число вычислений, поэтому ошибки округления могут влиять на сходимость и качество моделей. Выбор формата влияет и на требования к памяти, и на время обучения.



При проектировании систем приходится выбирать между компактностью и надёжностью: в массовых вычислениях удобнее float32, а там, где результат критичен, разумнее использовать float64 или специальные десятичные подходы.

# Память, скорость и точность: неизбежный компромисс

Увеличение числа битов повышает точность, потому что больше разрядов мантиссы позволяет точнее описывать двоичные приближения исходного числа. Но за это приходится платить ростом объёма памяти и вычислительных ресурсов.

При обработке больших массивов данных это напрямую влияет на скорость и на то, сколько значений можно хранить одновременно. Поэтому float32 обычно выбирают для массовых операций, а float64 — когда цена ошибки выше.



# Главный вывод: точность определяется тем, как число «помещается» в формат



В памяти компьютера вещественные числа хранятся по IEEE 754 в виде двоичной нормализованной записи. Ограниченная мантисса, округление, переполнение и специальные значения формируют точность и поведение программ, поэтому выбор float32 или float64 влияет на корректность вычислений.