

Если вы видите что-то необычное, просто сообщите мне.

## Часть 2: Обучение обучению

В первой части этой главы, мы проверили пугающий фактор Haskell. Увидели пару причин, почему люди видят Haskell вызывающим, и почему, возможно, они не должны этого делать. В этой главе, мы затронем несколько тем прошлой статьи. Изучи как изучать Haskell(и другие вещи). Изучим некоторые общие идеи обучения и обсудим как применять их к программированию.

Дальше перейдем к части 3, где вы изучите еще больше специфических техник для обучения. Мы начнем погружаться в применение этим идей к Haskell.

## Уорен Баффетт и составной интерес

Уорен Баффетт часто говорит о производительности. Он говорит, что он читает порядка 500 страниц в день, и это один из ключевых моментов его успеха. Знание, согласно Баффетту, это составной интерес. Чем больше ты получаешь и устанавливаешь связи, тем большее это собирается в единую картину и становится возможным строить на её основе.

В чем заключается апофеоз этой фразы. Я нахожу её правильно звучащей при изучении разных тем. Я увидел, как мои знания стали строиться сами по себе. До сих пор неправильное понимание этой фразы ведет людей проводя много времени реализуя этот принцип.

Простой факт, что средний человек, не имеет времени для чтения 500 страниц в день. Первое, если он читает так много, Уорен Баффетт скорее всего опытный быстрочитающий

человек, поэтому ему нужно меньше времени. Второе, он гораздо больше контролирует свое время, в отличие от большинства других людей. В моей работе разработчика ПО, я не смогу проводить полностью 80% моей работы в чтении и думании. Этим я заставляю свою команду и проект менеджера делать, что-то со мной.

В среднем люди будут видеть этот совет и решат, начать читать тонну литературы вне рабочего времени. И они даже преуспеют в чтении 500 страниц в день ... на пару дней. Ну а потом жизнь вернется в обычное русло. Они не захотят тратить своё время через несколько дней на чтение, и привычка будет отложена.

# Лучшее применение

Ну что же как достичь эффект описанный выше? Реальное непонимание, я нашел в следующем. Ключевой момент в подходе это время, но не среднее. Делая маленькие, повторяющиеся вклады, будут иметь большее вознаграждение позже. Конечно, чем больше это вложение тем больше вознаграждение тоже. Но если вложение заставляет нас бросить привычку, то это плохо.

Пользуясь этой идеей, мы можем применить её к другим темам, включая Haskell. Мы можем быть настроены посвятить час каждый день для изучения некоторые частичек идей Haskell. Но это часто не возможно. Гораздо проще посвятить 15 минут в день, или даже 10 минут в день. Это будет признаком того, что мы тратим на обучение. В любой день, может быть трудно выделить это время для чего-то. Ваше расписание, не должно позволять длиться этому долго. Но вы всегда можете найти 15 минут. Это будет гораздо проще, чем "начать в любой день", и даст большой результат.

Согласно принципу, прогресс основан на времени. Отдавая 15 минут паре различных проектов, я довольно далеко продвинулся. Мне удалось гораздо больше, чем если бы я вытаскивал час времени тут и там. Я смог начать писать статьи, так как этому посвятил 20 минут в день. И как только я провел месяц таким образом, я оказался в отличной форме.

# Джош Вайцкин и преодоление трудностей.

Еще с одной хорошей идеей обучения обучению я столкнулся в "The Art of Learning" Джош Вайцкин. Он одаренный шахматист и международный мастер. Он описал историю, которая была всем слишком знакома, так как в детстве я тоже играл в шахматы. Он видел множество молодых ребят со способностями. Они могли победить всех вокруг в школе и в шахматном кружке. Но они никогда не боролись с сильными игроками. Как результат, они заканчивали выходом из шахмат вовсе. Они столько вкладывали в идею победы в каждой игре, что сильно ущемляло гордость в моменты когда они проигрывали.

Если мы слишком сосредоточимся на нашем эго, мы испугаемся показаться слабыми. Это заставляет нас избегать конфронтации со знаниями, где мы слабы. Это и есть то, что нам нужно усилить. Если мы никогда не обращались в эту часть, мы никогда не улучшим её, и не сможем побороть большой вызов.

## Побороть HASKELL

Как это влияет на изучение Haskell, или на программирование в общем? В конце концов, программирование не соревновательная игра. И все еще есть способы которые могут повредить нашему мышлению. Наверное, стоит держаться по дальше от этой темы, так как она кажется сложной. Мы сомневаемся, что можем преуспеть в изучении. И переживаем что эта неудача раскроет нам, что мы совершенно не подходим для работы разработчиком на Haskell. Хуже если мы боимся просить других разработчиков о помощи. Что если они посмотрят на нас сверху вниз если у нас не будет хватать знаний?

У меня есть на это три ответа. Первый, я повторюсь заметкой из первой части. Тема кажется бесконечно пугающей когда вы ничего о ней не знаете. Как только вы узнаете базовые вещи, у вас есть понимание того, что вы упускаете из виду. Поймите идею как можете, запишите её простым языком. Вы можете не знать сам объект. Но он не будет для вас чем-

то неизведанным.

Второе, кого волнует, результат приложенных сил к вашему обучению? Попробуйте еще раз! Изучение темы может потребовать несколько подходов, прежде чем вы поймете её. У меня это заняло три попытки прежде чем я понял монады!

Наконец - те самые люди, перед которыми мы боимся признать нашу слабость, это те же люди, которые на самом деле могут помочь нам преодолеть эту самую слабость. Даже больше, они часто рады нам помочь! Это результат нашего первобытного страха показаться неполноценным и быть отвергнутым другими. Это сложно, но ни не возможно.

# Заключение

Поэтому помните, главное! Сфокусируйтесь на малом в начале. Не тратте на изучение больше чем 15 минут в день, возьмите проект с явным прогрессом. Сохраняйте импульс продолжая работать каждый день. Не переживайте если идея кажется вам сложной! Вполне нормально, если вам потребуется несколько попыток, чтобы что-то изучить. И самое главное, не бойтесь просить помощи.

Отличный способ сохранить импульс - это прочитать главу 3. Мы углубимся в практики и применим их!

---

Revision #5

Created 11 March 2022 05:26:32 by gasick

Updated 4 September 2022 11:04:35 by gasick