Яндекс

C++ Russia

Что-то у меня тормозит: заглядываем внутрь C++ контейнеров

Илья Шишков, старший разработчик

Кто я?

- Программирую на C++ с 2006 года
- Старший разработчик Яндекса
- 10 лет в Яндексе, почти все в роли разработчика

Кто я?

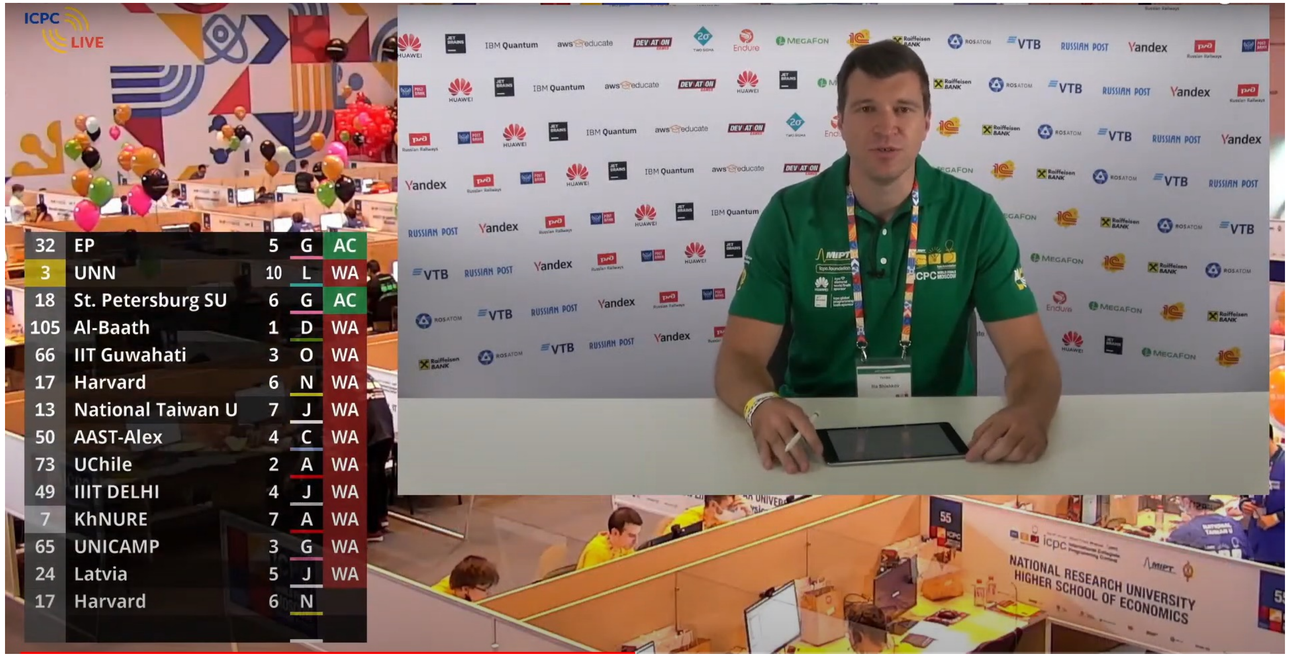
Пояса по C++

https://cppcourse.ru

Алгоритмический фундамент программиста

https://algo-base.ru

Кто я?



Участник финалов ACM ICPC

Зачем слушать мой доклад?

Чтобы узнать границы применимости default’ов в C++

- Зачем? — Чтобы создавать более качественные продукты и делать это быстрее
- Зачем? — Чтобы получать большее удовлетворение от работы и больше денег
- Зачем? — Чтобы быть счастливее

Дефолты в C++

- vector — для хранения последовательности элементов
- unordered_map — для словаря

Они используются, потому что являются наиболее эффективными для своего применения

«Наиболее эффективными» — это как?

vector

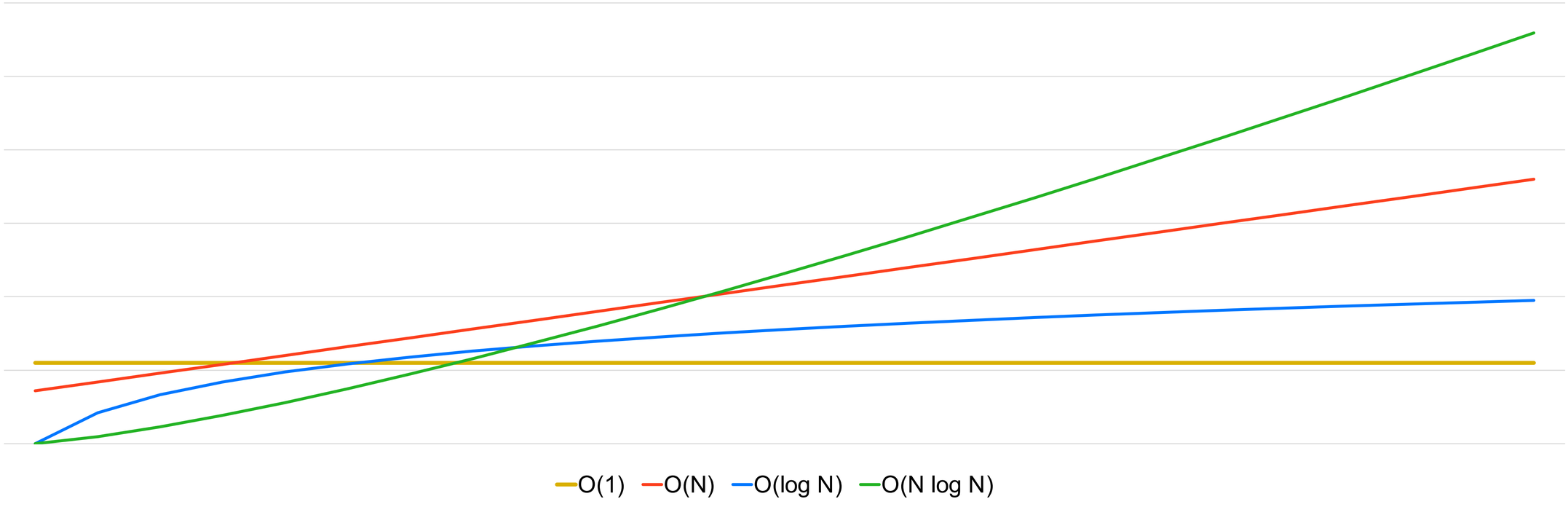
- push_back — O(1)
- Итерация — O(N), cache friendly
- Обращение по индексу — O(1)

unordered_map

- вставка — O(1)
- поиск по значению — O(1)
- удаление — O(1)
- Итерация — O(N)

Асимптотика времени работы

Асимптотика времени работы — это описание того, как изменяется время выполнения алгоритма с увеличением размера входных данных.



$O(1)$ — время работы
не зависит от размера входных данных

std::deque<T,Allocator>::push_back

```
void push_back( const T& value );    (1)
void push_back( T&& value );         (2)    (since C++11)
```

Appends the given element value to the end of the container.

1) The new element is initialized as a copy of value.
2) value is moved into the new element.

All iterators, including the end() iterator, are invalidated. No references are invalidated.

Parameters

value - the value of the element to append

Type requirements

- T must meet the requirements of *CopyInsertable* in order to use overload (1).
- T must meet the requirements of *MoveInsertable* in order to use overload (2).

Return value

(none)

Complexity

Constant.

std::vector<T,Allocator>::push_back

`void push_back( const T& value );`	(1)	(until C++20)
`constexpr void push_back( const T& value );`		(since C++20)
`void push_back( T&& value );`	(2)	(since C++11) (until C++20)
`constexpr void push_back( T&& value );`		(since C++20)

Appends the given element value to the end of the container.

1) The new element is initialized as a copy of value.

2) value is moved into the new element.

If the new `size()` is greater than `capacity()` then all iterators and references (including the `end()` iterator) are invalidated. Otherwise only the `end()` iterator is invalidated.

Parameters

value - the value of the element to append

Type requirements

- T must meet the requirements of *CopyInsertable* in order to use overload (1).
- T must meet the requirements of *MoveInsertable* in order to use overload (2).

Return value

(none)

Complexity

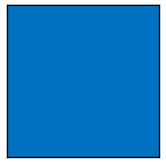
Amortized constant.

Ароматизированная константа

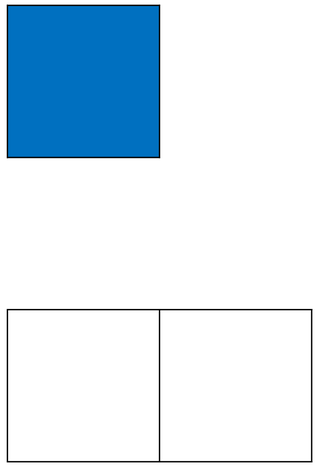
Амортизационная константа

Амортизированная константа

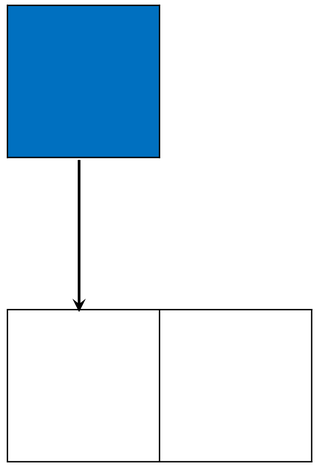
Добавление в конец вектора



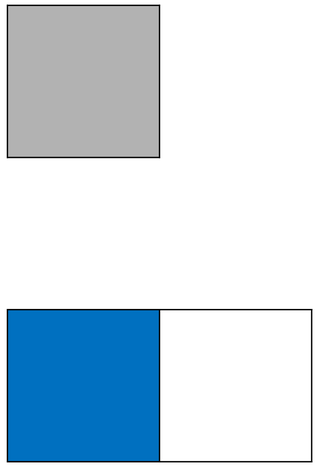
Добавление в конец вектора



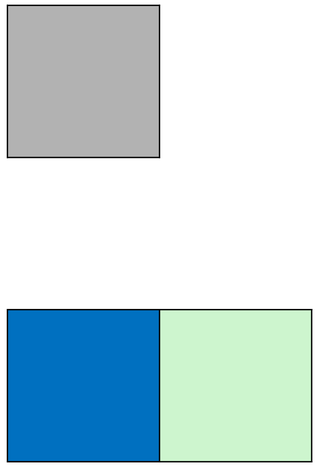
Добавление в конец вектора



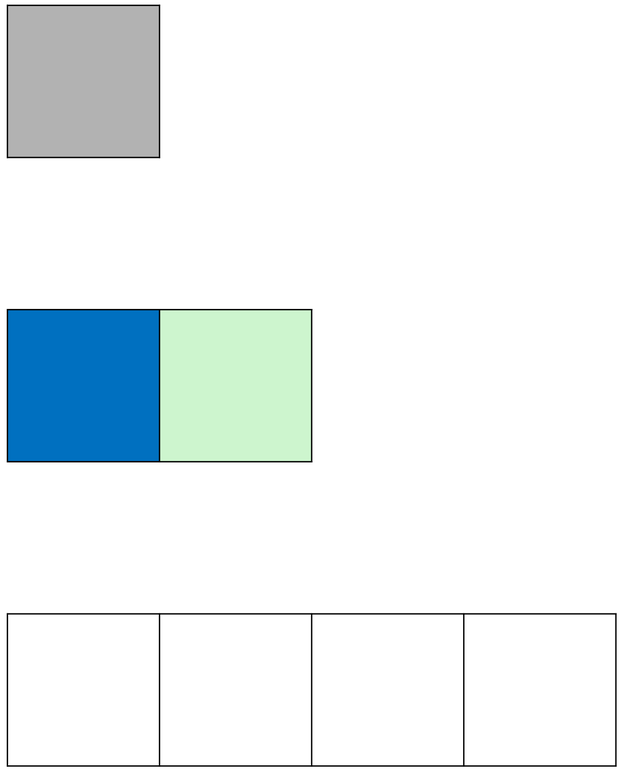
Добавление в конец вектора



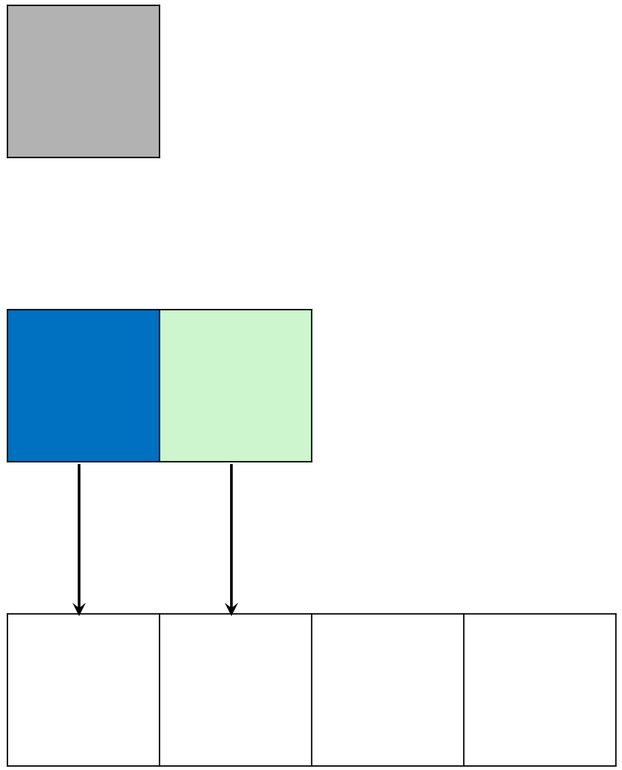
Добавление в конец вектора



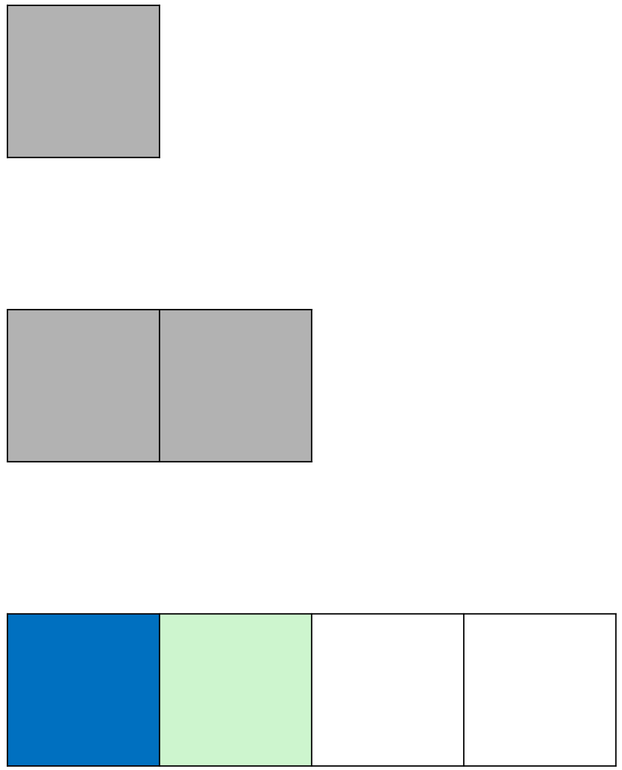
Добавление в конец вектора



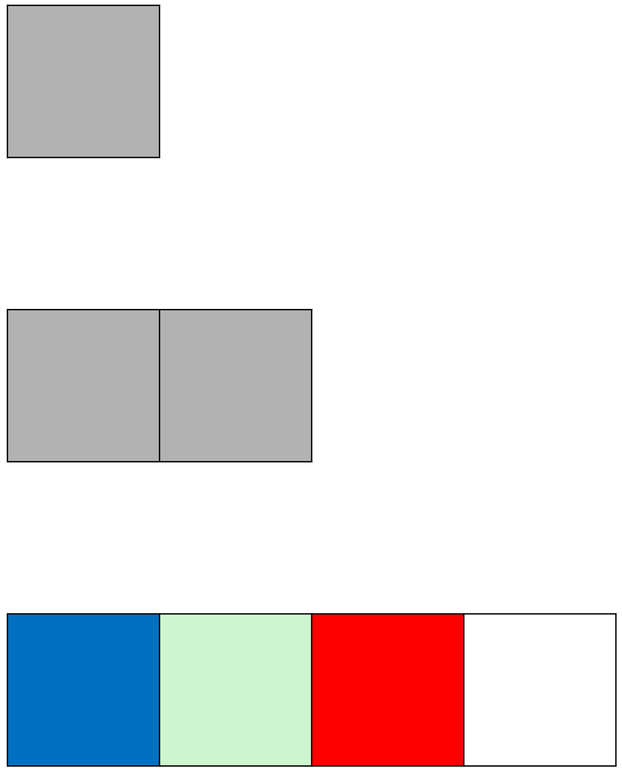
Добавление в конец вектора



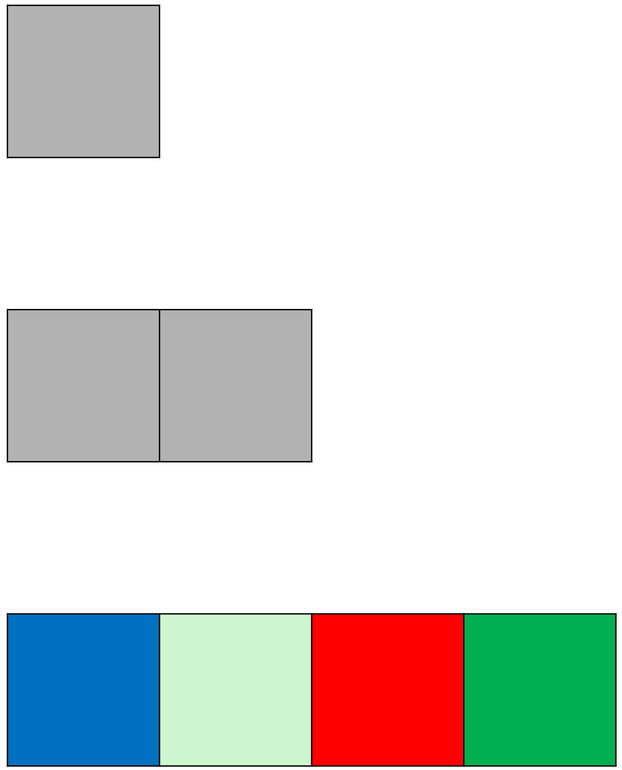
Добавление в конец вектора



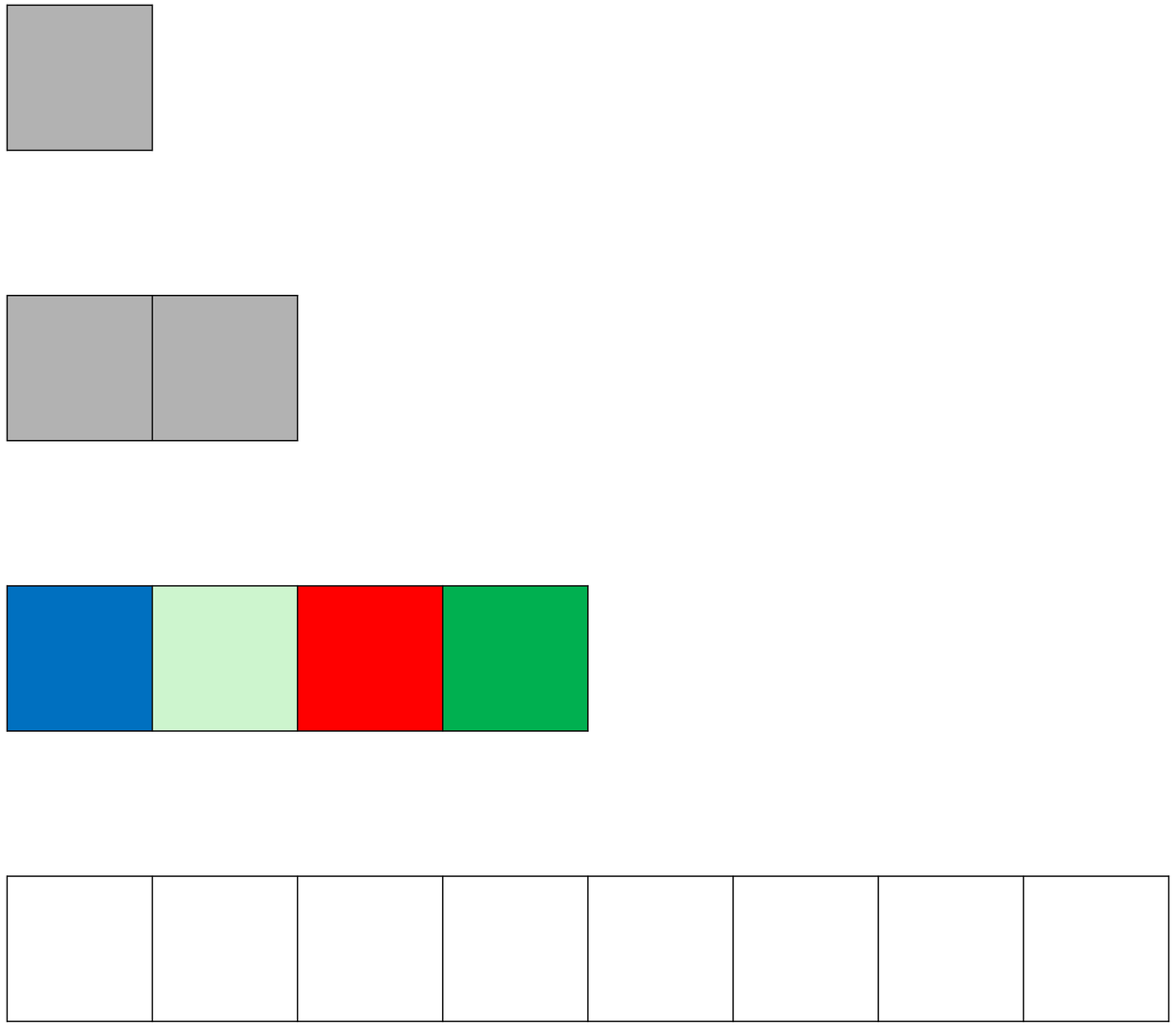
Добавление в конец вектора



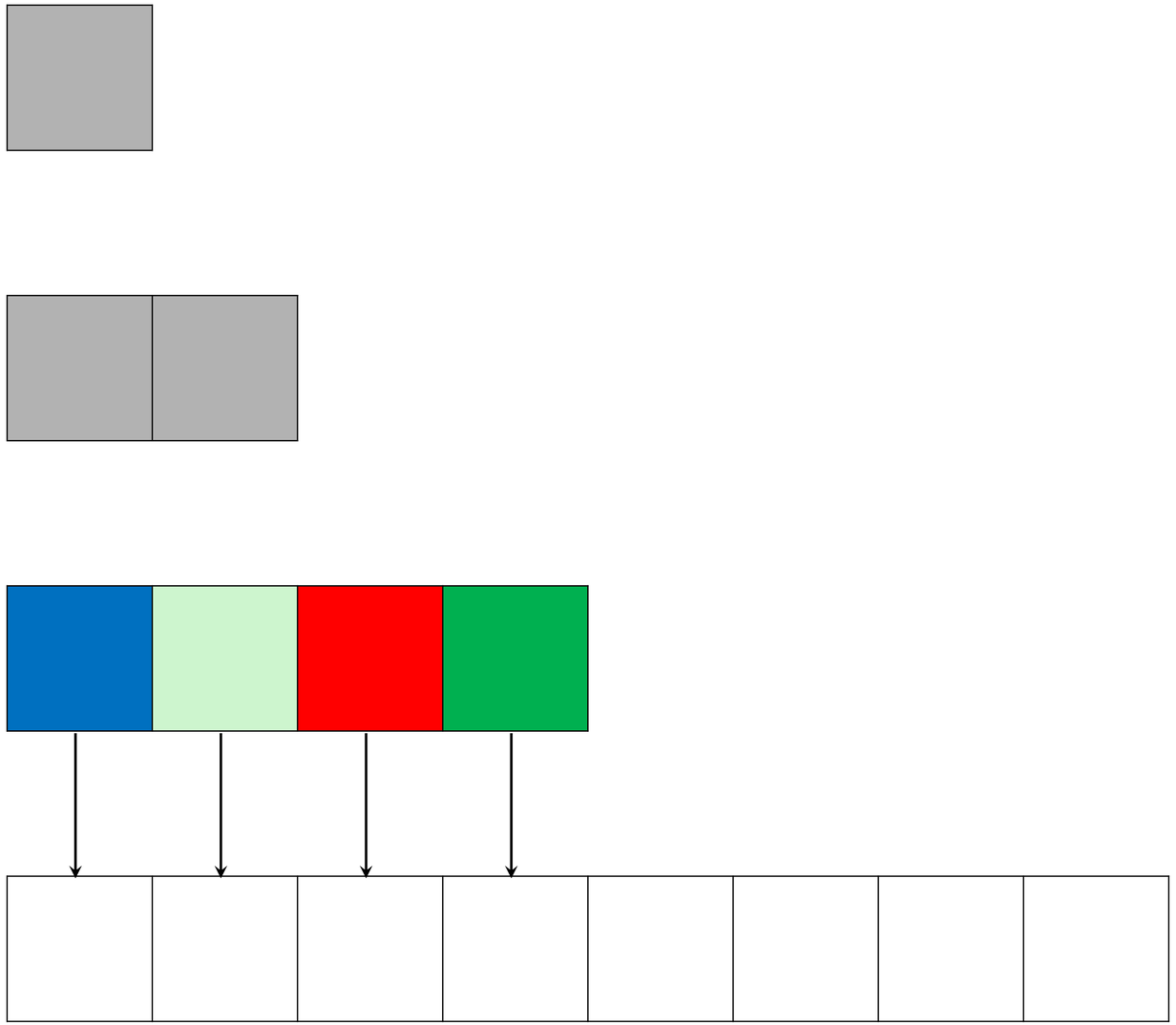
Добавление в конец вектора



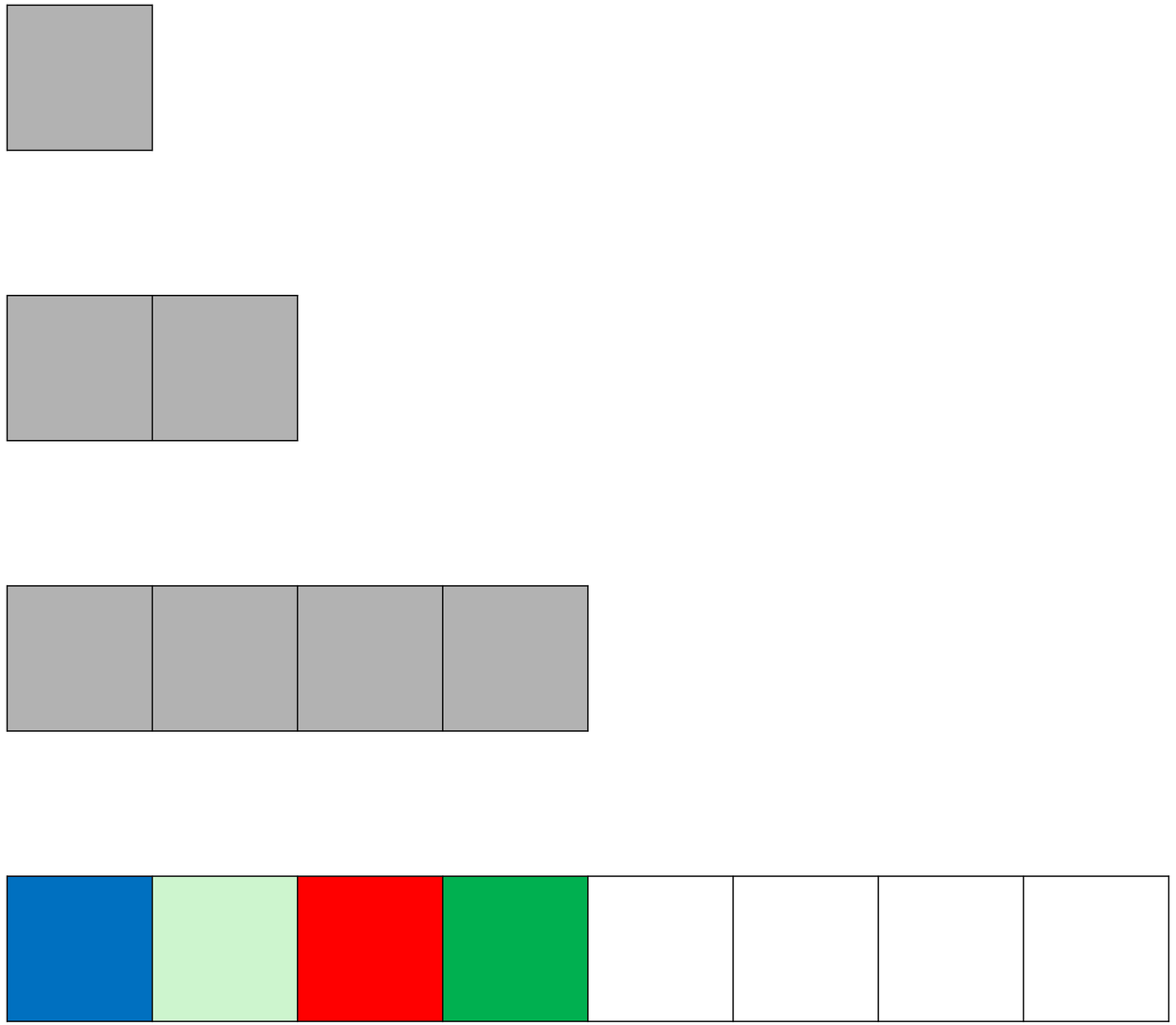
Добавление в конец вектора



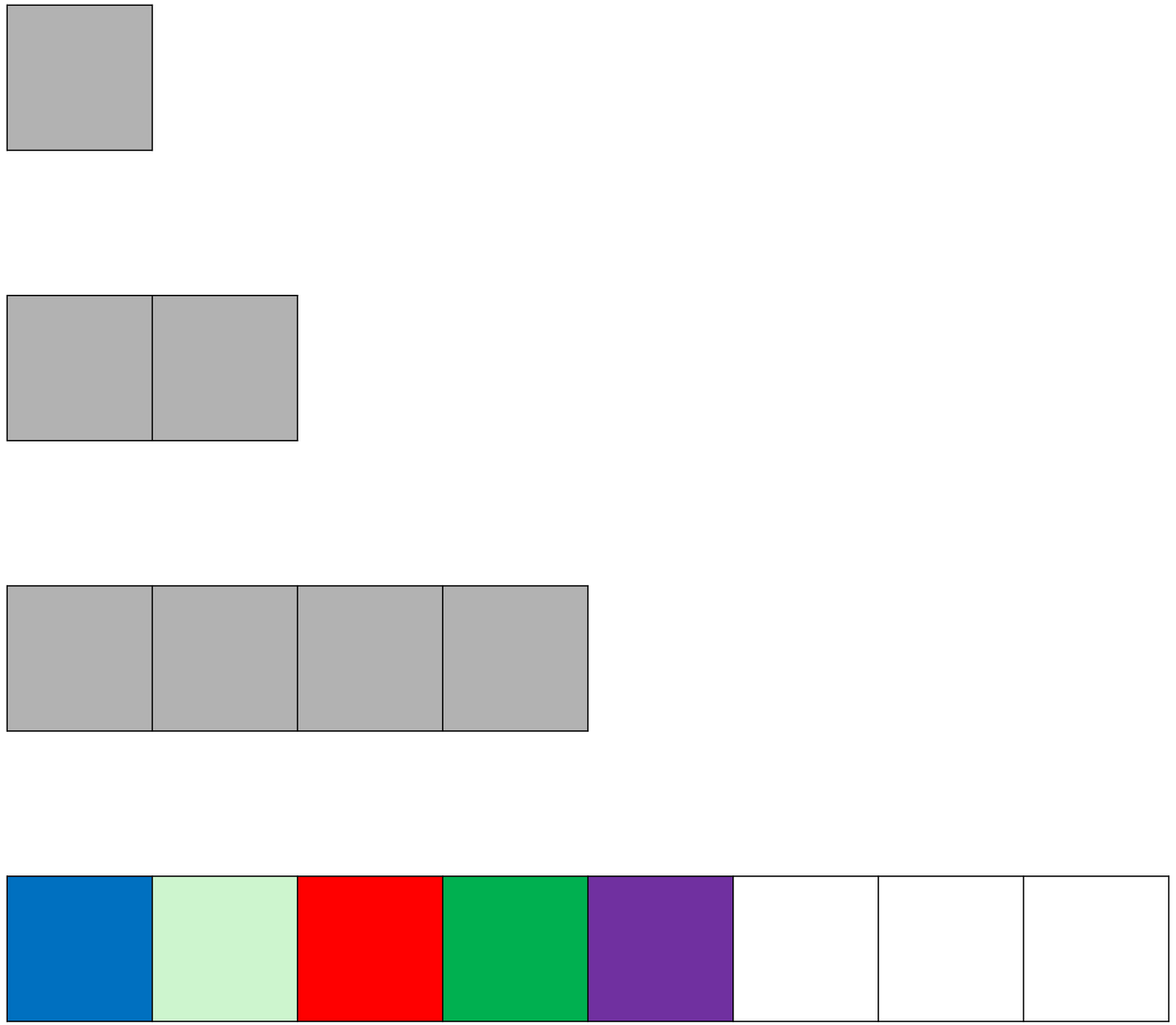
Добавление в конец вектора



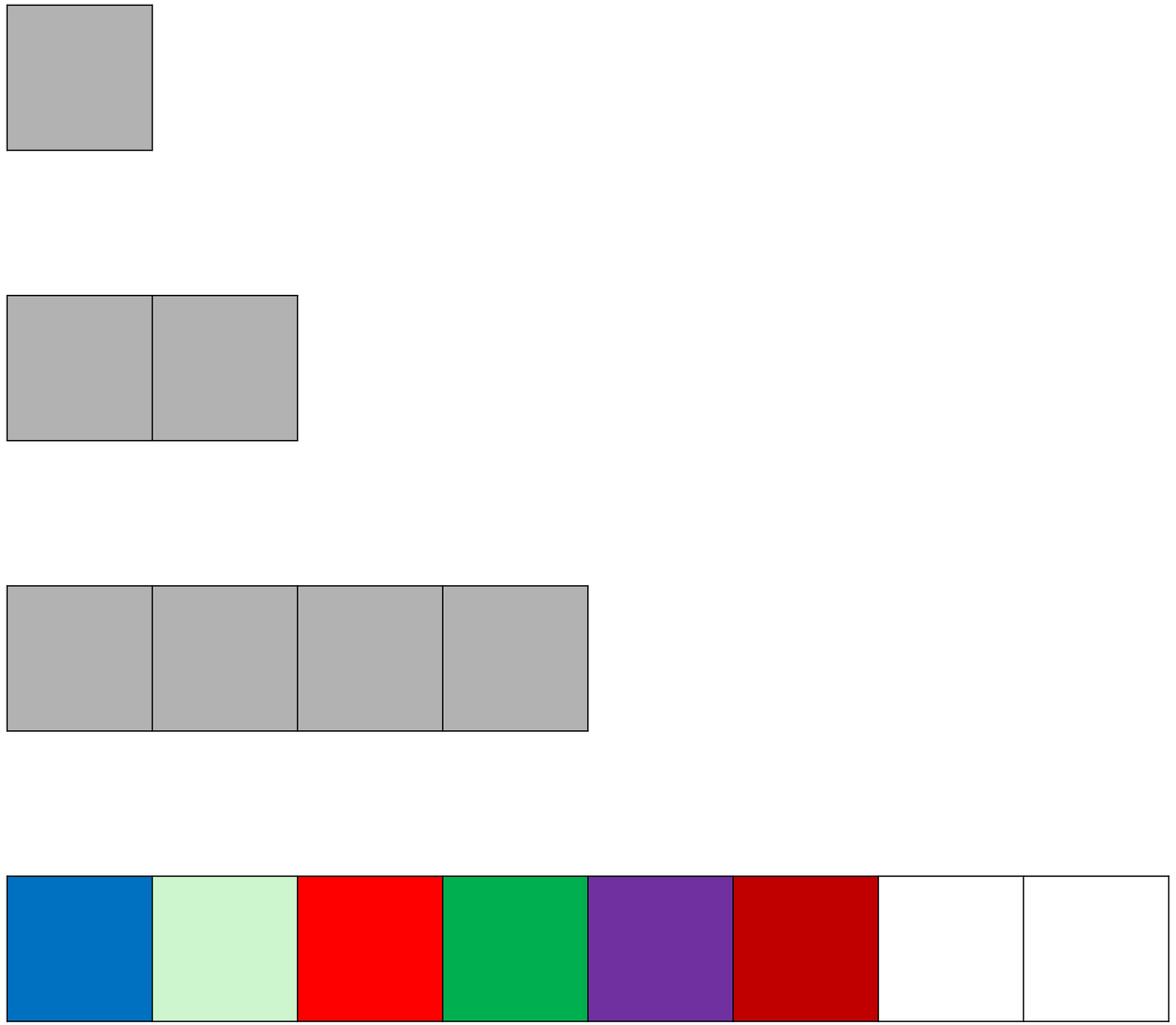
Добавление в конец вектора



Добавление в конец вектора



Добавление в конец вектора



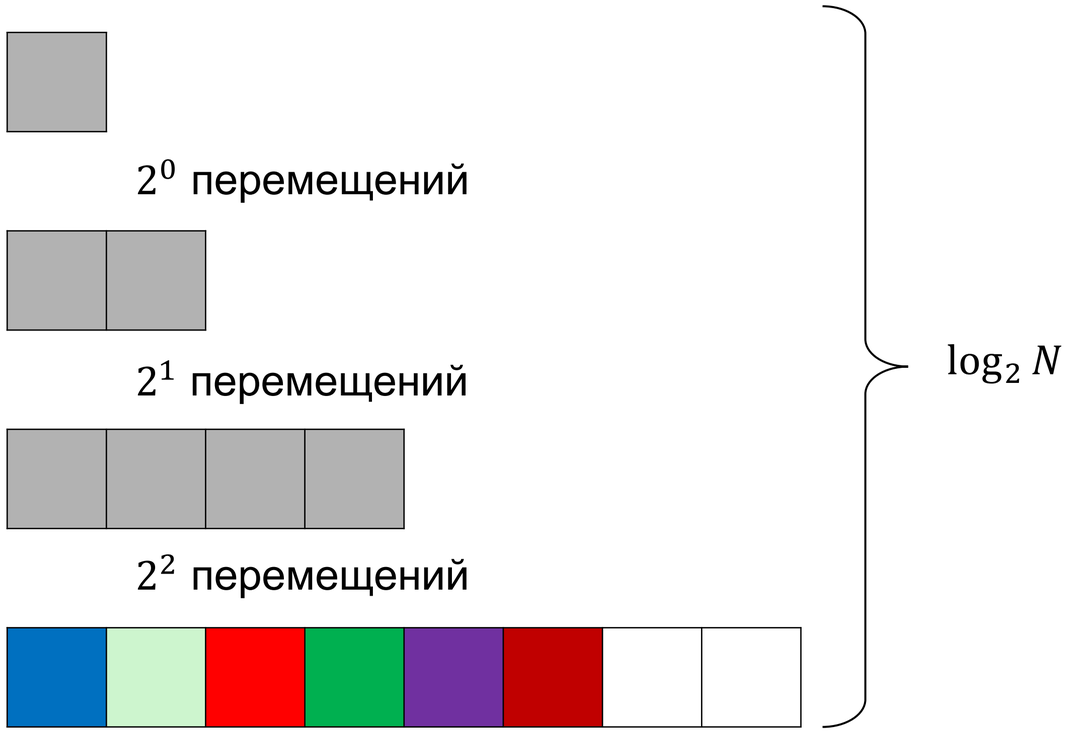
Время работы vector::push_back

- Когда в буфере есть свободное место — за $O(1)$
- Когда свободного места нет — за $O(N)$ за счёт перемещения элементов

Сколько будет таких перемещений после выполнения N вставок?

Время работы vector::push_back

- Так как буфер растёт в два раза, то после N вставок будет совершено $\log_2 N$ реаллокаций
- В результате реаллокации с номером i совершается $O(2^i)$ перемещений
- Всего перемещений получается
$$1 + 2 + 4 + \cdots + 2^{\log_2 N}$$
- Это сумма первых $\log_2 N$ членов геометрической прогрессии с первым членом 1 и знаменателем 2:
$$\frac{2^{\log_2 N} - 1}{2 - 1} = N - 1$$



Добавление в конец вектора

- В результате выполнения N push_back'ов будет выполнено $N - 1$ перемещение
- То есть N операций вставки работают за $O(N)$
- А значит, одна операция вставки в конец вектора работает за **амортизированное** $O(1)$

Амортизированное $O(1)$ = N операций всегда выполняются за $O(N)$

Итоги

Не все $O(1)$ одинаковы

- $O(1)$ может быть амортизированным
- Амортизированное $O(1)$ означает, что N операций всегда выполняются за $O(N)$

Особенности применения std::vector

- Отдельные вызовы push_back могут работать за $O(N)$
- push_back **может** приводить к инвалидации ссылок, итераторов и указателей

Устройство unordered_set

Асимптотики времени работы операций

unordered_set

- вставка — O(1)
- поиск по значению — O(1)
- удаление — O(1)

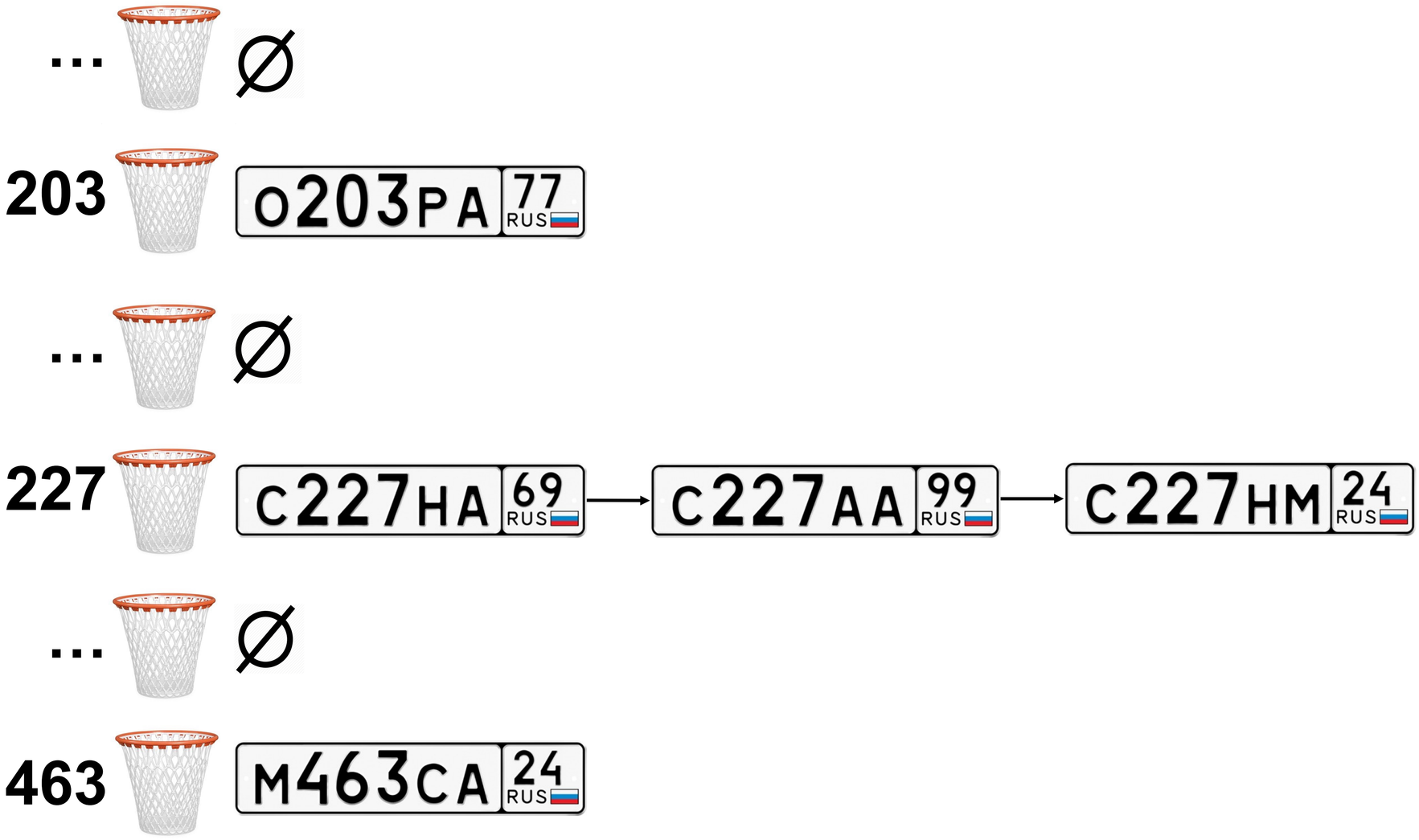
cppreference.com:

- вставка — Average case: O(1), worst case: O(size())
- поиск по значению — Constant on average, worst case: O(size()).
- удаление — Average case: O(1), worst case: O(size())

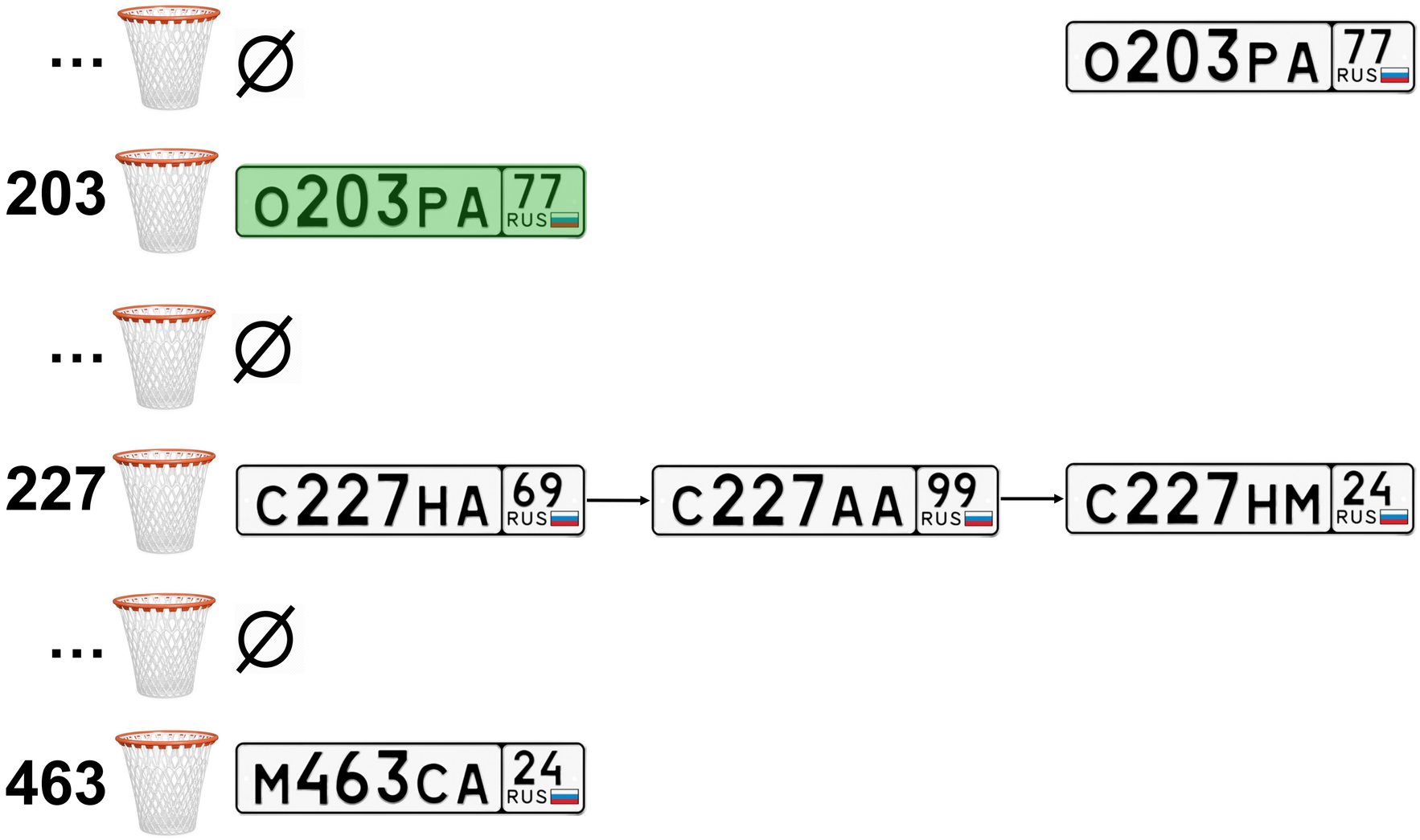
Устройство unordered_set

- Хеш-таблица
- Состоит из набора бакетов
- Цепочки для разрешения коллизий

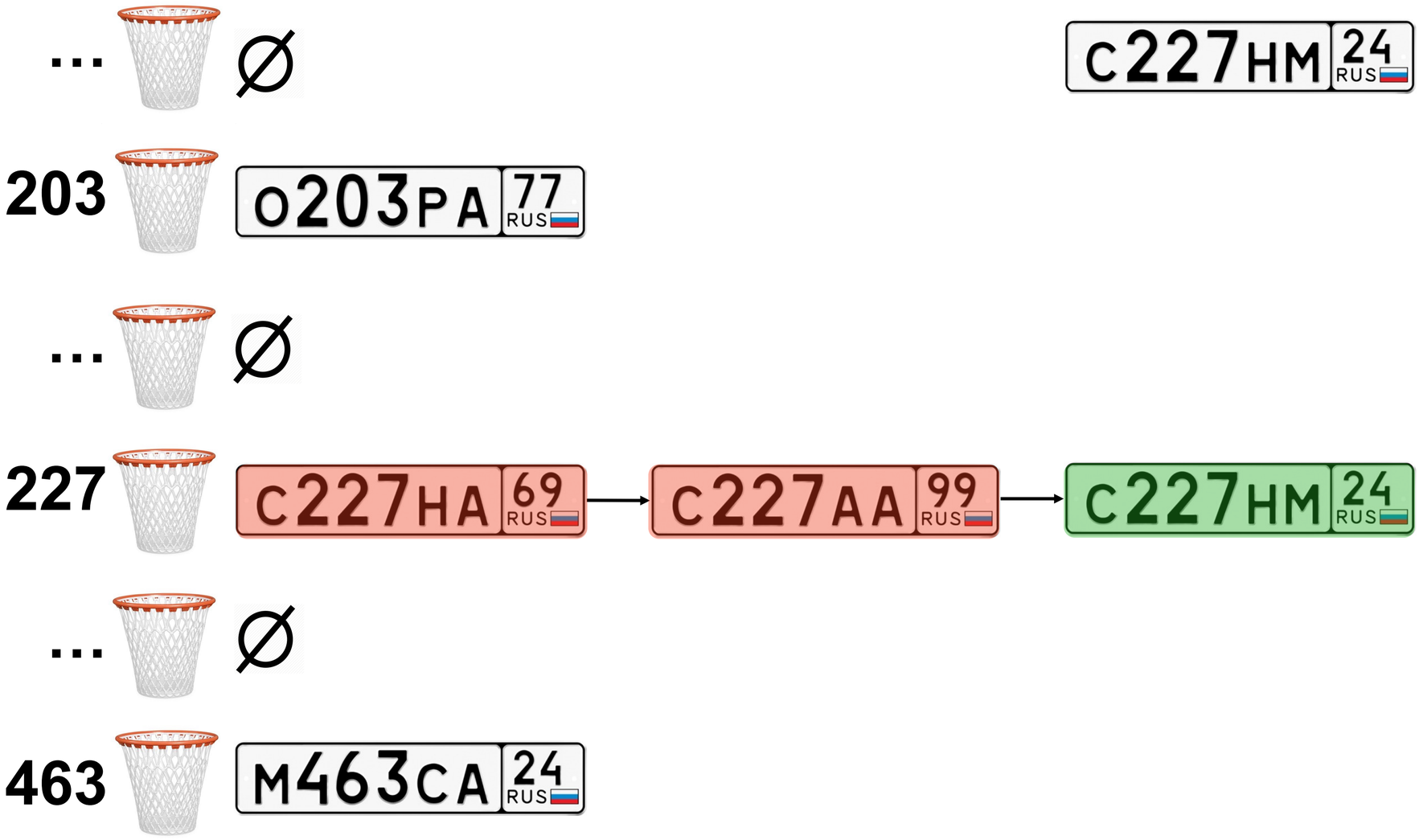
Устройство unordered_set



Поиск в unordered_set

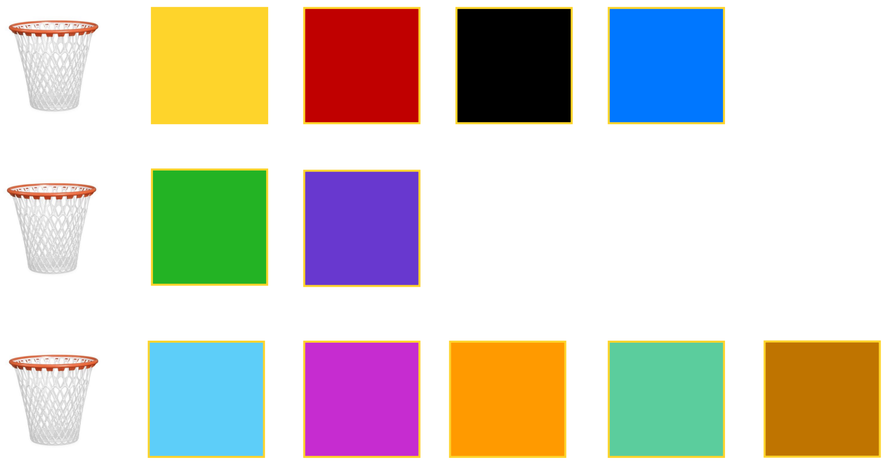


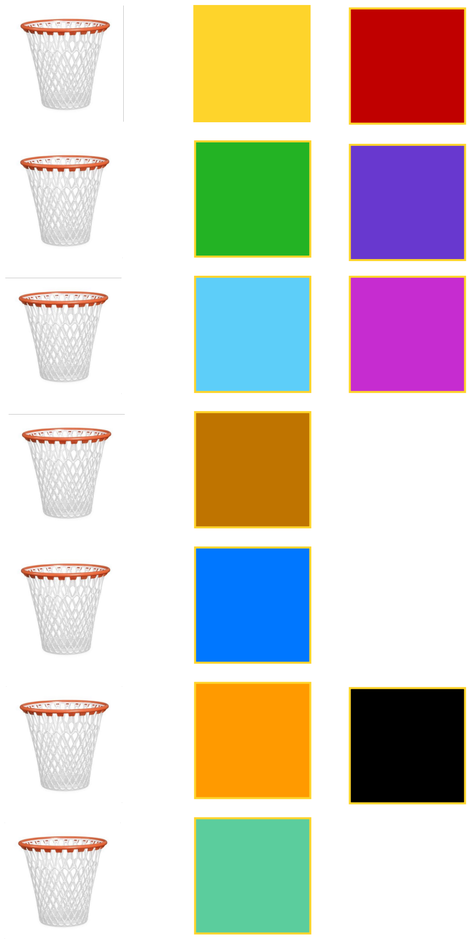
Поиск в unordered_set



Время работы поиска в unordered_set зависит от размера таблицы?

Рехеширование





Load factor

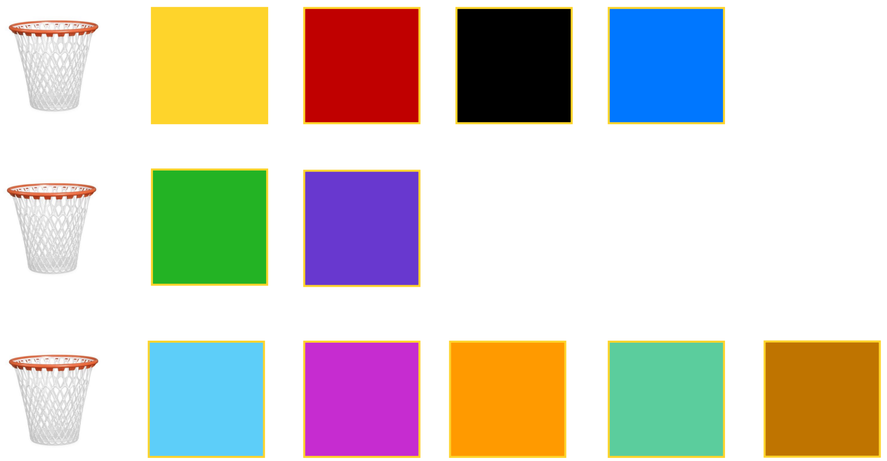
$load\ factor = {}^{N}/_{B}$, где

N — число элементов в хеш-таблице,

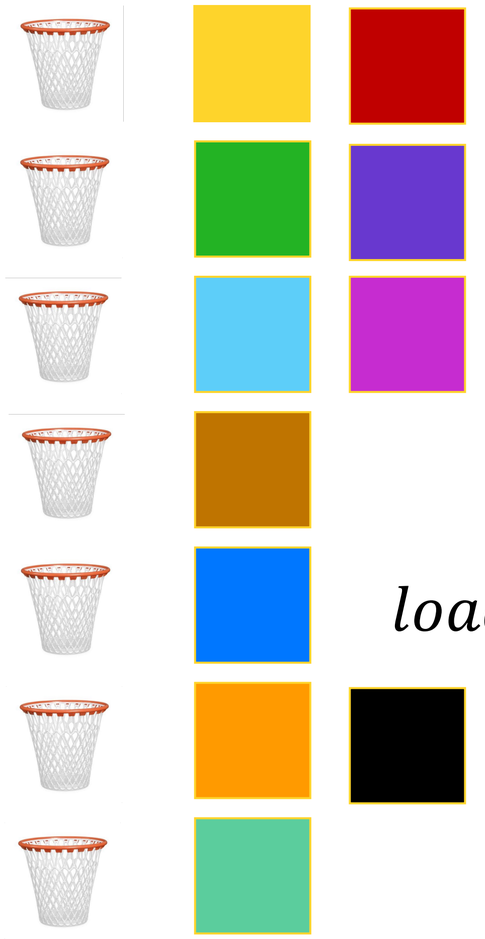
B — число бакетов в хеш-таблице

- Load factor задаёт среднюю длину цепочки в хеш-таблице
- Если load factor достигает высокого значения, мы выполняем рехеширование

Load factor



$$load\ factor = \frac{11}{3} \approx 3{,}67$$



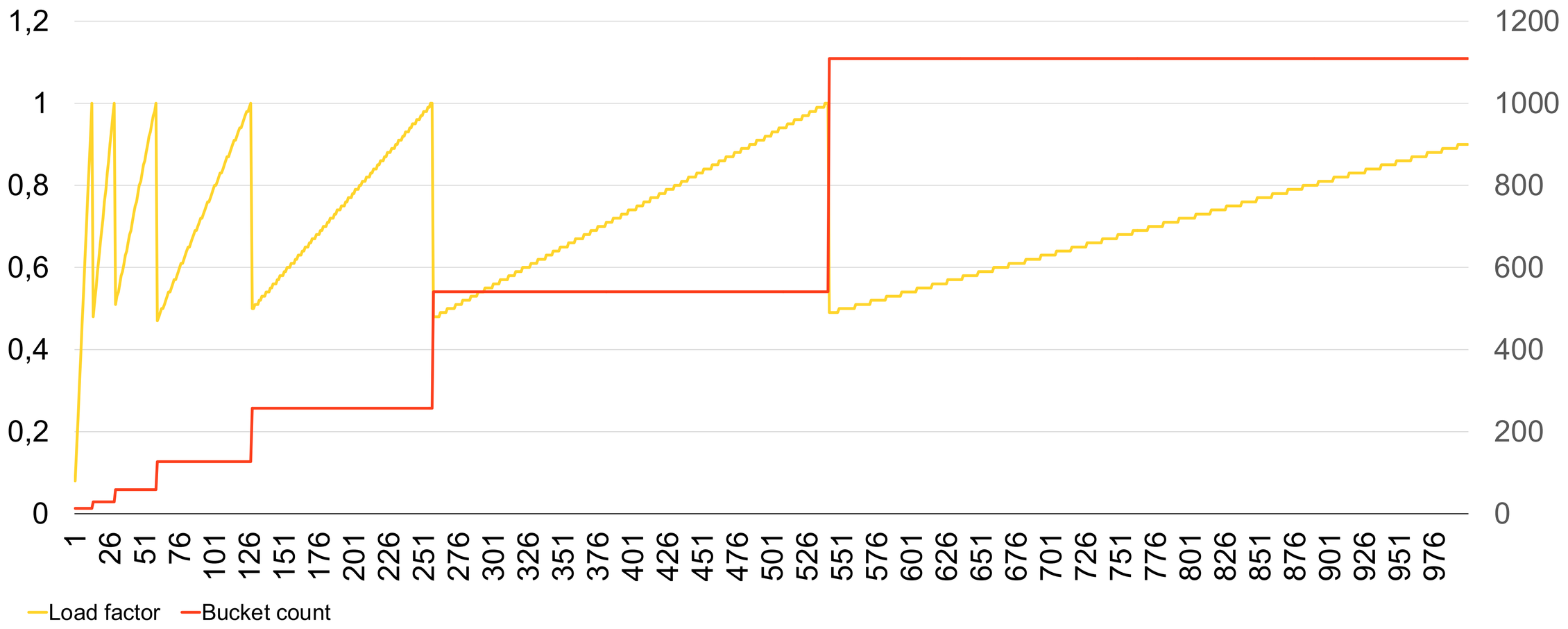
$$load\ factor = \frac{11}{7} \approx 1{,}57$$

Рехеширование и load factor

```
unordered_set<int> numbers;
vector<Stats> stats;
for (int i = 0; i < item_count; ++i) {
  numbers.insert(i);
  stats.push_back({.size = numbers.size(),
                   .bucket_count = numbers.bucket_count(),
                   .load_factor = numbers.load_factor()});
}

ofstream out(“bucket_count_load_factor.csv");
out << "size,bucket_count,load_factor\n";
out << fixed << setprecision(2);
for (const auto& stat_item : stats) {
  out << stat_item.size << ',' << stat_item.bucket_count << ',' << stat_item.load_factor << '\n';
}
```

Рехеширование и load factor



Длины цепочек в unordered_set

> **При равномерности хеш-функции и равновероятности всех ключей длины цепочек не превышают load_factor**

Интерфейс unordered_set

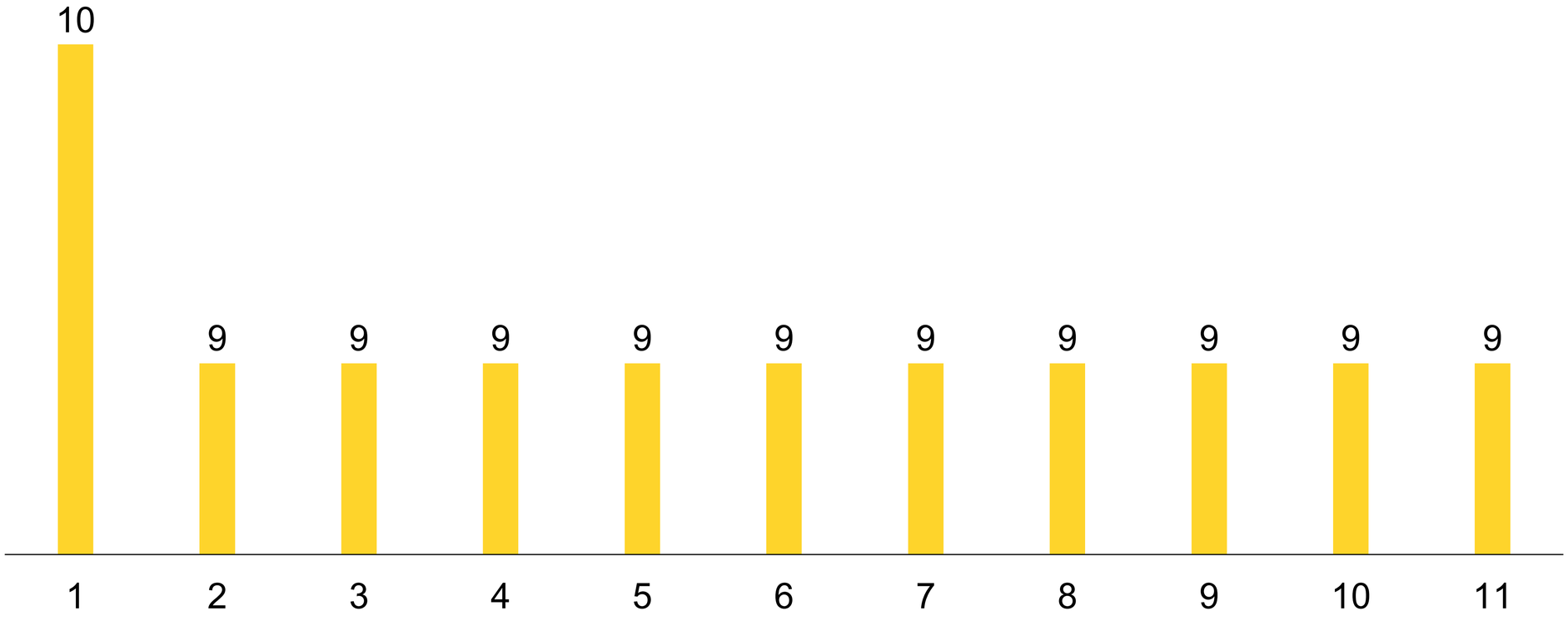
- float load_factor() – возвращает текущее значение load factor
- float max_load_factor() – возвращает значение load factor, при достижении которого выполняется рехеширование
- void max_load_factor(float) — позволяет задать пороговое значение load factor

Время работы unordered_set::find

- Средняя длина цепочки в unordered_set никогда не превосходит max_load_factor
- Поиск выполняется за длину цепочки, то есть в среднем за $O(max\_load\_factor)$
- max_load_factor не зависит от заполненности таблицы
- Значит, время поиска не зависит от числа элементов в таблице
- Значит, unordered_set::find работает за $O(1)$ в среднем

Качество хеш-функции и длины цепочек

bucket_size, N = 100, max_load_factor = 10



Равномерность хеш-функции важна

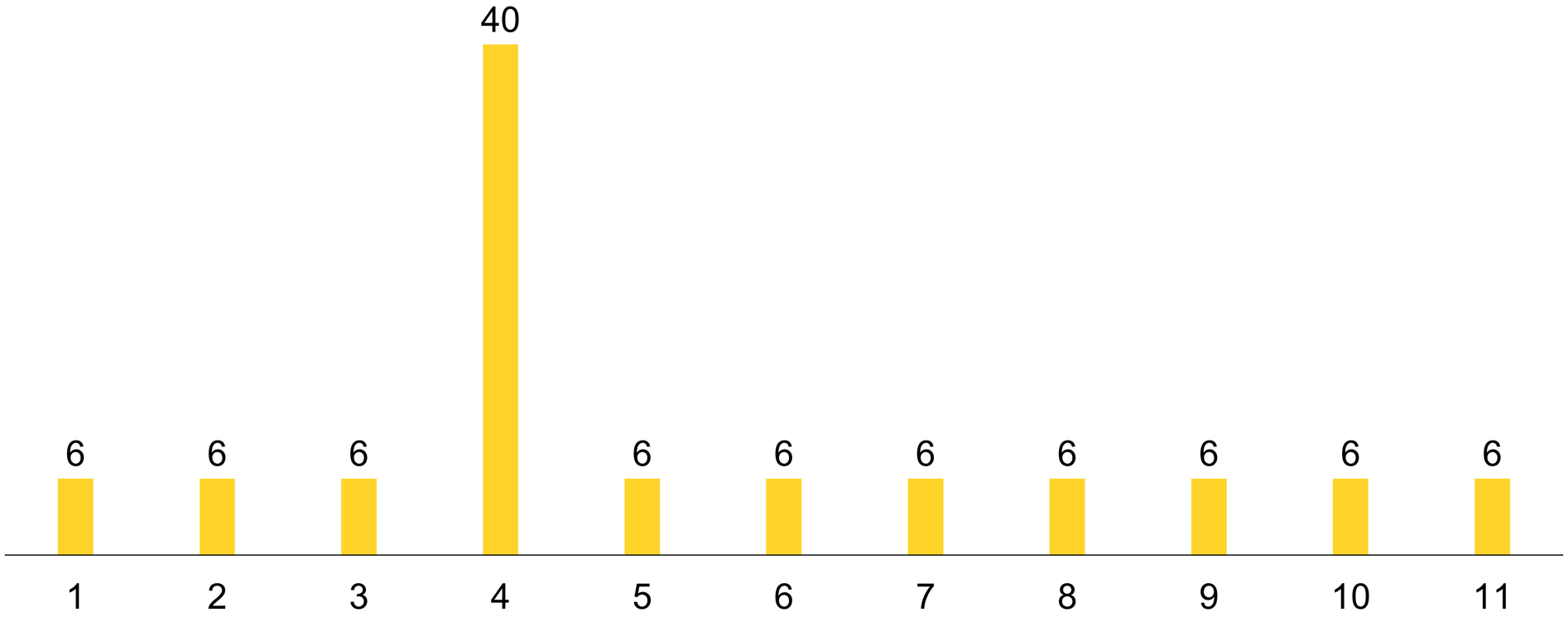
```
struct NonuniformHash {
  size_t operator()(int x) const {
    if (x % 3 == 0) { return 3; }
    return std::hash<int>()(x);
  }
};

std::unordered_set<int, NonuniformHash> numbers;
numbers.max_load_factor(max_load_factor);

for (int i = 0; i < item_count; ++i) {
  numbers.insert(i);
}
```

Равномерность хеш-функции важна

bucket_size, N = 100, max_load_factor = 10

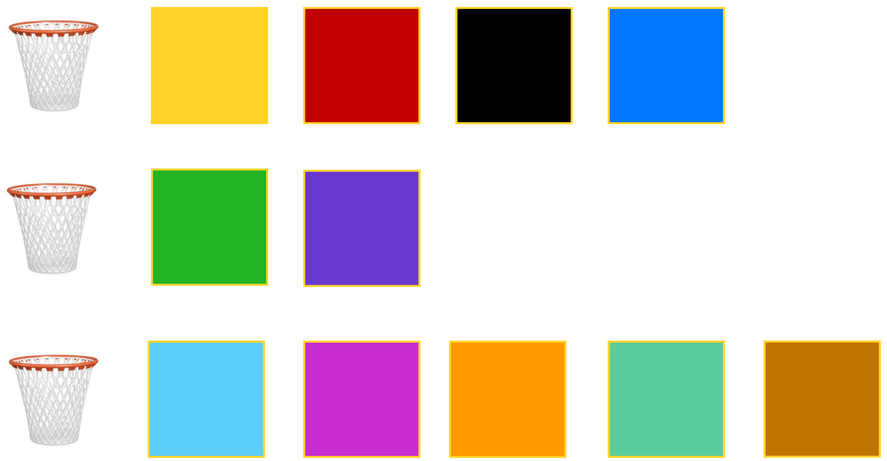


Время работы unordered_set::insert

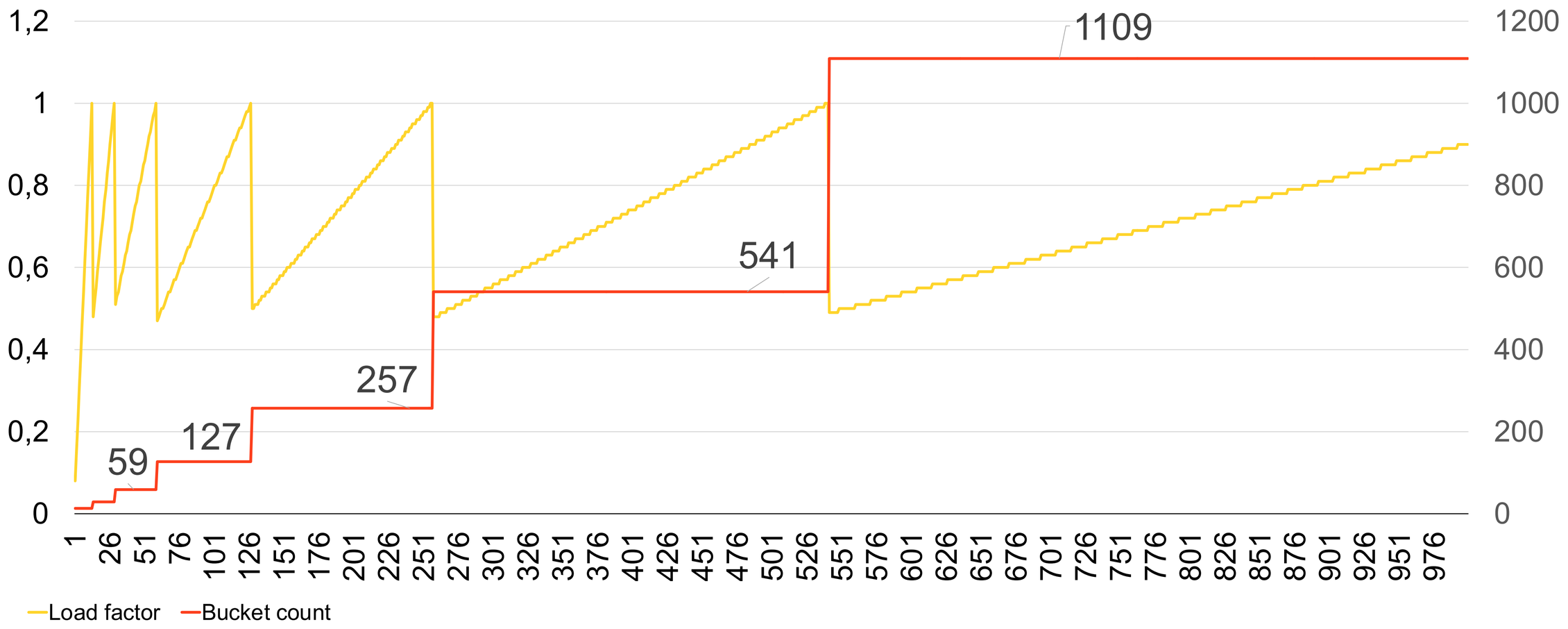
unordered_set::insert — чистое O(1)?

1. Вычислили хеш — O(1)
2. Выбрали по нему бакет — O(1)
3. Вставили в связный список — O(1)

Рехеширование



Рехеширование и load factor



Время работы unordered_set::insert

- Обычно за $O(1)$
- На каждую вставку с номером $O(2^i)$ — $O(N)$

Итого unordered_set::insert работает за амортизированное $O(1)$

Время работы unordered_set::insert

cppreference.com:

- вставка — Average case: O(1), worst case: O(size())
- Чтобы обеспечить уникальность, нужно сначала выполнить поиск

Итоги

Не все O(1) одинаковы

- Вставка, удаление и поиск работают в unordered_set за $O(1)$ **в среднем**
- Это обеспечивается за счёт контроля load factor

Особенности применения std::unordered_set

- Отдельные вызовы insert могут работать за $O(N)$
- insert **может** приводить к инвалидации итераторов

Всегда ли асимптотика решает?

Всегда ли асимптотика решает?

unordered_set

- вставка — O(1)
- поиск по значению — O(1)
- удаление — O(1)

flat_set

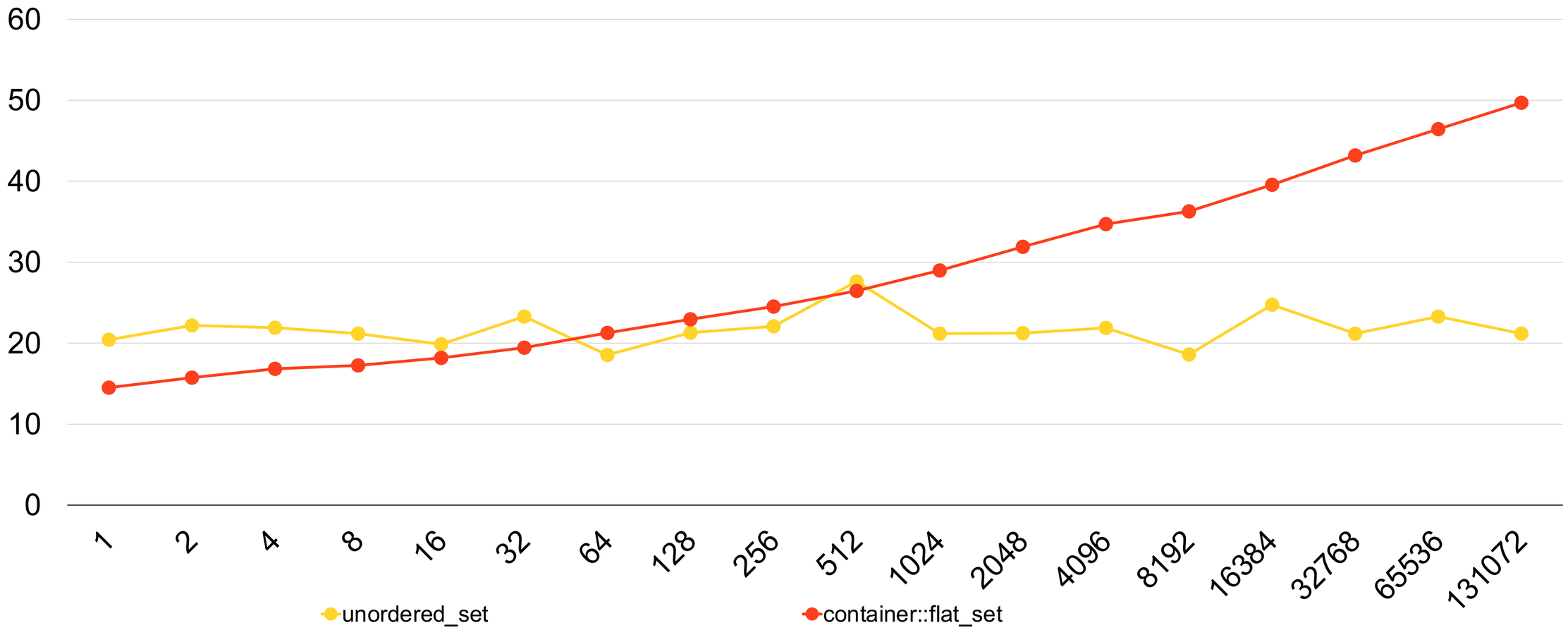
- вставка — O(N)
- поиск по значению — O(log N)
- удаление — O(N)

Код сравнения unordered_set и flat_set

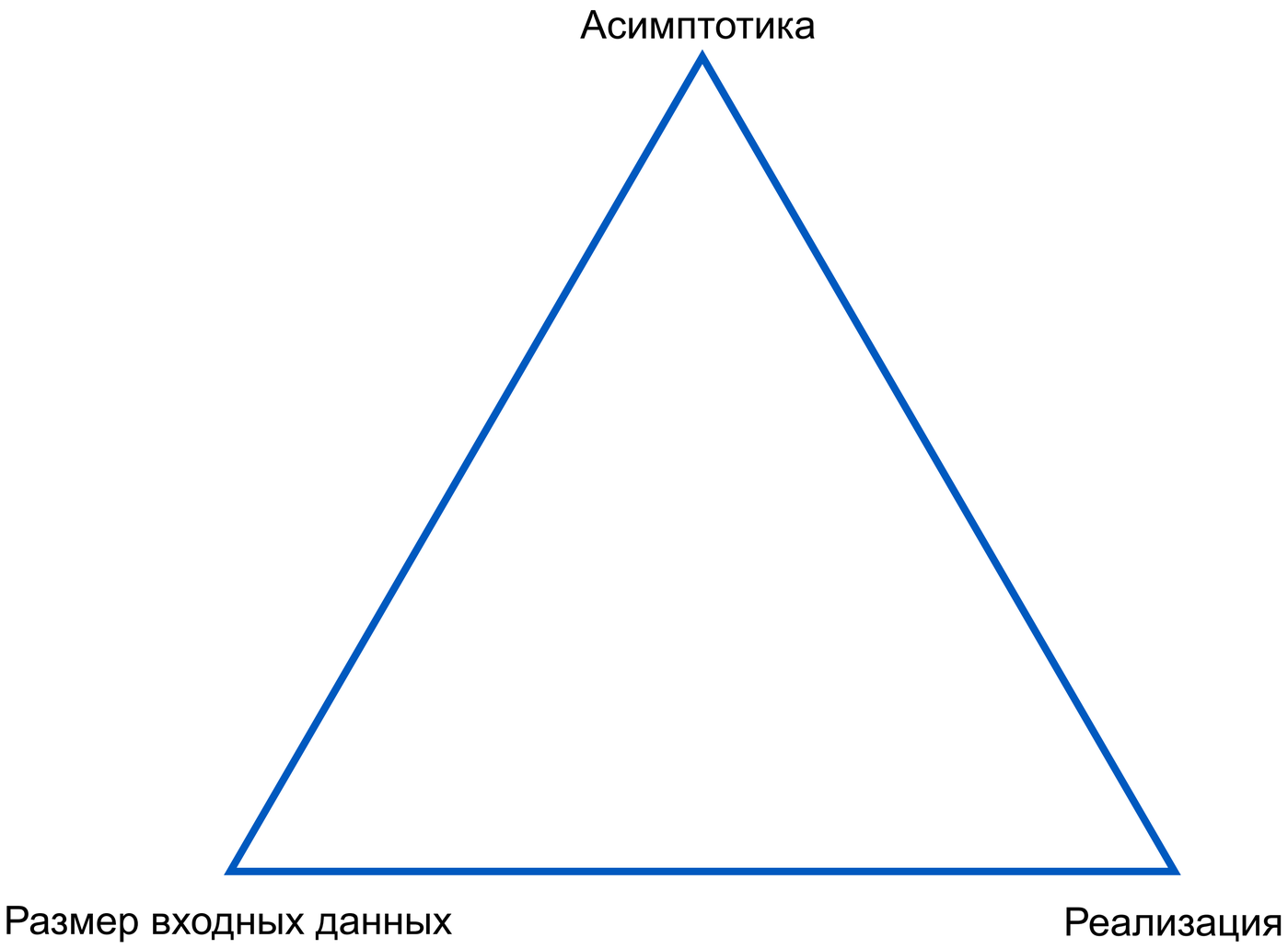
```
template <typename Set> void Find(benchmark::State& state) {
  std::default_random_engine re(20230518);
  const vector<int> values = PositiveRandomInts(state.range(0), re);
  Set sud(values.begin(), values.end());

  std::uniform_int_distribution<size_t> take_absent(1, 100);
  for (auto _ : state) {
    int x = values.front();
    if (take_absent(re) > 50) { x = -x; }
    auto it = sud.find(x);
    benchmark::DoNotOptimize(it);
  }
  state.SetComplexityN(values.size());
}
```

График сравнения unordered_set и flat_set



Треугольник выбора



Главное правило оптимизации кода

Измеряйте!

Итоги

- при выборе контейнера всегда рассматривайте default в первую очередь
- если производительность для вас важна, измеряйте

Общие итоги

- Оценка $O(1)$ означает, что время работы операции не зависит от размера входных данных
- Понимание основ важно, чтобы создавать качественный код и ускорять запуск продуктов
- Проявляйте здоровые интерес и любопытство, чтобы узнавать, как устроено то, чем вы пользуетесь
- Это будет растить вашу квалификацию, а следовательно и счастье

Яндекс

C++ Russia

Спасибо

Илья Шишков

Старший разработчик

ishfb@yandex-team.ru

@ishfb

Яндекс

NDA

Заголовок презентации

Иван Иванов, менеджер

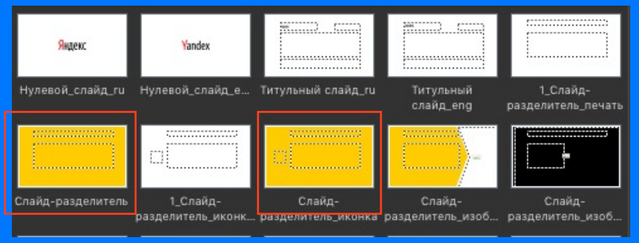
01

Название раздела

Содержание раздела

Этот и другие слайды-разделители с белым фоном оптимизированы для печати

Если презентация не будет печататься, и нужен яркий контрастный фон, то можно выбрать такие же разделители, но с фоном в окне **Создать слайд**



Заголовок слайда

Образец текста

Ключевая мысль

› Маркированный список

1. Нумерованный список

Образец текста

Как выделить ключевую мысль и сделать списки

Выдели текст, зайди на вкладку **Главная** и нажми клавишу **Больший отступ:**

1. **Один раз**, чтобы выделить **ключевую мысль**
   Важно: чтобы жёлтая линия не разрывалась, поставь курсор в конце строки и нажми **Enter**.

2. **Два раза**, чтобы создать **маркированный список** в виде стрелочки

3. **Три раза**, чтобы создать **нумерованный список**

Важно: если нужно вернуться на предыдущий способ выделения, нужно нажать на клавишу **Меньший отступ**

Заголовок слайда

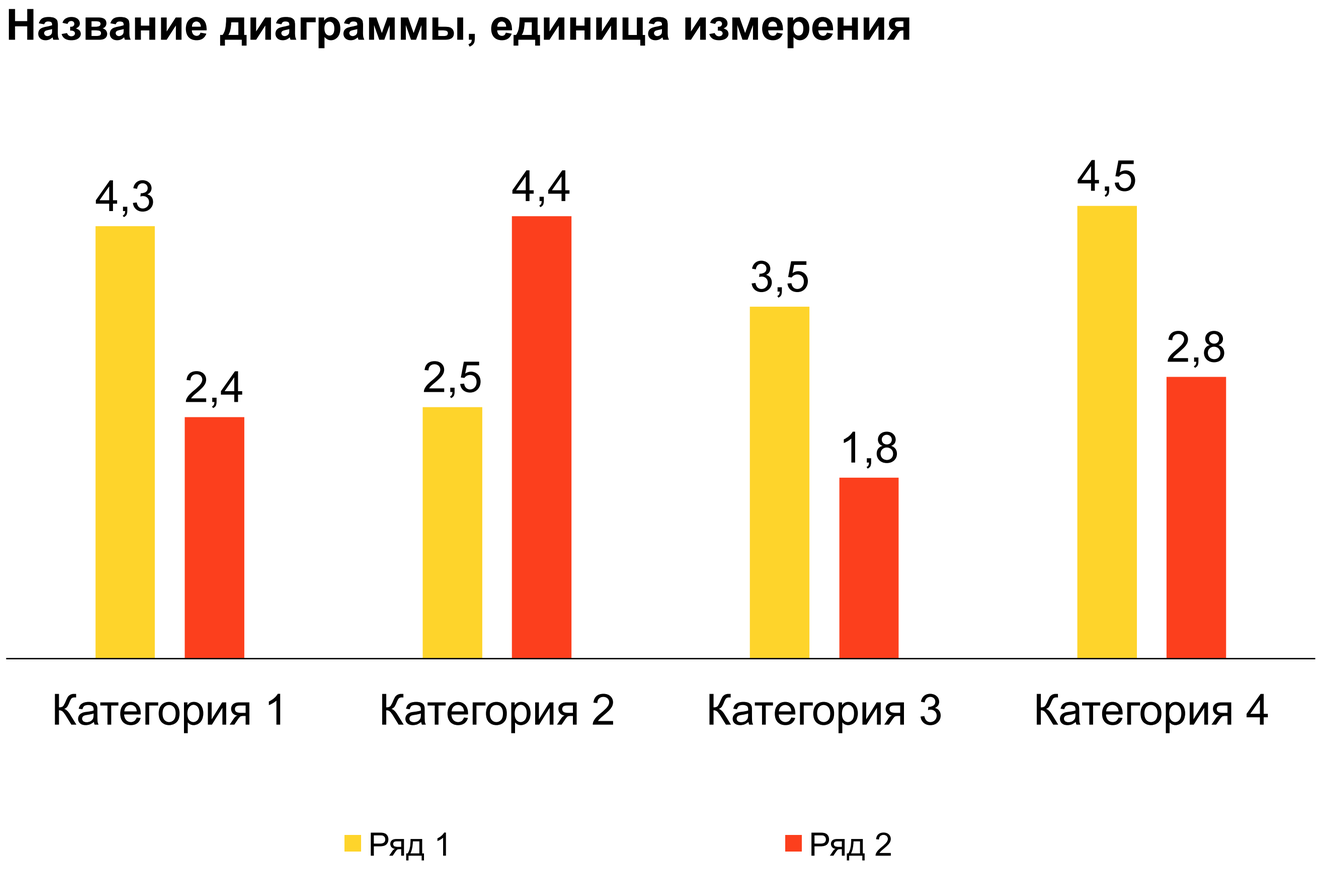
Образец текста

Ключевая мысль

› Маркированный список

1. Нумерованный список

Образец текста



Стиль для оформления таблиц

Текст	Столбец 1	Столбец 2	Столбец 3
Текст 1	8 921	1 114	124
Текст 2	827	9	8 742
Текст 3	654	345	667
Текст 4	3 445	3 456	653
Текст 5	7 668	7 643	65

Правила оформления таблиц

Шапка таблицы должна быть визуально отделена от основной части.

Указывайте **единицы измерения**, если в вашей таблице присутствуют числовые значения. Если единицы измерения одинаковы для всех чисел в столбце, вынесите их в шапку.

Если вы указываете **числовые значения**, сохраняйте единый формат числа в рамках одной единицы измерения: целые, десятичные и т. д.

В столбцах с числовыми значениями принято выравнивать текст по правому краю, для удобства сравнения чисел между собой. Простой текст выравнивается по левому краю.

Не перегружайте информацией таблицу, сохраняйте хорошую **читабельность**.

Оформление таблицы с выделением

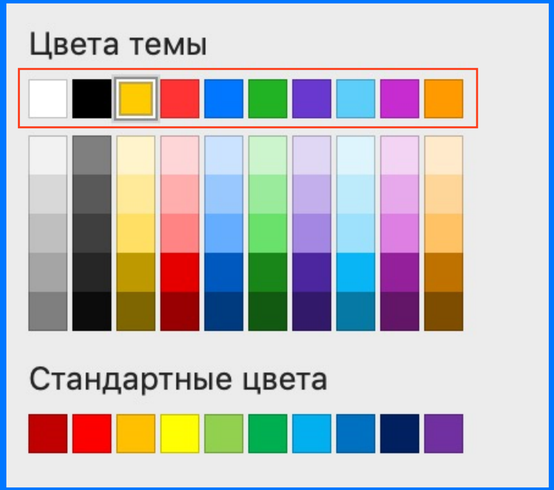
Пример

Текст	Столбец 1	Столбец 2	Столбец 3
Текст 1	10	10	20
Текст 2	10	10	20
Текст 3	10	10	20
Текст 4	10	10	20
Итого:	**40**	**40**	**80**

Выделение цветом

Помимо выделения текста при помощи начертания (жирный, курсив) и цвета, используйте заливку строк и столбцов.

Внимание! Используйте только фирменную палитру Яндекса.



Оформление таблицы с выделением

Пример

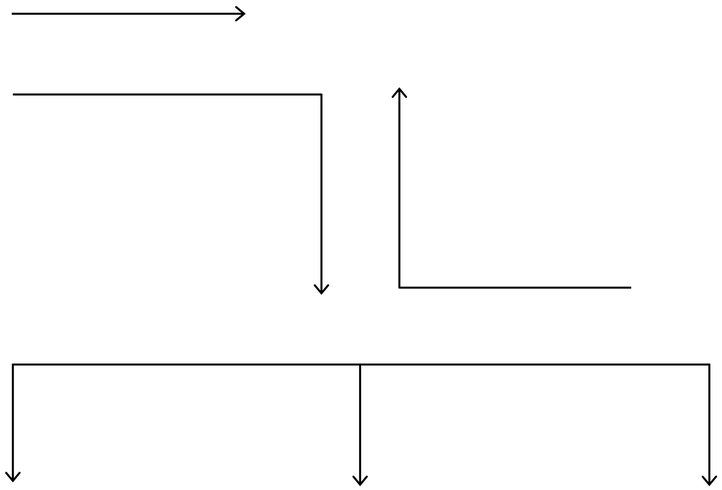
Текст	Столбец 1	Столбец 2	Столбец 3
Текст 1	10	10	10
Текст 2	10	10	10
Текст 3	10	10	20
Текст 4	10	10	10
Итого:	**40**	**40**	**50**

Выделение цветом

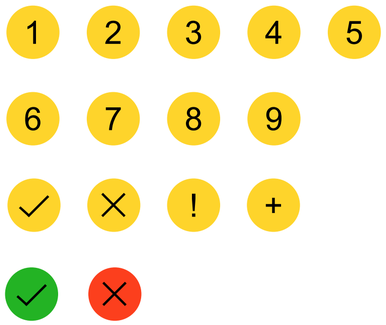
Чтобы подсветить число нестандартным путём, создайте фигуру (например, круг), покрасьте её в один из фирменных цветов и поместите на задний план, предварительно разместив фигуру в нужном месте

Элементы оформления схем

Образцы стрелок



Графические элементы



Образцы текстовых блоков

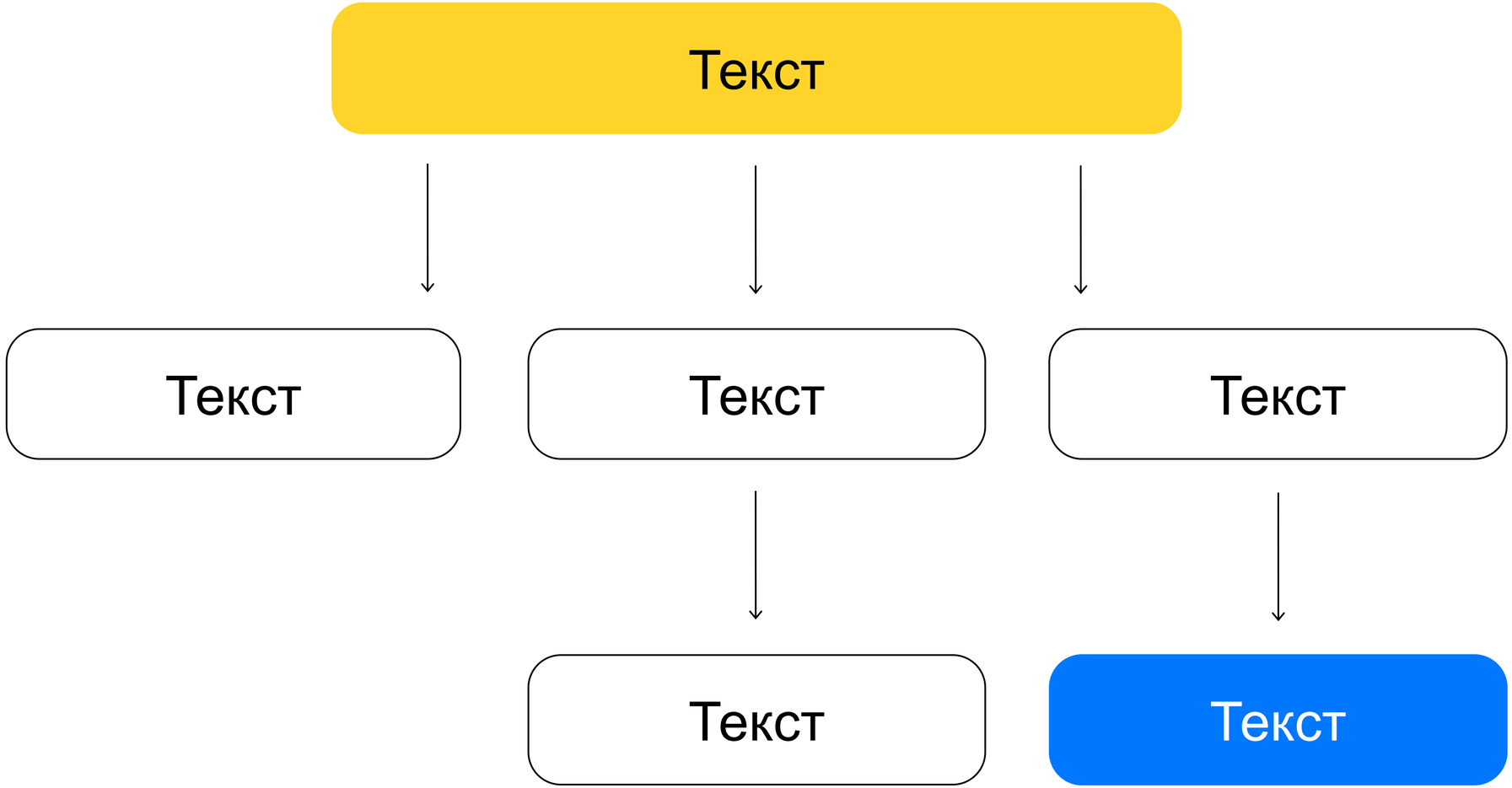
Текст 48 пт

Длинный текст 32 пт

Очень длинный текст 24 пт

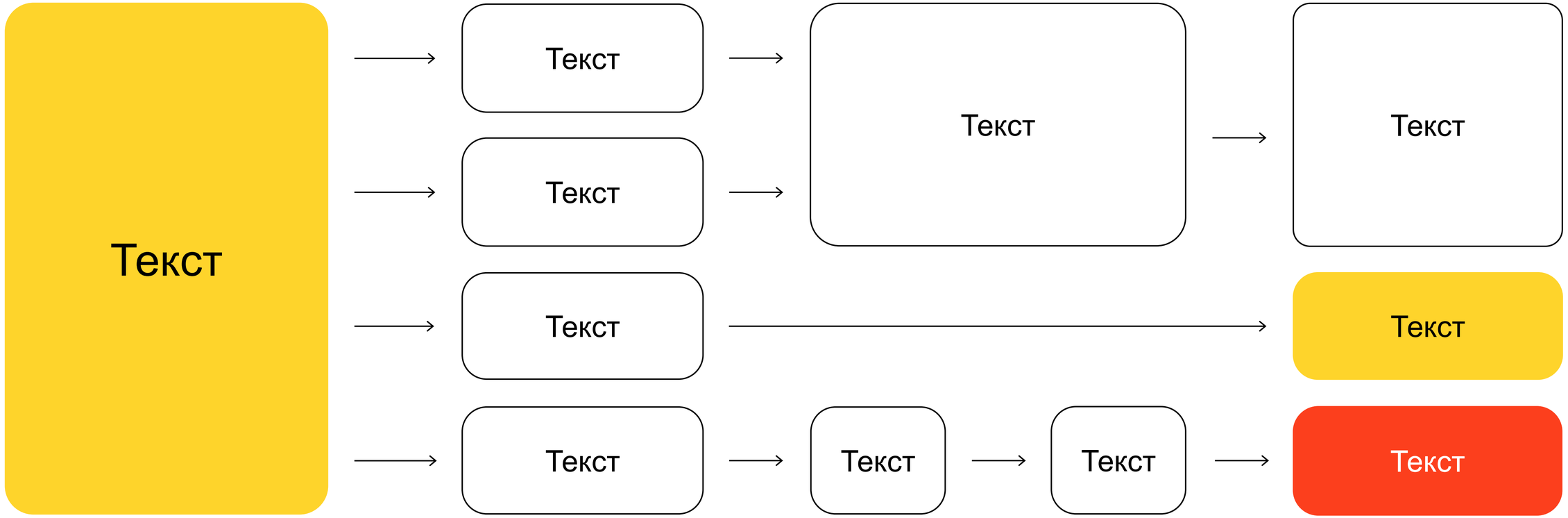
Простая схема

Пример

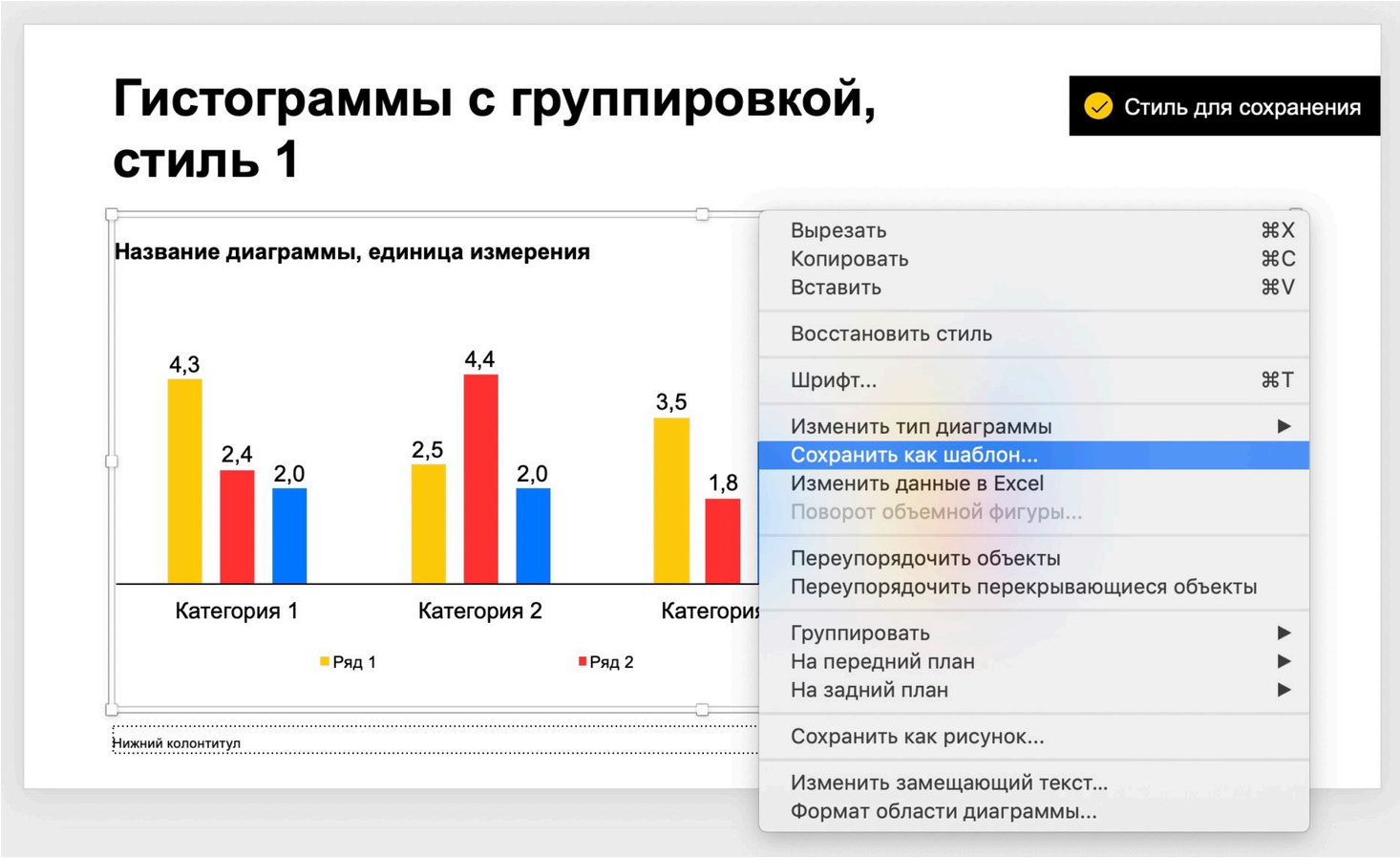


Сложная иерархическая схема

Пример



Как сохранить стиль диаграммы



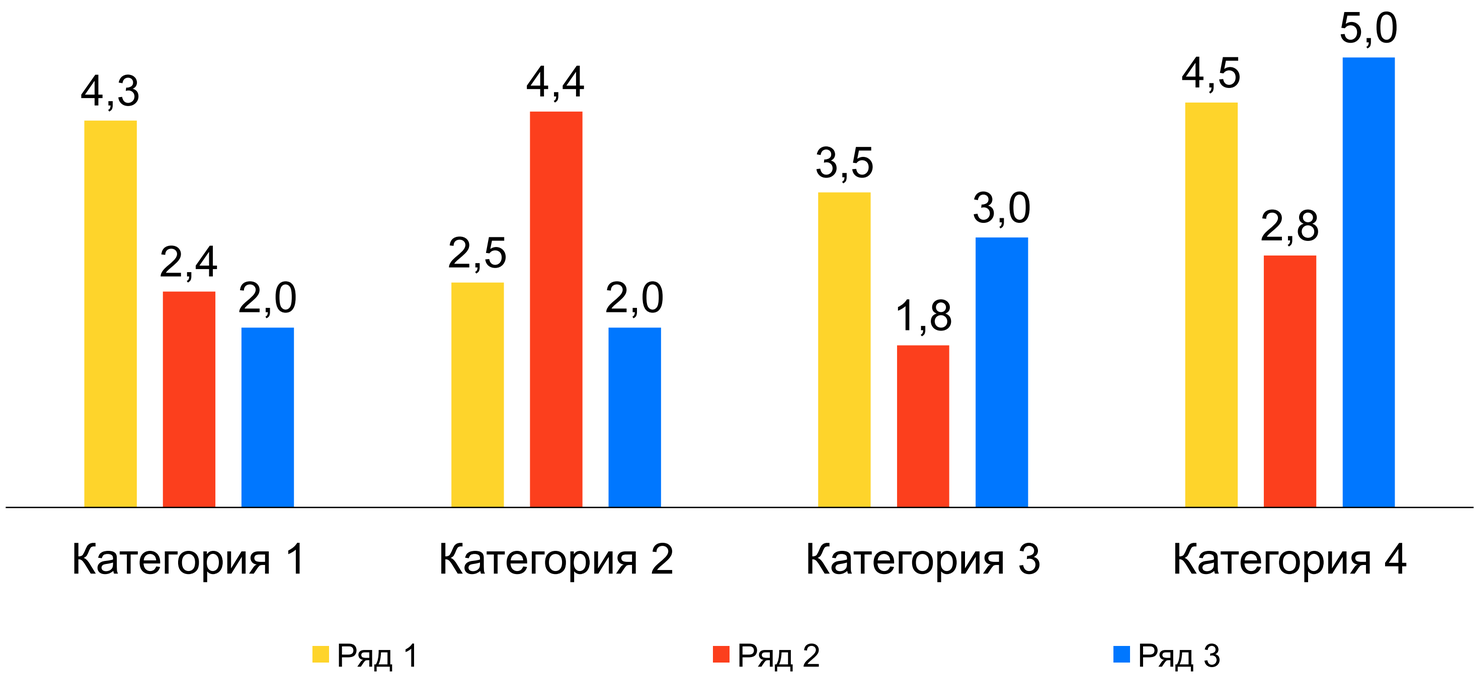
Сохранение стиля диаграммы

1. Перейдите на слайд с меткой **Стиль для сохранения**
2. Выделите диаграмму, нажмите на неё правой кнопкой мыши и в появившемся меню выберите команду **Сохранить как шаблон**
3. Задайте имя и сохраните диаграмму в папку по умолчанию

Гистограммы с группировкой, стиль 1

Стиль для сохранения

Название диаграммы, единица измерения



Правила оформления диаграмм

Указывая название диаграммы, не забудьте указать используемые **единицы измерения** (шт., руб. и т.д.)

Если вы указываете **числовые значения**, сохраняйте единый формат числа в рамках одной диаграммы: целые, десятичные, двоичные.

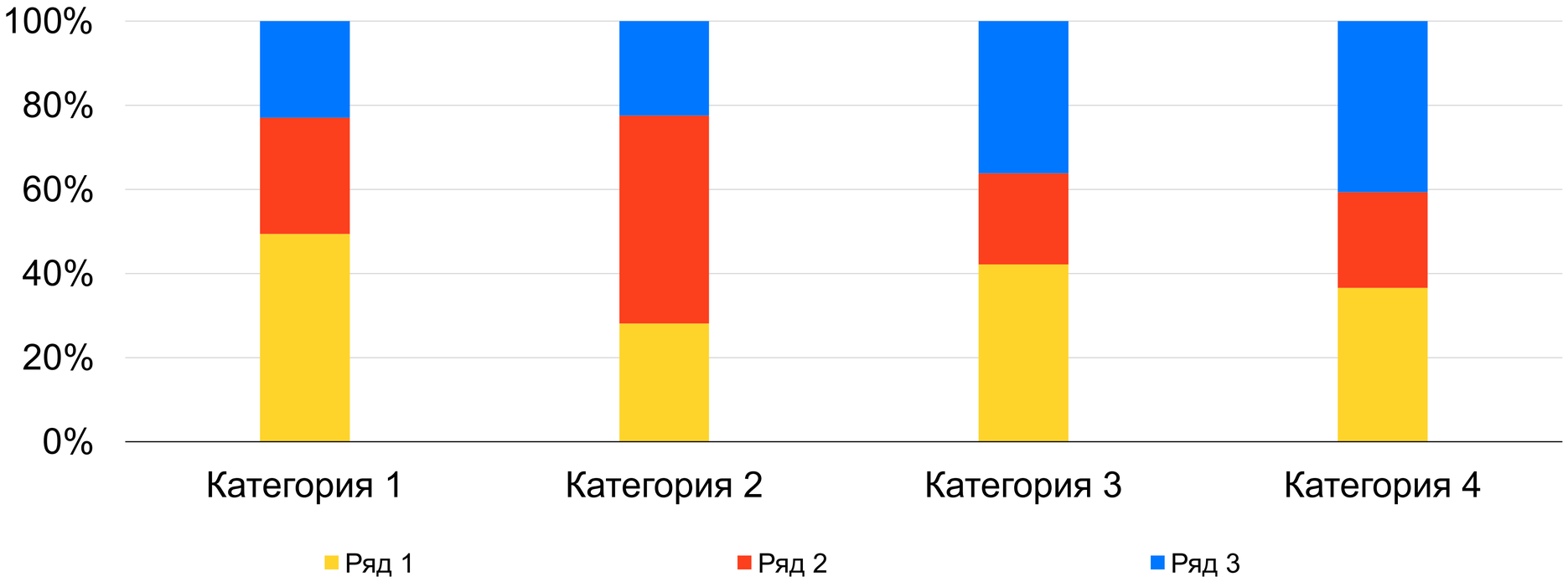
График недопустимо перегружать информацией, необходимо сохранять хорошую **читабельность**.

Расположите **легенду** под диаграммой в случае, если у вас 3 и более рядов, в правом верхнем углу — если рядов 3 и менее.

Гистограмма нормированная с накоплением, стиль 2

Стиль для сохранения

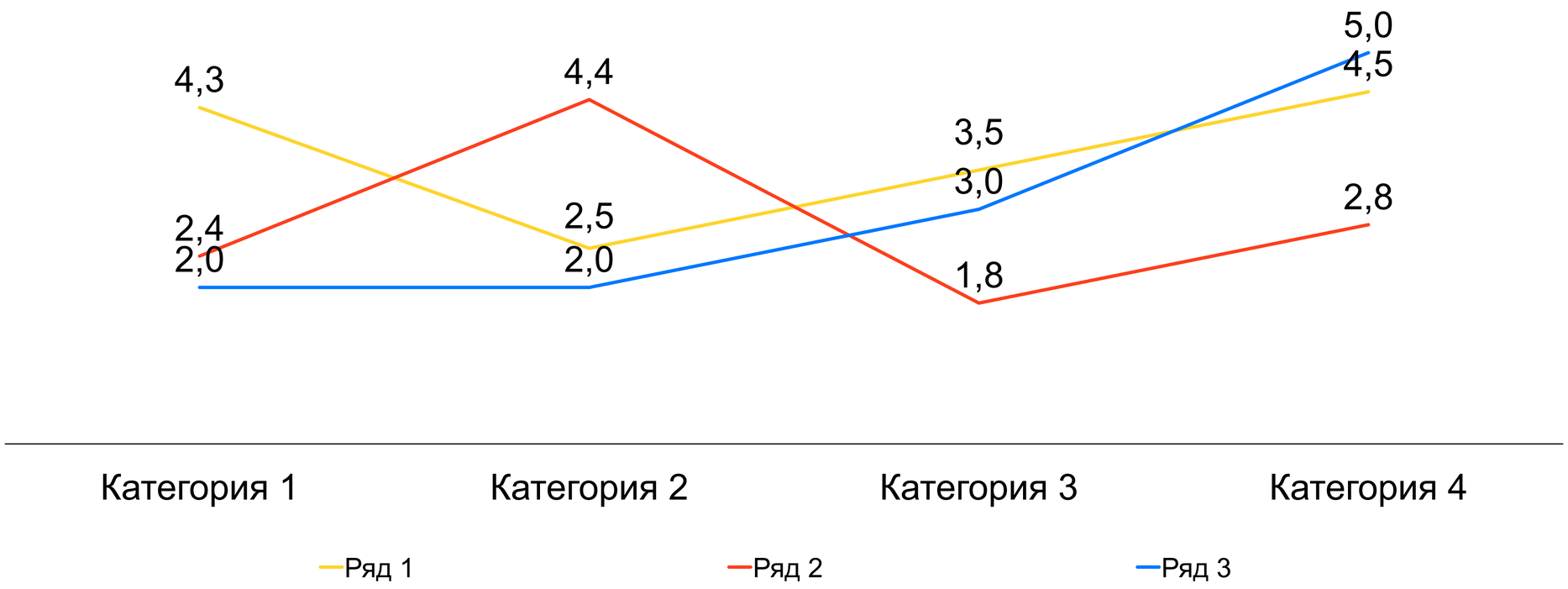
Название диаграммы, единица измерения



График, стиль 3

Стиль для сохранения

Название диаграммы, единица измерения



График, стиль 4

Стиль для сохранения

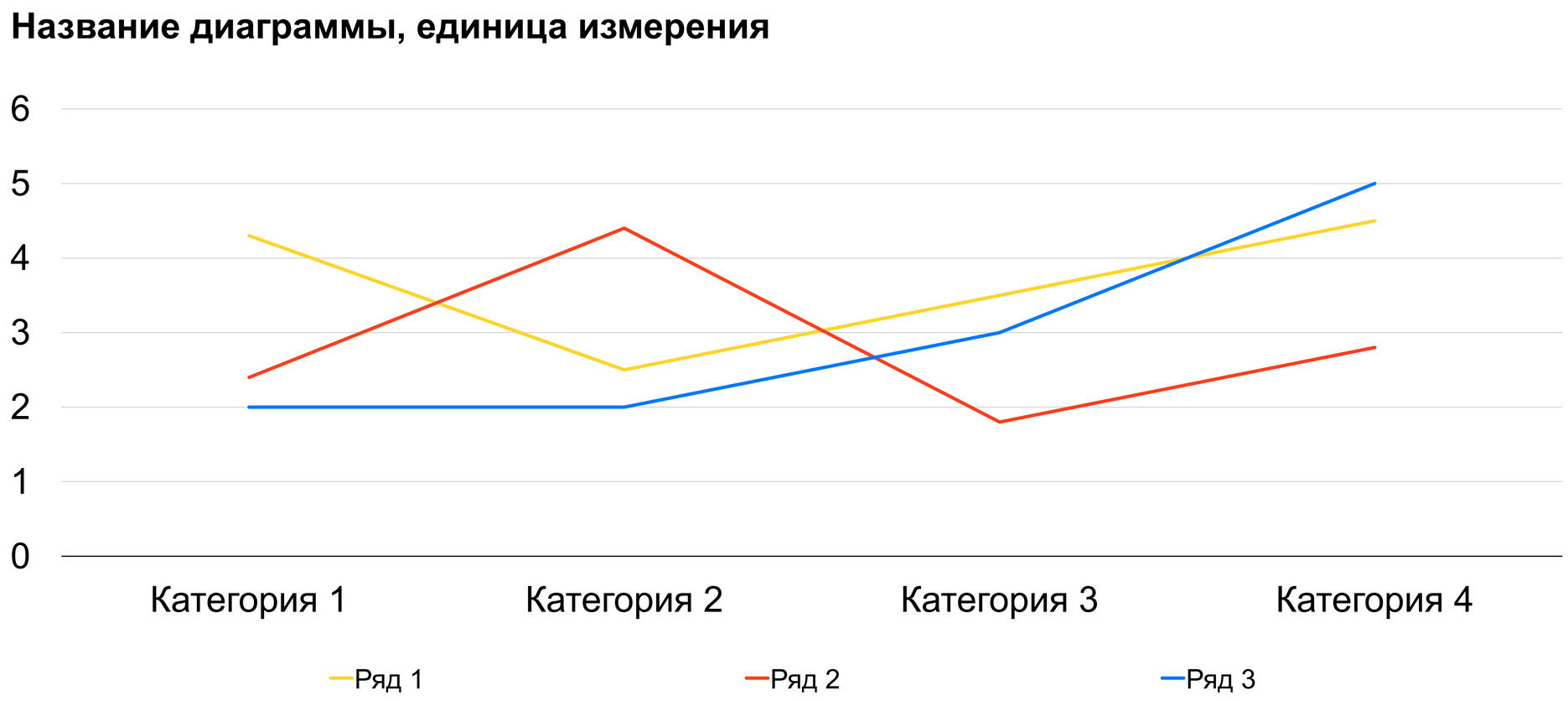


График с маркером, стиль 5

Стиль для сохранения

Название диаграммы, единица измерения

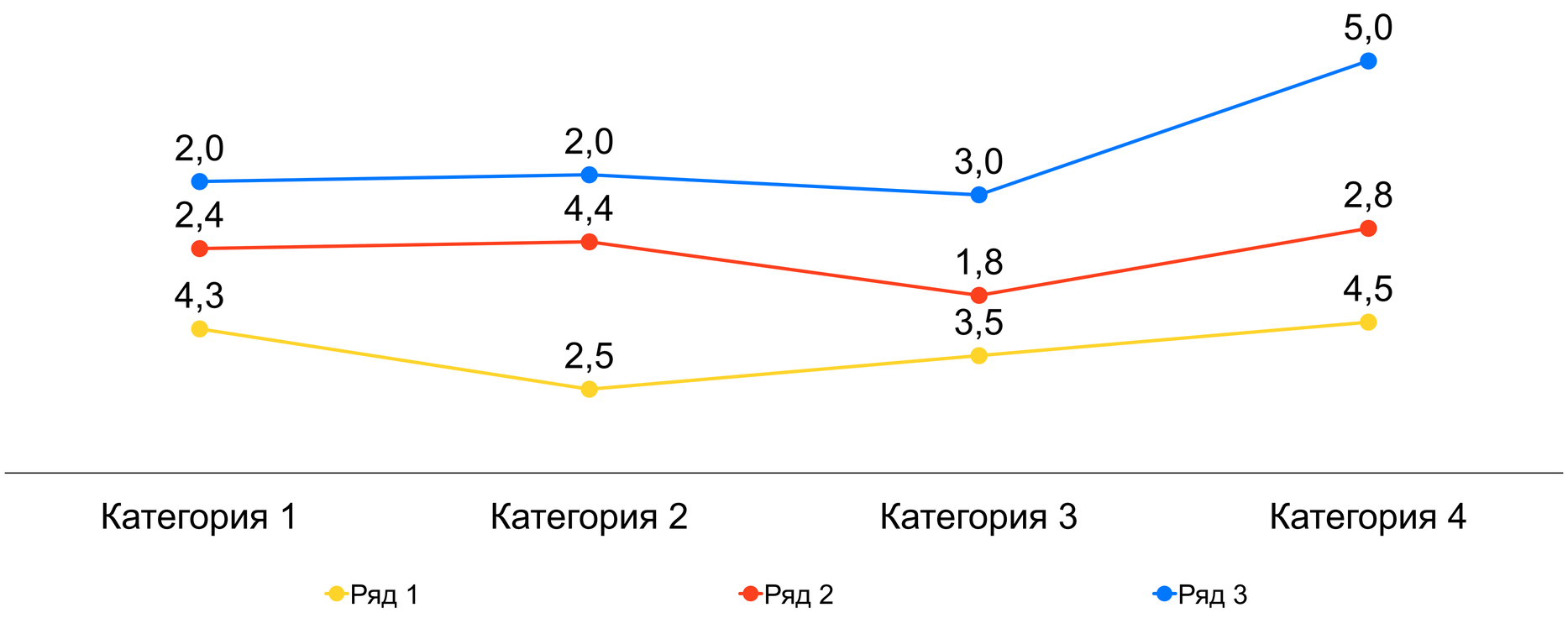
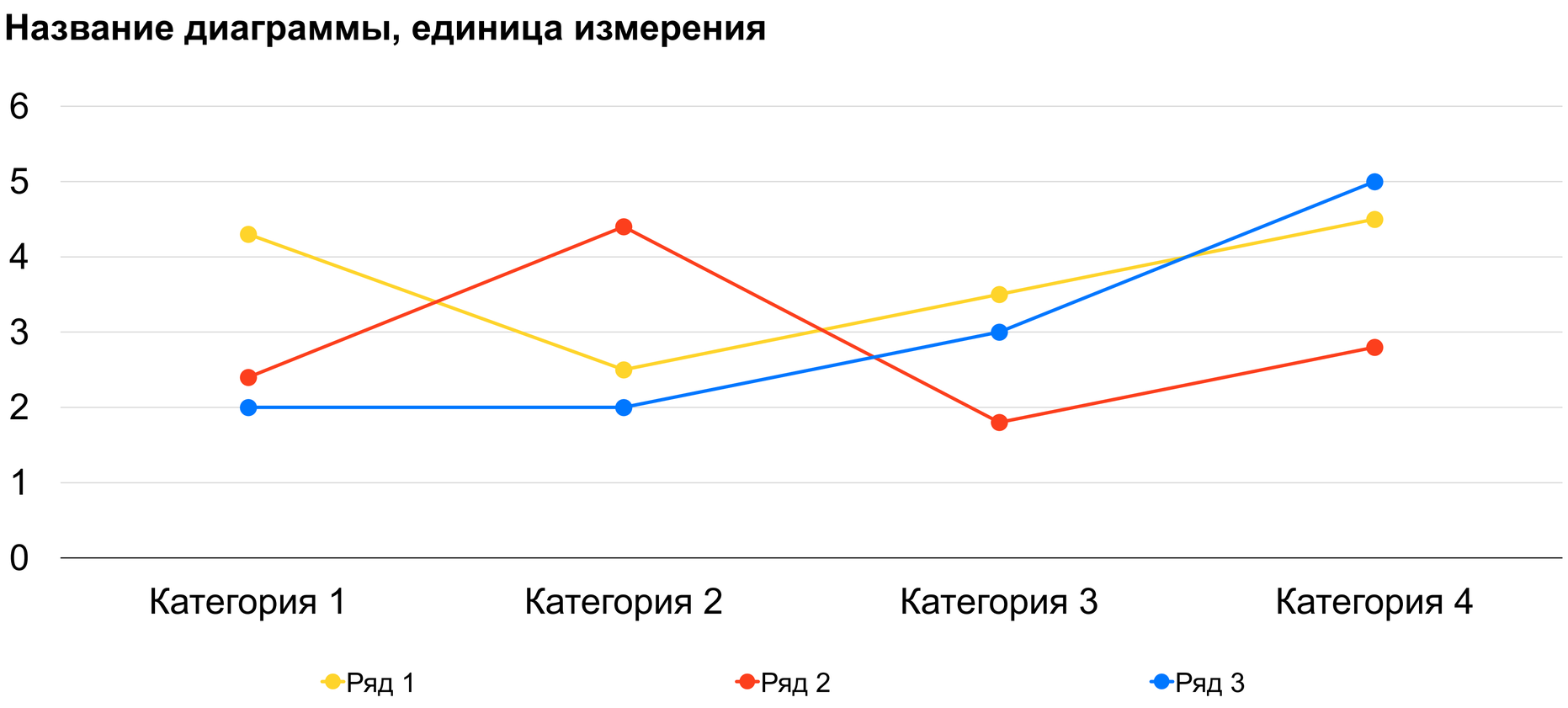


График с маркером, стиль 6

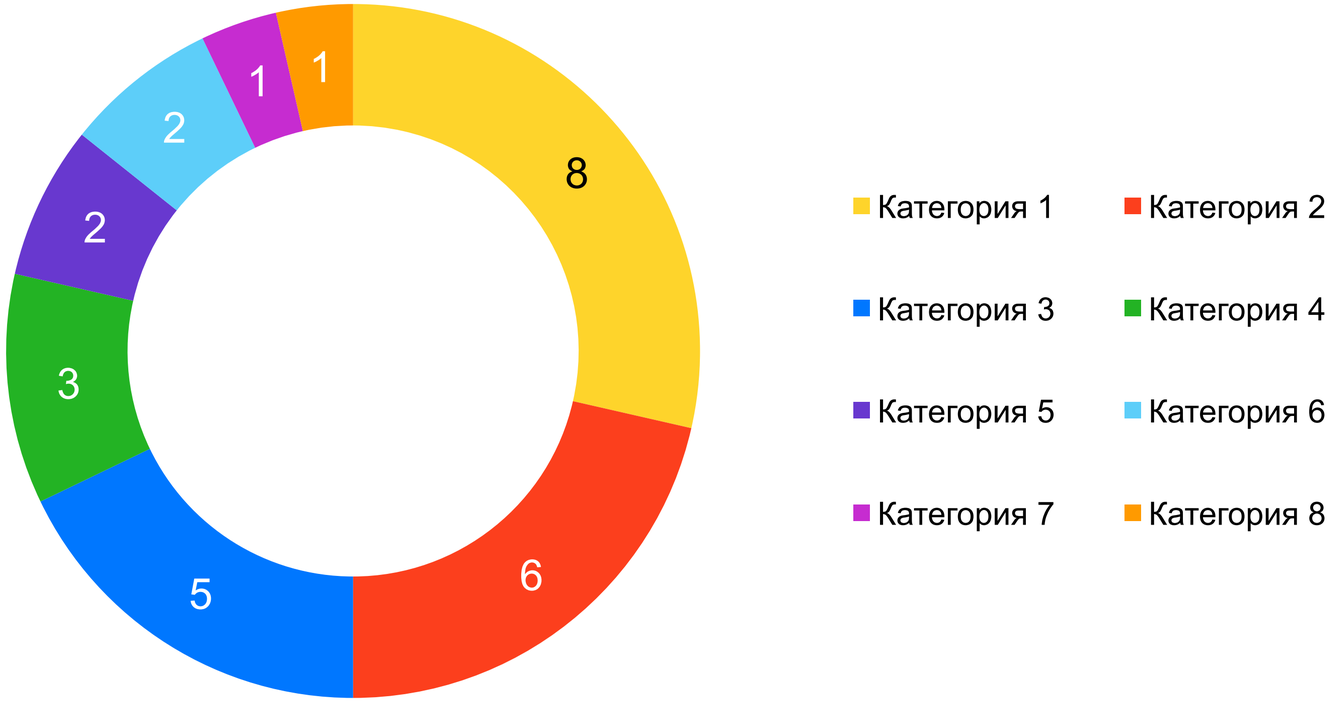
Стиль для сохранения



Круговая диаграмма, стиль 7

Стиль для сохранения

Название диаграммы, единица измерения



Особенности использования круговых диаграмм

Большое количество сегментов круговой диаграммы усложняет сравнение данные между собой.

Чем больше категорий на диаграмме, тем хуже видно маленькие сегменты.

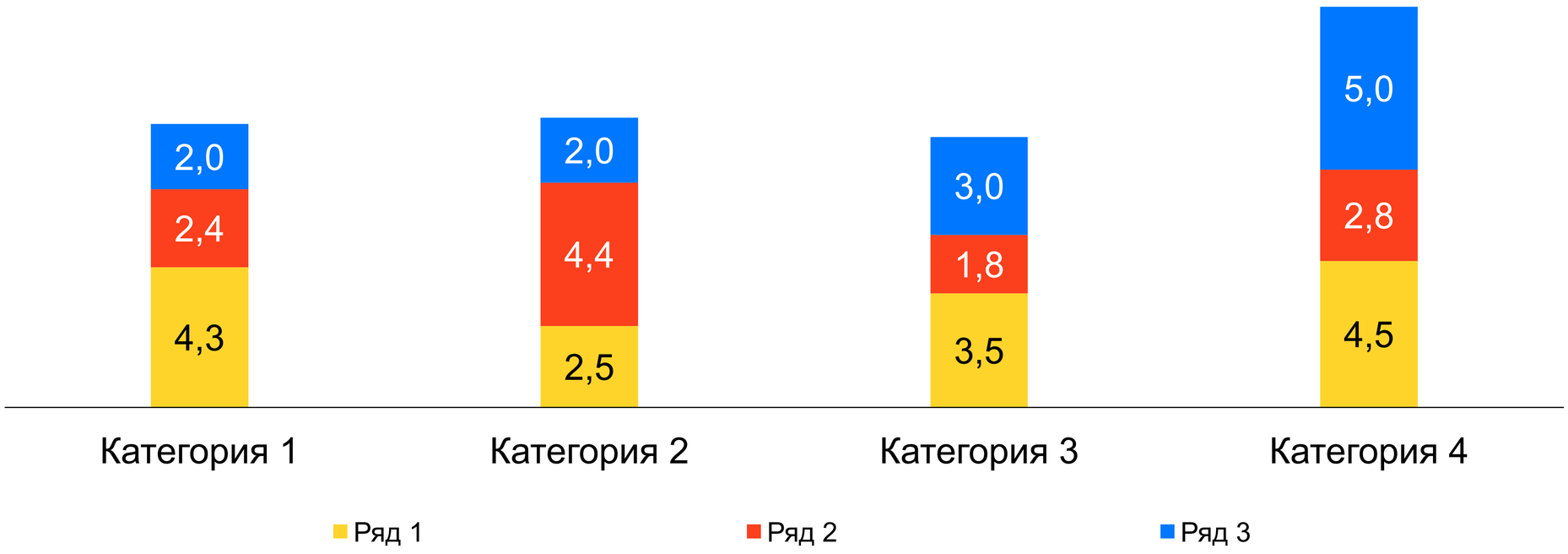
Читаемость диаграммы падает после 5 сегментов, поэтому мы рекомендуем использовать **не более 5 сегментов** для диаграмм небольшого размера, и не более 8 — для крупных.

Круговые диаграммы не подходят для точных сравнений данных по категориям.

Гистограмма с накоплением, стиль 1

Пример

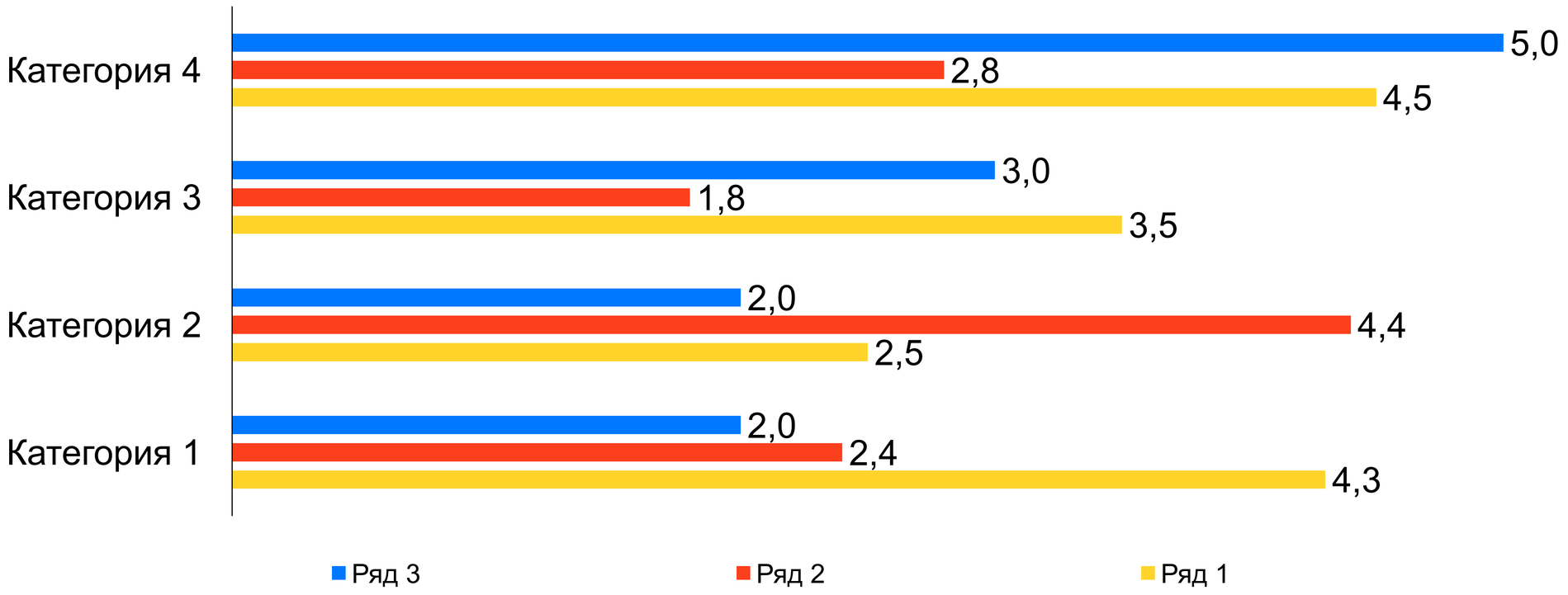
Название диаграммы, единица измерения



Гистограмма линейчатая с группировкой, стиль 1

Пример

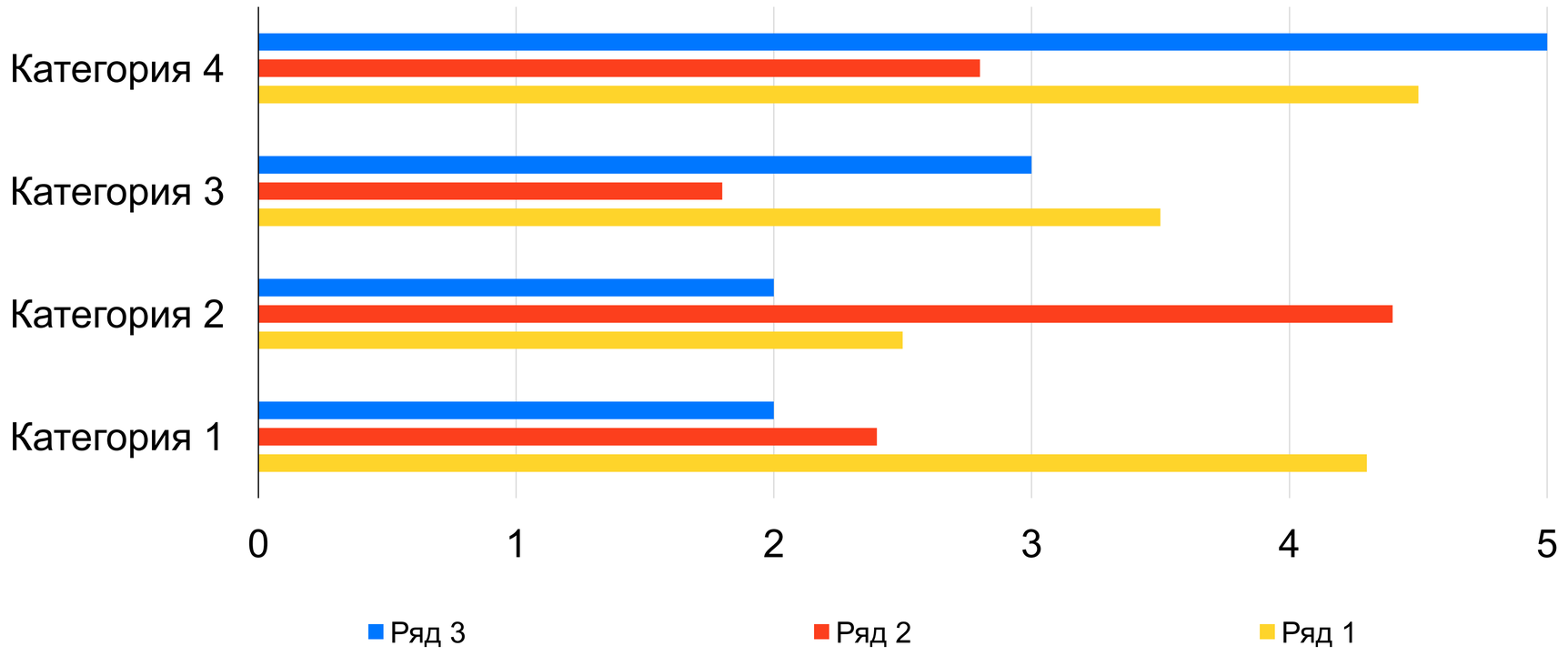
Название диаграммы, единица измерения



Гистограмма линейчатая с группировкой, стиль 1

Пример

Название диаграммы, единица измерения



Круговая диаграмма, оформление с таблицей, стиль 7

Пример

Название диаграммы, единица измерения

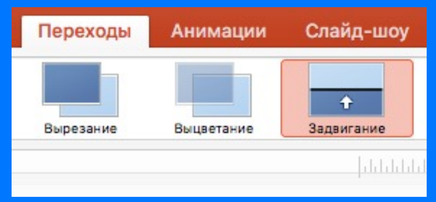
Категория диаграммы 1	8
Категория диаграммы 2	6
Категория диаграммы 3	5
Категория диаграммы 4	3
Категория диаграммы 5	2
Категория диаграммы 6	2
Категория диаграммы 7	1
Категория диаграммы 8	1

Код на чёрном фоне

```
#coding=utf-8
from vanilla import *
from defconAppKit.windows.baseWindow import BaseWindowController
from mojo.events import addObserver, removeObserver
import math
class ShowMouseCoordinatesTextBox(TextBox):


def __init__(self, *args, **kwargs):
```

В случае, если ваш код не помещается на одной странице, его можно продлить на следующем слайде; в разделе **Переходы**, между слайдами поставить переход **Задвигание**



Код на белом фоне

```
#coding=utf-8
from vanilla import *
from defconAppKit.windows.baseWindow import BaseWindowController
from mojo.events import addObserver, removeObserver
import math
class ShowMouseCoordinatesTextBox(TextBox):

def __init__(self, *args, **kwargs):
```

Яндекс

Яндекс

NDA

Иконки для слайда с контактами

Email

Phone

Telegram

Skype

Facebook

Вконтакте

Instagram

Twitter

Bitbucket

Github

Site

LinkedIn

Скопируй иконку с этого слайда

Ctrl + C Cmd + C

и вставь её в бокс «Icon»
для иконки на слайде с контактами:
нажми на бокс, а затем

Ctrl + V Cmd + V