

Язык «Схема»: tutorial с нуля

Для компаратора и мобильного заруба-таймера

Черновик методички

2026-05-28

Содержание

0. Зачем вообще нужен язык «Схема»	2
1. Самая простая схема: работа / отдых	3
2. Непрерывная работа: отдых равен нулю	3
3. Повторение: *N	3
4. Интенсивность: суффикс @	3
Почему intensity важна для модели пульса	4
5. Повторы вместо темпа: @rN	4
6. Темп: @tRPM	4
7. Как сочетаются intensity, reps и tempo	5
Пример HIIT / ВИИТ	5
Пример SIT / СИТ	5
Если intensity не указана	5
8. Группы: несколько атомов как один блок	6
9. Вложенные группы	7
10. Tail: название, голос, метроном и подписи	7
11. Жёсткое завершение # и мягкое продолжение ;	7
12. Tail-no-rest: обратная кавычка после repeat	7
13. Локальная длительность: dur:N(...)	8
14. Отложенные значения: ~ и ~!N	8
15. Активные и отложенные схемы	8
Активная схема	8
Отложенная схема	9
Теоретический график, реальный пульс и прошлая проходка	9
16. Progression: лестницы и изменения параметров	9
17. Tempo ramp: темп внутри атома меняется	10
Адаптивные активные схемы: ^	10

18. Random-схемы: управляемая вариативность	11
19. Глобальные константы	11
20. Как читать любую схему: практический алгоритм	11
21. Типовые ошибки новичков	11
22. Практика: от простого к сложному	12
23. Что делает схему «крутой»	12
24. Предлагаемая лестница обучения	12
Уровень 1. Читать простые схемы	12
Уровень 2. Писать интервальные тренировки	13
Уровень 3. Писать комплексы	13
Уровень 4. Писать AMRAP и отложенные схемы	13
Уровень 5. Писать вариативные и прогрессивные сессии	13
25. Active, deferred, materialized и realized - коротко	13
Active schema	13
Deferred schema	13
Materialized schema	13
Realized schema	14
26. Схемы как язык обмена тренировками	14
27. Визуальный словарь	15
Atom / атом	15
Repeat / повтор	15
Group / группа	15
Scope / локальная область	15
Tail / хвост	15
Progression / прогрессия	15
Ramp / рампа	15
Deferred / отложенное значение	15
Adaptive atom / адаптивный атом	15
28. Короткая шпаргалка	16
29. Что добавить в следующей версии	16

0. Зачем вообще нужен язык «Схема»

Обычный таймер умеет примерно следующее: работа 30 секунд, отдых 30 секунд, повторить 10 раз. Это удобно, пока тренировка простая. Но реальные тренировки быстро становятся сложнее: сначала один темп, потом другой; где-то есть отдых, где-то работа непрерывная; одни упражнения идут по времени, другие по повторам; часть тренировки заранее известна, а часть зависит от того, сколько спортсменов реально сделал.

Язык «Схема» нужен для того, чтобы всё это можно было записать короткой строкой.

20/10@8.5*8#

Это уже полноценная Табата: 20 секунд работы, 10 секунд отдыха, intensity 8.5, 8 повторов, затем жёсткое завершение.

А вот 10 минут гиревого спорта по 20 повторов в минуту:

60/0@r20*10#title=ГС 10 минут

Главная мысль: **схема - это не заметка для человека, а компактная программа тренировки для таймера, графика и анализа.**

Схема используется и в компараторе, и в мобильном заруба-таймере. Компаратор удобен для сравнения разных методов и вариантов сессии. Заруба-таймер работает с одной выбранной схемой в конкретной тренировке, но зато может вести спортсмена по ней в реальном времени.

У обоих инструментов общая идея: **приблизить модель тренировки к реальному физиологическому результату**. Схема задаёт структуру сессии, а параметры спортсмена и intensity помогают заранее построить правдоподобное предсказание пульса, нагрузки, дрейфа и других метрик.

У мобильного таймера есть важное преимущество: он может работать с сердечным монитором. Поэтому во время тренировки можно видеть не только теоретический пульс по модели, но и фактический пульс спортсмена. Дополнительно спортсмен может использовать как опору график своей прошлой проходки той же схемы: день назад, неделю назад, месяц назад.

1. Самая простая схема: работа / отдых

Базовая единица языка - **атом**.

30/30

Читается так: 30 секунд работа / 30 секунд отдых. Первая цифра - ON, то есть рабочая часть. Вторая цифра - OFF, то есть отдых после работы.

Чтобы схема стала законченной тренировкой, обычно добавляют маркер конца:

30/30#

Символ # означает: **выполнить схему и завершить сессию**.

2. Непрерывная работа: отдых равен нулю

Если отдыха нет, его всё равно нужно записать явно:

300/0#

Это 300 секунд непрерывной работы, то есть 5 минут. Важно: лучше не писать просто 300. В языке схема должна явно сказать, есть отдых или нет.

3. Повторение: *N

Чтобы повторить атом несколько раз, используется *N.

30/30*5#

Читается так: 30 секунд работа / 30 секунд отдых, повторить 5 раз, завершить. Полная длительность считается автоматически:

$(30 + 30) * 5 = 300$ секунд = 5 минут

4. Интенсивность: суффикс @

К атому можно добавить параметры через @.

30/30@8*5#

Это значит: 30 секунд работа / 30 секунд отдых, intensity работы 8, повторить 5 раз.

Если нужно указать intensity и для работы, и для отдыха, используется / внутри суффикса:

30/30@8/2*5#

@8/2 полезно для компаратора и таймера: можно увидеть не только структуру времени, но и предполагаемую нагрузку.

Интенсивность можно использовать вместе с целевыми повторами или темпом:

30/30@8r12# intensity 8, цель 12 повторов

30/30@8t40# intensity 8, tempo 40

А вот r и t на одной и той же стороне атома не совмещаются: нужно выбрать либо цель по повторам, либо темп.

Почему intensity важна для модели пульса

Intensity - это не декоративная подпись. Она влияет на графическую модель сессии.

Если записать просто:

30/30*5#

таймер знает интервалы, но почти ничего не знает о характере работы.

А если записать:

30/30@8/2*5#

то появляется модель нагрузки: рабочие отрезки тяжёлые, отдых лёгкий, но не нулевой. На основе такой структуры и индивидуальных настроек спортсмена можно построить более правдоподобный прогноз пульса.

схема + intensity + параметры спортсмена

-> теоретический график пульса и нагрузки

-> сравнение с фактическим пульсом во время тренировки

Чем честнее задана intensity, тем ближе теоретическая кривая к реальной.

5. Повторы вместо темпа: @rN

В некоторых упражнениях важнее не условная intensity, а целевое количество повторений.

60/0@r20#

Читается так: 60 секунд работы, без отдыха, цель - 20 повторов.

Пример:

60/0@r20*10#title=ГС 10 минут

Это 10 минут, где каждая минута имеет цель 20 повторов.

6. Темп: @tRPM

Если нужно задать метроном или темп, используется t.

60/0@t60#

Читается так: 60 секунд работы, tempo 60 ударов/счётов в минуту.

Пример 5-минутной зарубы:

60/0@t60*5#title=Заруба 5 минут

Важно: @r20 и @t60 похожи по смыслу, но это не одно и то же. r20 - цель сделать 20 повторов за блок. t60 - метроном/темп 60 в минуту.

7. Как сочетаются intensity, reps и tempo

Здесь легко ошибиться, поэтому важно сразу запомнить правильное правило.

г и t действительно взаимоисключающие: в одном work-side нельзя одновременно сказать «цель 20 повторов» и «темп 60».

А вот intensity - это отдельный слой. Она может использоваться и сама по себе, и вместе с г, и вместе с t.

60/0@8# 60 секунд с intensity 8
60/0@8r20# 60 секунд, intensity 8, цель 20 повторов
60/0@8t60# 60 секунд, intensity 8, tempo 60

Схема	Что задано
60/0@8#	только intensity
60/0@r20#	целевые повторы, intensity будет взята по умолчанию
60/0@t60#	tempo, intensity будет взята по умолчанию
60/0@8r20#	intensity + целевые повторы
60/0@8t60#	intensity + tempo

Пример HIIT / ВИИТ

15/15@8.5t40/2t20*10`#

Читается так: 15 секунд работы - intensity 8.5, tempo 40; 15 секунд отдыха - intensity 2, tempo 20; повторить 10 раз; последний хвостовой отдых убрать.

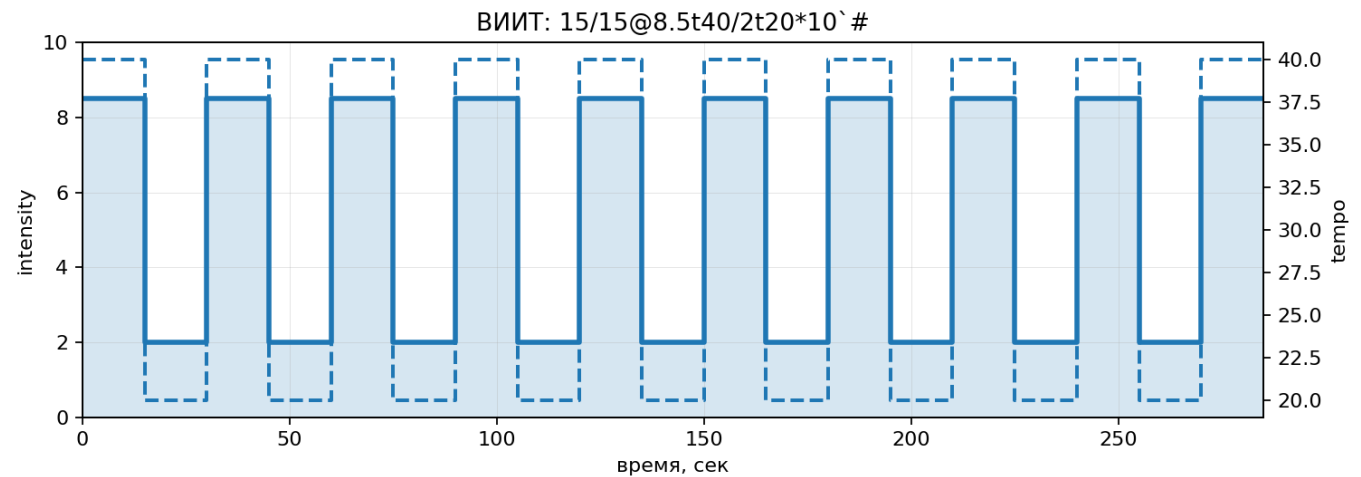


Рис. 1: График ВИИТ

Пример SIT / СИТ

30/210@9.5t30*5`#

Читается так: 30 секунд очень тяжёлой работы - intensity 9.5, tempo 30; 210 секунд отдыха; повторить 5 раз; последний хвостовой отдых убрать.

Если intensity не указана

Если intensity не указана явно, таймер и компаратор не остаются без данных. Они подставляют значение по умолчанию из текущих настроек, чтобы всё равно можно было построить теоретический график пульса.

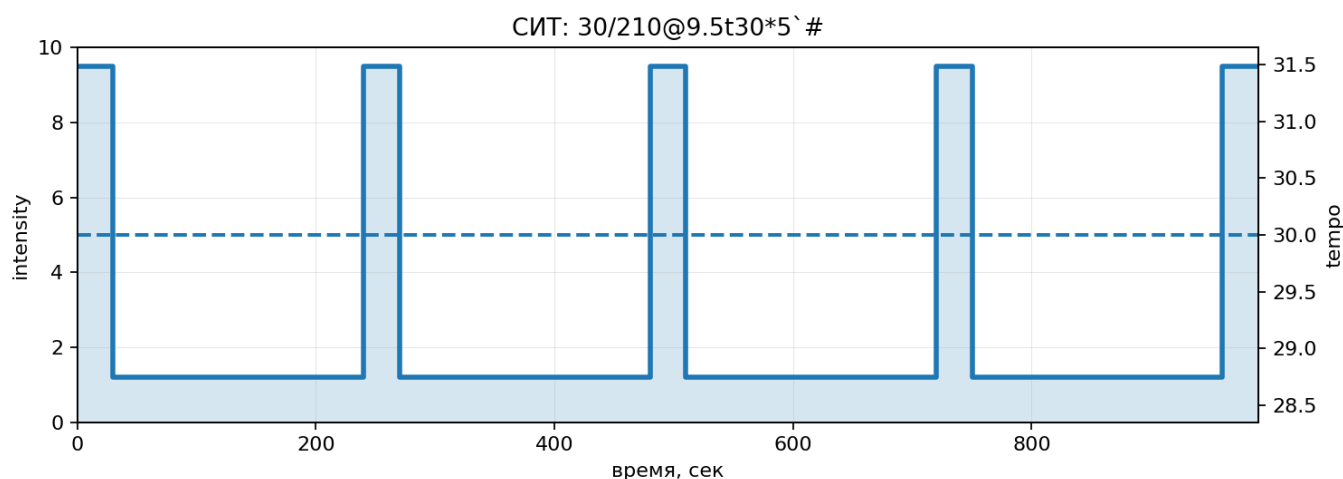


Рис. 2: График СИТ

20/10@5 30/15#

В первом атоме модель будет использовать intensity 5. Во втором атоме intensity не задана, поэтому будет использовано текущее значение по умолчанию.



Рис. 3: Явная и подставленная intensity

8. Группы: несколько атомов как один блок

Иногда повторяется не один атом, а целая последовательность.

(30/30 20/40)*3#

Читается так: сначала 30/30, потом 20/40, и весь этот блок повторить 3 раза.

Группа всегда берётся в скобки. После закрывающей скобки обязательно должен быть repeat.

Правильно:

(30/30 20/40)*3#

Неправильно:

(30/30 20/40)@8*3#

Параметры через @ относятся к атомам, а не к группе целиком.

9. Вложенные группы

Группы можно вкладывать друг в друга.

`((30/30)*2 (20/40)*3)*4#`

Читается так: внутри большого цикла 30/30 повторить 2 раза, затем 20/40 повторить 3 раза; всё это повторить 4 раза.

Полезный способ чтения: сначала найти внешние скобки, понять repeat внешнего блока, потом разобрать внутренние блоки и только потом отдельные атомы.

10. Tail: название, голос, метроном и подписи

После # можно добавлять метаданные.

`60/0@t60*5#title=Заруба 5 минут`

Core-схема:

`60/0@t60*5#`

Metadata:

`title=Заруба 5 минут`

Metadata не меняют физику схемы, но помогают UI и таймеру.

Пример с голосом и метрономом:

`30!1/0@t60 30!2/0@t60 30!3/0@t60 30!4/0@t60#title=Упор на кольцах 2мин 1=уровень пельмень 2=ур`

Здесь !1, !2, !3, !4 - именованные метки. Они получают текст из tail.

11. Жёсткое завершение # и мягкое продолжение ;

Обычно схема заканчивается #.

`20/10*8#`

Это значит: выполнить схему и закончить.

Есть ещё ;:

`20/10*8;`

Смысл другой: выполнить схему один раз, а потом остаток сессии добить стандартным механизмом таймера.

Для новичка правило простое: почти всегда используй #; ; нужен для специальных случаев.

12. Tail-no-rest: обратная кавычка после repeat

Иногда отдых нужен между повторами, но не нужен после последнего повтора.

Обычная схема:

`40/30*5#`

Тут после пятого рабочего блока тоже будет 30 секунд отдыха.

Схема без последнего отдыха:

40/30*5`#

Обратная кавычка после repeat говорит: в последнем повторе убрать финальный отдых.

13. Локальная длительность: dur:N(...)

Иногда вся схема длится не столько, сколько получается арифметически, а имеет локальный бюджет времени.

Пример короткой «Синди» на 30 секунд:

```
dur:30((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)# manual=1 metro=0 title=Синди короткий тест 1=подтягивания
```

Главная часть:

```
dur:30((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)#
```

Читается так: внутри 30 секунд повторять круг: подтягивания цель 5; отжимания цель 10; приседания цель 15; сколько успеешь.

Важно: длительность пишется в соге-части схемы, а не в хвосте.

Правильно:

```
dur:30((...)*~)#
```

Неправильно:

```
(...)*~# dur=30
```

14. Отложенные значения: ~ и ~!N

До сих пор схемы были активными: таймер заранее знает длительность каждого блока.

Но в реальной тренировке бывает иначе. Например, спортсмен делает подтягивания не фиксированные 20 секунд, а столько, сколько реально занимает подход.

Для этого есть отложенные значения.

```
~/0@r~*~#title=блокнот
```

Это «блокнот»: длительность работы неизвестна заранее; отдых 0; количество повторов неизвестно заранее; количество подходов неизвестно заранее.

Именованный слот:

```
~!1/0@r5# 1=подтягивания
```

~!1 значит: длительность будет определена во время выполнения, а метка 1 берётся из metadata.

Если есть !1 или ~!1, в хвосте обязательно должно быть 1=...

15. Активные и отложенные схемы

Активная схема

Активная схема заранее знает, что будет происходить.

```
20/10@8.5*8#
```

Таймер может построить полный график до старта.

Отложенная схема

Отложенная схема содержит ~.

~/~@r~*~#title=блокнот с отдыхом

Таймер не знает полный график заранее. Он узнаёт его по мере выполнения.

Активная схема хороша для интервальных тренировок, темповых сессий, метронома, сравнения планов в компараторе и предварительного прогноза пульса. Отложенная схема хороша для AMRAP, комплексов с ручной фиксацией, дневникового режима и случаев, где фактическое выполнение важнее заранее заданного времени.

Теоретический график, реальный пульс и прошлая проходка

В активной схеме таймер может заранее построить теоретическую кривую: как примерно должен вести себя пульс при заданной структуре, intensity и индивидуальных параметрах спортсмена.

Во время тренировки мобильный таймер может получать фактический пульс с сердечного монитора. Тогда на одном экране появляются два слоя: теоретический пульс и фактический пульс. Можно добавить и третий слой: прошлую проходку этой же схемы.

Так схема становится якорем для сравнения. Спортсмен сравнивает не абстрактные тренировки, а одну и ту же формальную структуру, выполненную в разные дни.

16. Progression: лестницы и изменения параметров

Progression - это способ сказать: «повторяй, но каждый следующий шаг немного меняй».

30/30->[for:3,delta:-5/+5]#

Читается так: начать с 30/30; сделать 3 шага; на каждом шаге работу уменьшать на 5 секунд, отдых увеличивать на 5 секунд.

Развёртка:

30/30

25/35

20/40

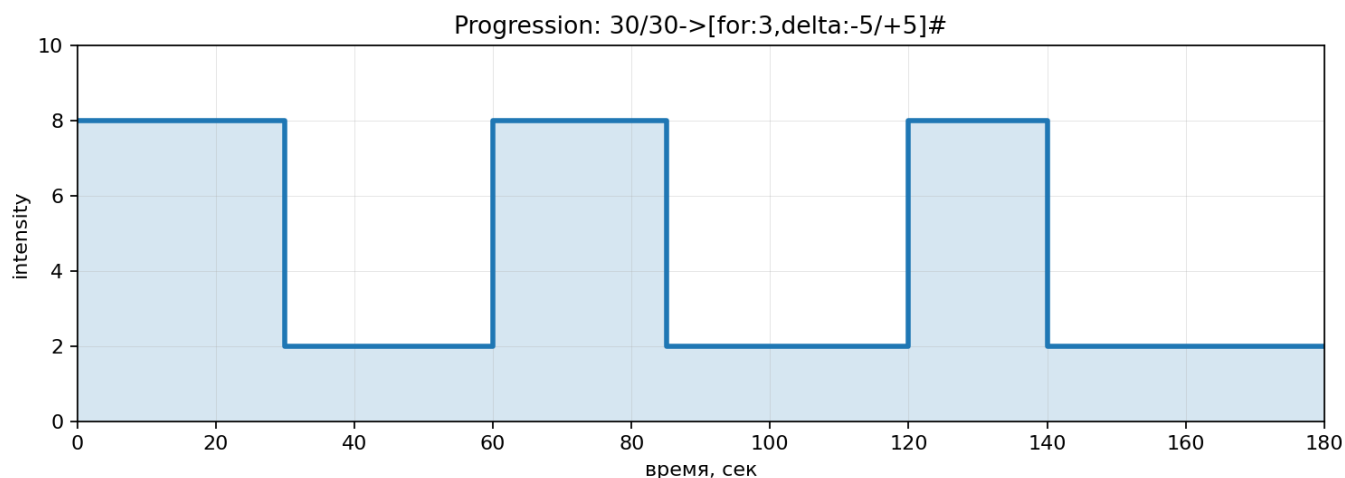


Рис. 4: График progression

Progression особенно полезен для лесенок.

Пример подтягиваний:

```
(2/2@r1->[to:auto/auto@r10, delta: +2/+2@r+1]->[to:auto/auto@r1, delta: -2/-2@r-1])*1`#
```

Смысл: начать с 1 повтора, подняться до 10, затем спуститься обратно до 1.

17. Tempo ramp: темп внутри атома меняется

Иногда темп должен меняться не от блока к блоку, а прямо внутри одного рабочего отрезка.

```
60/10@t[15->25]#
```

Читается так: 60 секунд работы с разгоном tempo от 15 до 25, затем 10 секунд отдыха.

Если добавить delta, можно задать шаг изменения темпа:

```
60/10@t[15->25 delta:2]#
```

На уровне языка это один атом, но runtime может материализовать его как несколько внутренних кусочков.

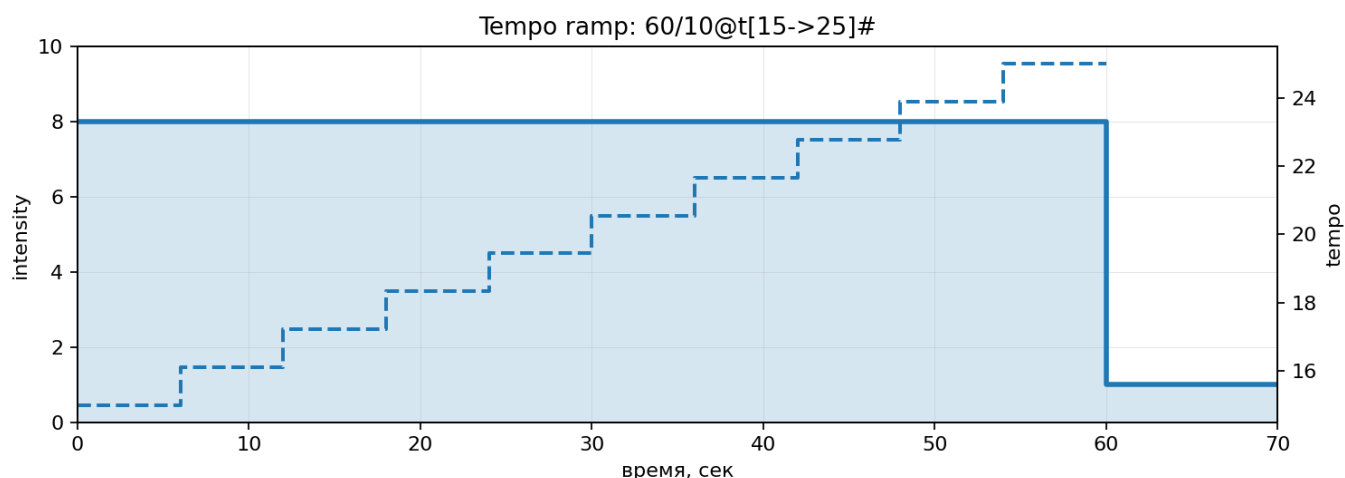


Рис. 5: График tempo ramp

Пример:

```
58/2@t[18->26]*10`#title=армейский рывок с ускорением
```

Это 10 рабочих блоков с ускорением внутри каждого блока и без финального отдыха после последнего.

Адаптивные активные схемы: ^

Есть особый продвинутый вариант активных схем - адаптивные атомы, помеченные ^.

```
300/0@t180^#
```

Обычная активная схема заранее задаёт темп. Адаптивная активная схема тоже имеет теоретический план, но во время выполнения спортсмен может менять tempo голосовой командой или кнопками: ускорить, замедлить, поставить конкретный tempo.

Для новичка достаточно запомнить: без ^ tempo - заранее заданный режим; с ^ tempo может управляемо меняться во время активной сессии; такая схема всё равно остаётся активной, а не превращается в свободный блокнот.

18. Random-схемы: управляемая вариативность

Для некоторых тренировок полезно не заранее прописывать точный порядок, а задать семейство вариантов.

```
random(30/0@9t220 30/0@6t140 30/0@7t180 turb:7 dur:580 min:8) 20/0@t[200->280]#title=бег битый
```

Идея: есть несколько возможных атомов; runtime выбирает их в переменном порядке; dur задаёт длительность random-блока; turb задаёт степень турбулентности; min задаёт минимальное удержание выбранного атома.

random - это не хаос, а контролируемая вариативность с правилами.

19. Глобальные константы

Чтобы не повторять числа, можно вынести их в начало.

```
[low=12 high=24](150/0@t[low->high] 145/0@t[high->low] 0/10)*2#title=рывок непрерывная лесенка
```

Константы low и high потом используются внутри схемы. Это делает схему легче менять.

20. Как читать любую схему: практический алгоритм

Когда видишь новую схему, не пытайся понять всё сразу. Разбери по слоям.

1. Найди конец core-схемы: до # или ; исполняемая часть, после - metadata.
2. Найди большие скобки.
3. Найди repeats: *5, *~, *5`.
4. Разбери атомы: ON/OFF, @, r, t.
5. Прочитай хвост: title=..., l=..., voice=1, metro=1.

Пример:

```
dur:1200((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)# manual=1 metro=0 title=Синди 1=подтягивания 2=отжимания
```

Читается как: 20 минут повторять подтягивания 5, отжимания 10, приседания 15, фиксируя фактическое время каждого упражнения.

21. Типовые ошибки новичков

Пробелы внутри атома:

```
30/30 @8 t20 # нельзя  
30/30@8      # правильно
```

Дробный repeat:

```
30/30*3.5#    # нельзя
```

Попытка повесить @ на группу:

```
(30/30 20/40)@8*3#    # нельзя  
(30/30@8 20/40@8)*3#  # правильно
```

Темпо как отложенное значение:

```
30/0@t~#       # нельзя
```

Забыли metadata для !1:

```
30!1/0#        # нельзя  
30!1/0# 1=подтягивания # правильно
```

22. Практика: от простого к сложному

Простая интервальная сессия:

20/40*10#

То же самое, но работа на 8, отдых на 2:

20/40@8/2*10#

Табата:

20/10@8.5*8#

5 минут метронома:

60/0@t60*5#title=Заруба 5 минут

10 минут по повторам:

60/0@r20*10#title=ГС 10 минут

Комплекс из трёх упражнений:

(30!1/0@r5 30!2/0@r10 30!3/0@r15)*5# 1=подтягивания 2=отжимания 3=приседания

AMRAP с ручной фиксацией:

dur:1200((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)# manual=1 metro=0 title=Синди 1=подтягивания 2=отжи

23. Что делает схему «крутой»

Схема хороша не потому, что она короткая. Она хороша потому, что одна и та же запись может использоваться сразу в нескольких местах:

1. В компараторе - чтобы построить график, сравнить методы и заранее оценить сессию.
2. В заруба-таймере - чтобы реально провести тренировку по той же формальной записи.
3. В редакторе схем - чтобы быстро менять тренировку и сразу видеть, как изменится график.
4. В пресетах - чтобы сохранить готовые варианты.
5. В анализе - чтобы понимать структуру сессии, нагрузку, цели и фактическое выполнение.
6. В работе с пульсом - чтобы сравнивать теоретическую модель, реальный пульс и прошлые проходки.
7. В общении - чтобы передать другому человеку не «примерно такую тренировку», а точную исполняемую программу.

Обычный текст:

Сделай 20 минут Синди: 5 подтягиваний, 10 отжиманий, 15 приседаний, сколько кругов успеешь.

Схема:

dur:1200((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)# manual=1 metro=0 title=Синди 1=подтягивания 2=отжи

Первое понятно человеку. Второе понятно человеку, таймеру, графику, анализатору и истории тренировок.

24. Предлагаемая лестница обучения

Уровень 1. Читать простые схемы

Нужно понимать ON/OFF, #, *N, @8, @r20, @t60.

30/30#

20/10*8#

60/0@r20*10#
60/0@t60*5#

Уровень 2. Писать интервальные тренировки

Нужно понимать intensity work/rest, группы, backtick, title в metadata.

20/40@8/2*10#
(30/30 20/40)*3#
40/30*5`#

Уровень 3. Писать комплексы

Нужно понимать !N метки, metadata binding, вложенные группы, ручной режим.

(15!1/15!2@6r3/3r4 15!3/0@3)*2#title=комплекс 1=подтягивания 2=отжимания 3=прыжки manual=1

Уровень 4. Писать AMRAP и отложенные схемы

Нужно понимать ~, ~!N, dur:N(...), *~.

dur:1200((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)# manual=1 metro=0 title=Синди 1=подтягивания 2=отжимания

Уровень 5. Писать вариативные и прогрессивные сессии

Нужно понимать progression, tempo ramp, constants, random.

30/30->[for:3,delta:-5/+5]#
60/10@t[15->25]#
[low=12 high=24](150/0@t[low->high] 145/0@t[high->low] 0/10)*2#
random(30/0@9t220 30/0@6t140 30/0@7t180 turb:7 dur:580 min:8)#

25. Active, deferred, materialized и realized - коротко

Active schema

Активная схема заранее знает свою структуру.

20/10@8.5*8#

По ней можно заранее построить график, рассчитать теоретическую модель пульса и сразу вести спортсмена по готовому плану.

Deferred schema

Отложенная схема содержит неизвестные заранее значения.

~/0@r~*~#title=блокнот

Она нужна, когда спортсмен сам определяет длительность подходов, количество повторов или количество кругов во время выполнения.

Materialized schema

Когда отложенная схема впервые реально выполнена, появляются конкретные значения: подход 1 - 42 секунды, 12 повторов; подход 2 - 38 секунд, 10 повторов; подход 3 - 45 секунд, 13 повторов.

После этого прохода может быть материализована: то есть превращена в конкретную активную схему, которую уже можно сохранить и использовать как отдельный тренировочный шаблон.

сначала: свободная отложенная схема
потом: спортсмен реально её прошёл
затем: таймер сохранил фактическую структуру
итог: появилась новая активная схема по реальной проходке

Так можно не только вести дневник, но и превращать удачные реальные тренировки в повторяемые планы.

Realized schema

Realized-схема - это фактическая реализация спортсменом конкретной схемы. Она может отличаться от исходного плана: где-то спортсмен ускорился, где-то замедлился, где-то изменил темп в адаптивном атоме ^.

Для анализа это особенно ценно: можно сравнивать исходный план, materialized/realized форму и реальный пульс.

26. Схемы как язык обмена тренировками

У схем есть ещё одно важное свойство: их легко передавать.

Обычное описание тренировки может быть длинным, неоднозначным и зависеть от того, как человек его понял. Схема компактна и исполнима.

Громоздкое описание:

Сделай 10 раундов ВИИТ: 15 секунд активно, intensity примерно 8.5, держи темп 40; потом 15 секунд лёгкого восстановления, intensity 2, темп 20; после последнего раунда отдых уже не нужен.

В виде схемы:

15/15@8.5t40/2t20*10`#

Или описание СИТ:

Пять мощных спринтов по 30 секунд с intensity 9.5 и темп 30, между ними по 210 секунд восстановления, после последнего спринта финальный отдых не выполнять.

В виде схемы:

30/210@9.5t30*5`#

Это удобно не только одному спортсмену. Тренер может иметь арсенал схем для командных тренировок: разминки, интервалы, спринты, комплексы, тесты, восстановительные сессии. Такие схемы можно хранить, публиковать, отправлять ученикам и использовать повторно.

Возможны и отдельные ресурсы с разработанными схемами: библиотеки, подборки, пресеты, схемы под разные виды спорта и уровни подготовки.

Мобильный таймер дополнительно упрощает обмен: схема может быть передана через QR-код. Один человек создаёт схему, другой сканирует её и сразу получает готовый исполняемый план.

В этом смысле Схема - это не просто внутренний формат таймера. Это переносимый язык тренировок.

27. Визуальный словарь

Atom / атом

Минимальная исполняемая единица.

30/30@8

Означает: работа 30 секунд, отдых 30 секунд, intensity 8.

Repeat / повтор

Повторяет атом или группу.

30/30*5#

Group / группа

Объединяет несколько элементов в один повторяемый блок.

(30/30 20/40)*3#

Scope / локальная область

Задаёт локальный бюджет времени или повторов.

dur:1200((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)#

Tail / хвост

Метаданные после # или ;.

60/0@t60*5#title=Заруба 5 минут voice=1 metro=1

Core-схема управляет исполнением. Tail помогает интерфейсу, голосу, названию и настройкам.

Progression / прогрессия

Повторяет структуру, постепенно меняя значения.

30/30->[for:3,delta:-5/+5]#

Ramp / рампа

Меняет tempo внутри одного атома.

60/10@t[15->25]#

Deferred / отложенное значение

Значение будет определено во время выполнения.

~/0@r~*~#title=блокнот

Adaptive atom / адаптивный атом

Активный атом, в котором tempo может управляемо меняться во время выполнения.

300/0@t180^#

28. Короткая шпаргалка

30/30#	работа 30, отдых 30, закончить
30/30*5#	повторить 5 раз
30/30@8#	работа с intensity 8
30/30@8/2#	работа 8, отдых 2
60/0@r20#	60 секунд, цель 20 повторов
60/0@t60#	60 секунд, tempo 60
60/0@8r20#	intensity 8 + цель 20 повторов
60/0@8t60#	intensity 8 + tempo 60
(30/30 20/40)*3#	группа из двух атомов, повторить 3 раза
40/30*5`#	убрать финальный отдых
~/0@r~*~#	ручной блокнот
~!1/0@r5#	1=подтягивания именованная ручная длительность
dur:1200((...)*~)#	выполнять внутренний блок в пределах 20 минут
30/30->[for:3,delta:-5/+5]#	progression на 3 шага
60/10@t[15->25]#	разгон tempo внутри атома
300/0@t180^#	адаптивный активный tempo-атом

29. Что добавить в следующей версии

В следующих версиях стоит доработать:

1. ещё более красивые реальные графики сессий;
2. мини-шпаргалку на одну страницу отдельным листом;
3. визуальный словарь с крупными иллюстрациями;
4. более аккуратное оформление PDF/HTML-версии.

Пока не раздуваем документ отдельными большими разделами про внутреннее исполнение таймера, полное устройство графика, авторов пресетов и задания с ответами.