



КУРСЫ ДЛЯ ВСЕХ, ДАРОМ,
И ПУСТЬ НИКТО НЕ УЙДЁТ ОБИЖЕННЫЙ!

Андрей Миськов, 1 сентября 2020

КУРСЫ ДЛЯ ВСЕХ, ДАРОМ,
И ПУСТЬ НИКТО НЕ УЙДЁТ ОБИЖЕННЫЙ!

*Невозможно решить проблему на том же уровне,
на котором она возникла. Нужно стать выше этой
проблемы, поднявшись на следующий уровень.*

— Эйнштейн

К чему вы учите?



ShriramKrishnamurthi
@ShriramKMurthi



Replies to [@blackeuler](#)

Because there's nothing quite as lovely as watching the smile on a student's face when they first get something difficult. It goes from blank to a whisper of a smile to a full-blown explosion of joy. Gaining hard-won knowledge may be the most powerful force in the world.

Откуда мнение про курсы и образование

- сотни собеседований со студентами и начинающими;
- десятки выпускников после корпоративных курсов Оджетто;
- дюжина производственных практик;
- пробовал менторить на обучающих платформах;
- разбирался с методиками преподавания в ВУЗах США и Европы.

Проблемы провинции

Проблемы провинции: компания

- Отток опытных профессионалов.
- Новые сотрудники в основном приходят без опыта через курсы.
- Новые знания в компанию приходят медленно.
- Плохие привычки передаются из поколения в поколение.

Проблемы провинции: ВУЗ

- Средний выпускник ВУЗа не соответствует ожиданиям работодателя.
 - Зашорен из-за неактуальных методик и посредственных преподавателей.
 - Согласен делать что угодно, потому что не знает, что вообще есть на рынке.
 - Не имеет задора для решения более-менее сложных задач, потому что не поймал достаточно кайфа во время учёбы.

Возможное решение

Возможное решение

- Латать дыры в фундаментальных знаниях без универа.
- Делегировать это профессиональным преподавателям.
- Прикладную часть про инструменты оставить сотрудникам.
- Принять тот факт, что никто не знает, как правильно писать программы и тем более не знает, как правильно учить программистов. Быть готовым к экспериментам.
- Поощрять ротацию кадров по стэкам и областям деятельности.

Что такое «фундаментальные знания»?

- Алгоритмы?
- Структуры данных?
- Вычислительная сложность?
- Устройство языков программирования?
- ФП, ООП, <...>П?
- Паттерны?
- C, C++, Java, Swift, Python, Haskell, ...?

Для начала разберёмся, что делает
разработчик

Что делает разработчик?

- Получает нечёткую и, возможно, некорректную задачу.
- Проводит анализ предметной области, обсуждает и корректирует задачу.
- Превращает информацию из реального мира в данные, понятные ЭВМ.
- Формирует инструкции для ЭВМ — программу.
- Управляет сложностью, следит, чтобы программа была доступна для понимания и не превращалась запутанную лапшу.
- Обеспечивает корректный результат, тестируя разные входные данные.

Нужна методика,
обучающая системному подходу
для решения задач

Systematic Program Design

Rainfall Problem

Soloway, 1986, Yale

Есть список чисел:

[1, 2, -3, 4, -999, 0, 10]

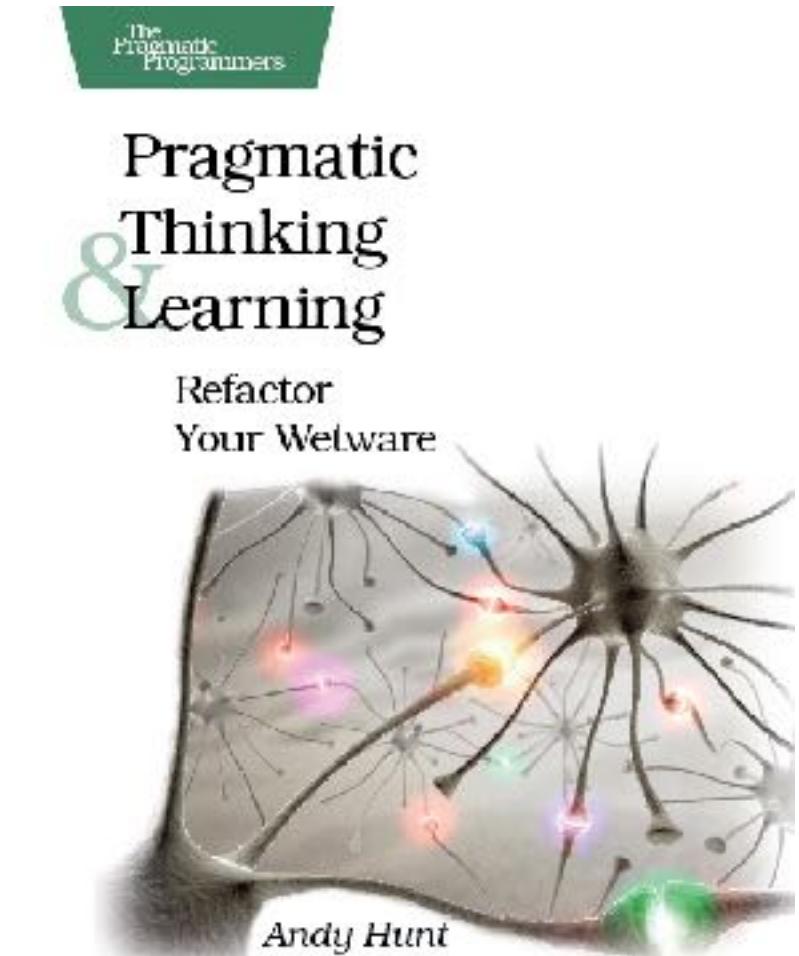
[1, 2, -3, 4, -999, 0, ~~10~~]

Посчитайте среднее значение положительных чисел, перед -999.

Обретение навыков по Дрейфусам

Novice

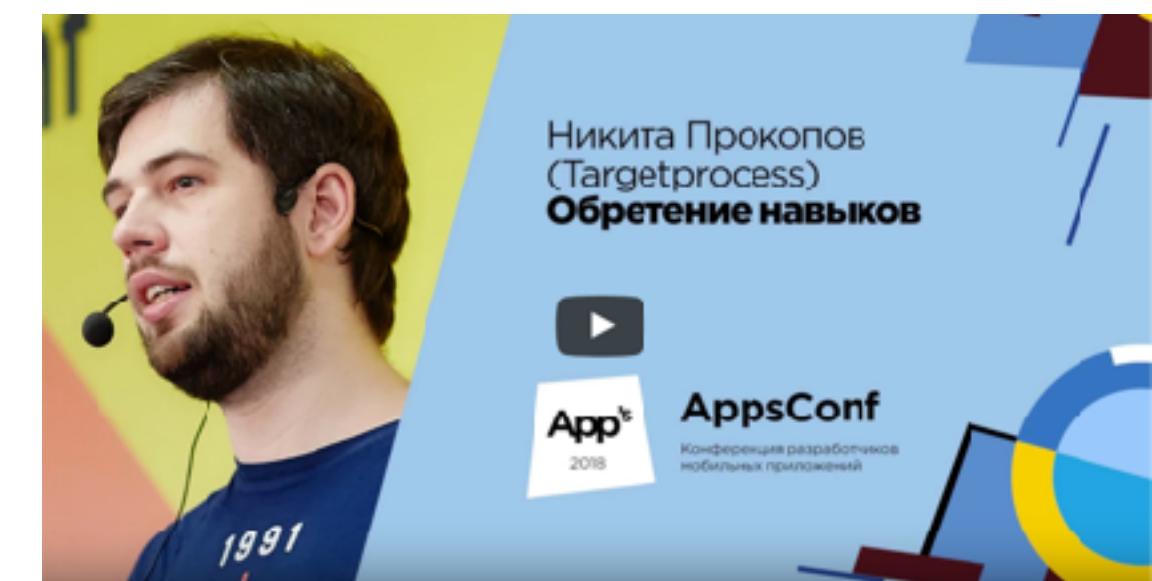
- не хочет учиться, хочет результатов
- нуждается в инструкциях (рецептах)



Advanced Beginner, Competent, Proficient

Expert

- работает интуитивно
- рамки (инструкции) только мешают



Новичкам нужны рецепты

Пример: рецепт, как написать функцию

- **Сигнатура**

Запишите название функции, что она принимает и что возвращает.

- **Заглушка**

Запишите шаблон функции без реализации.

- **Тесты**

Запишите примеры использования, чтобы проверить правильность.

- **Код**

Запишите тело функции, добавьте вспомогательные функции, если нужно.

- **Корректировка**

Исправьте ошибки, улучшите алгоритм.

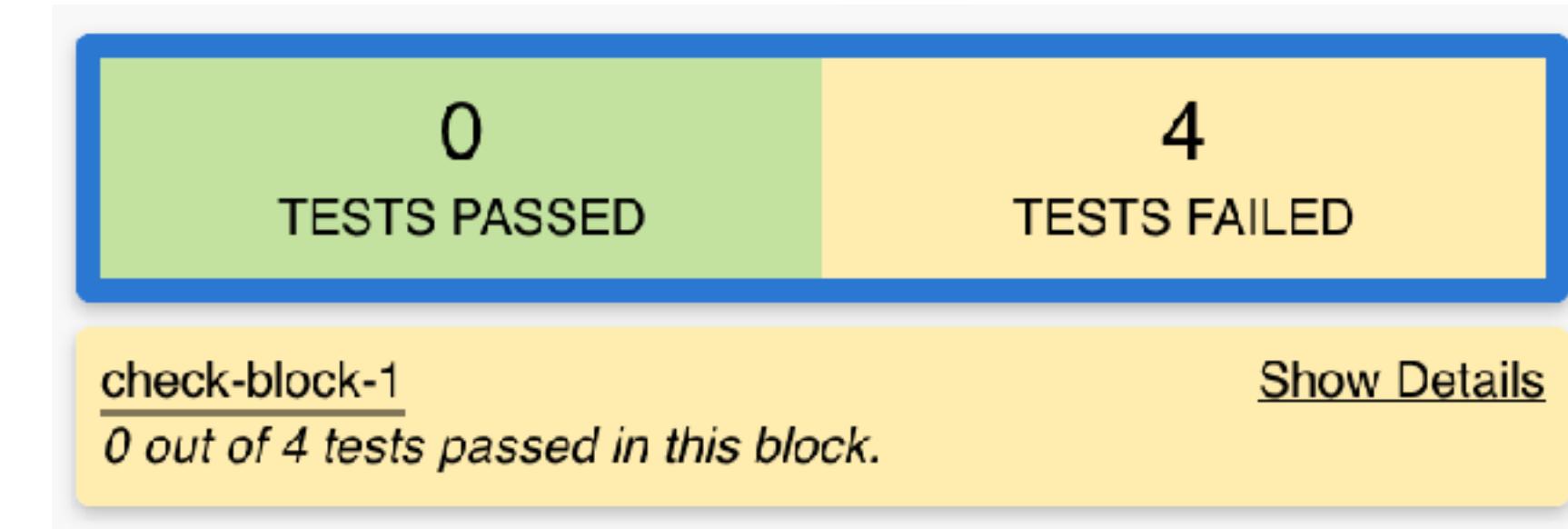
Реализация



```
# Сигнатура  
rainfall :: List → Number
```

```
# Заглушка  
fun rainfall(l):  
    ... # валидный синтаксис, говорим компилятору, что вернёмся сюда позже  
end
```

```
# Примеры, они же тесты  
check:  
    rainfall([list: 1, -2, 5, -999, 8]) is 3  
    rainfall([list: -3, -20]) is 0  
    rainfall([list: -999]) is 0  
    rainfall([list:]) is 0  
end
```



После запуска не должно быть синтаксических ошибок,
просто должны провалиться тесты.

Алгоритм и код для Rainfall Problem

```
rainfall([-1, 2, -3, 4, -999, 0, 10])
  |> отсечь до -999      [-1, 2, -3, 4]
  |> убрать отрицательные [2, 4]
  |> посчитать среднее     3
```

```
[-1, 2, -3, 4, -999, 0, 10]
  |> take-while(not= -999)
  |> filter(only n > 0)
  |> average
```

```
average([2, 4])
  |> посчитать количество
  |> вернуть ноль, если пусто
  |> иначе просуммировать
  |> и поделить сумму на количество
```

```
fun average(l):
    len = length(l)
    if (len == 0):
        0
    else:
        sum(l) / len
    end
end
```

Алгоритм и код для Rainfall Problem

;; Love letter to Clojure 🖐️

```
(defn avg [l]
  (let [c (count l)]
    (if (= 0 c)
        0
        (/ (apply + l) c))))
```

```
(defn rainfall [l]
  (->> l
       (take-while #(not= % -999))
       (filter #(> % 0))
       avg))
```

Рецепты снижают когнитивную нагрузку, помогают сделать первый шаг

I used to hate CPSC 110 and Dr. Racket, which I thought was a bullshit language and total waste of time.

Now I have a job doing software dev and I'm constantly reminding myself, "write the signature, then the tests, then the function," and can almost hear Dr. Gregor Kiczales lame youtube videos, 'trust the recursion'.

Damn you UBC, for making me learn stuff I didnt want to learn but is useful. you win this round.

Личный топ 3 вводных курсов по CS

Университет	Название	Язык	Учебник
Berkley	<u>CS061A</u> <u>Structure and Interpretation of Computer Programs</u>	<u>Python</u> , <u>Scheme</u>	<u>SICP</u> → <u>Composing Programs</u>
UBC	<u>CPSC 110</u> <u>Computation, Programs and Programming</u>	<u>Racket</u> (Student Languages)	<u>HtDP</u>
Brown	<u>CS019</u> <u>Accelerated Introduction to Computer Science</u>	<u>Pyret</u>	<u>PaPL</u>

<http://andreymiskov.ru/posts/intro-cs-bravit/> — статья с разбором каждого курса

Кому это надо?

Кому это надо?



FOO FIGHTERS

Кому это надо?



Simplicity Matters
by Rich Hickey

Кому это надо?

Если вы намеренно собираетесь стать менее значительной личностью, чем позволяют ваши способности, вы будете глубоко несчастной личностью.

— Маслоу

Кому это надо?

- **Студентам.** Изучайте то, что даст вам максимальные знания при минимальных отвлекающих факторах. CPSC110 если хотите научиться писать программы осознанно, CS61A если хотите быстрых результатов, CS019 если вам интересны языки программирования.
- **Компаниям.** Повышайте уровень сотрудников. Вариант – адаптировать CS61A.
- **Разработчикам.** Расширяйте кругозор, применяйте новые инструменты, находите новую работу. CS019 – ускоренный курс, материал двух семестров в одном.
- **Преподавателям.** Перенимайте опыт, актуализируйте учебные программы.

Как лучше учиться

- Хорошие знания дают хорошие преподаватели.
- Большинство хороших преподавателей учит по-английски.
- Больше запомнить и меньше забыть помогают инструменты.

Инструменты для учёбы

The Best Way to Remember and Organize What You Learn.

RemNote is the first spaced-repetition powered note-taking tool that lets you structure knowledge exactly in the way you think about it.

[Get Started](#)

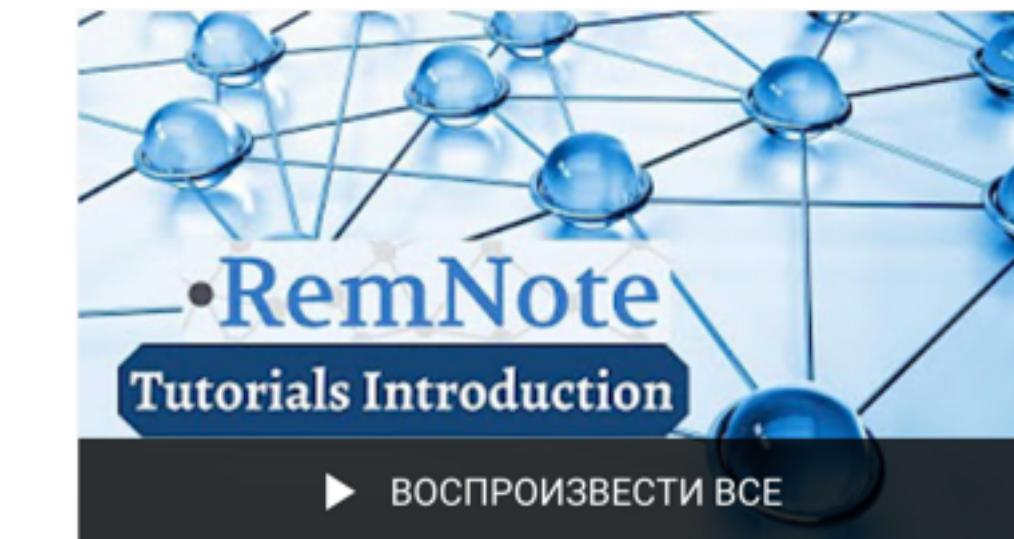
Queue 29

- Forgetting Curve
 - *model of:: Memory decay*
 - *shows :: [...]*

Show Answer

remnote.io

Переплетение знаний + Интервальное повторение



Tutorials

Альтернативы:

- [Anki](#)
- [Roam](#)
- [Obsidian](#)

Built at
MIT
Massachusetts Institute of Technology

Конец

vk.com/andreymiskov

andreymiskov.ru