**Федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«МИРЭА – Российский технологический университет»**
**РТУ МИРЭА**
**Институт искусственного интеллекта**
**Кафедра промышленной информатики**

# Клуб программистов

## Темы заседания «Профессия программист»

**Организатор Каширская Елизавета Натановна** (к.т.н., доцент, ФГБОУ ВО "МИРЭА - Российский технологический университет") e-mail: liza.kashirskaya@gmail.com

**Заседание 1 (вводная часть)**

# СЛОЖНОСТИ ПРОГРАММИРОВАНИЯ

У людей, занятых творческим или интеллектуальным трудом, существуют общие проблемы. Программирование — сложный предмет, и обычно за него берутся способные, амбициозные и склонные к перфекционизму люди. На начальном этапе у них не будет хорошо получаться. Привыкшие к высокой планке, они будут расстраиваться. Внутренний голос будет постоянно нашептывать: «У тебя никогда не получится, лучше оставь это дело».

В такие моменты думайте о том, что ваша самокритичность — это признак вашей экстраординарности, и верьте, что преодолеете этот «неумелый период».

Самое сложное в программировании – это преодоление разницы между высокими стандартами и своими низкими умениями.

# Преимущества программирования

1. Проблемы в коде в корне отличаются от проблем в физическом мире. Починить неисправный код можно одной лишь силой ума, в отличии от, например, сломанной машины, которая требует покупки дорогих запчастей.

2. Профессионально расти можно только **на границе зоны комфорта.** Занимаясь незнакомыми вещами, вы сделаете кучу ошибок, но зато и получите настоящие знания.

3. Ошибки в программировании — это не закрывающиеся двери перед вами, а ключи к обучению. Примите факт, что компьютер всегда прав, а вы — нет.

4. Если что-то пошло не так, не надо винить компьютер или программу. Не выясняйте с ними отношения. Просто задайтесь целью: «как это исправить». Если вы хотите выяснять отношения с языками программирования, почему они такие глючные и ваша программа дает сбой — то вы выбрали не ту специальность.

Готовьтесь к худшим сценариям

Ждите от пользователей программ самых неожиданных вещей. Они будут вводить цифры, где им не место, вставлять абзацы текста в поле для имени и делать кучу других несуразных вещей. Не создавайте формы, где можно указать возраст человека в тысячи лет. Будьте готовы ко всему, не доверяйтесь пользователям, предугадывайте худшие сценарии и стройте защиту от них.

## Контроль за эмоциями

Программирование, зачастую, это долгий, трудный и расстраивающий опыт. Бывает, месяцами изучаешь какую-то тему, потом много дней пишешь сложный запутанный код, который, наконец-то, делает то, что тебе нужно. А потом опытный программист берет и переписывает его за 3 минуты в 5 строчек. И ты чувствуешь себя раздавленным. Что бы ни случилось, не надо расстраиваться.

## Самостоятельность

Многие новички легко проходят разные курсы по программированию, но стоит им взяться за самостоятельную задачу, они впадают в транс. Или нет идей для написания, или есть идеи, но нет понимания как их реализовывать, с чего начать. Всё дело в том, что курсы дают вам синтаксическую грамотность, вы вроде бы помните разные команды, но не можете написать свою программу. И вам начинает казаться, что учебный курс был разводкой для лохов, где вас научили поверхностной ерунде, а саму суть оставили в секрете. И эта суть — это навык программно мыслить.

# Недостатки программирования

Логика программирования заключается в том, чтобы разбить большую задачу на маленькие подзадачи и последовательно реализовать их, а потом связать воедино. Программист — не тот кто на перегонки печатает текст кода со знанием всех команд, а тот, кто мыслит в логике программы. И когда у вас, наконец, получается сделать что-то самому, самостоятельно, этот момент невероятно вдохновляет и вы вспоминаете свою грандиозную идею, которая недавно казалась невыполнимой и думаете: «Теперь я смогу реализовать её!» Хотя, конечно, вам еще расти и расти до её реализации, но момент всё равно приятный.

# Недостатки программирования

Незнание, с чего начать

Вы не знаете, какой язык начать изучать: C, Python, Java, PHP, C++ и еще миллион других языков.

Вы не знаете, где изучать: по книге, онлайн материалам, или записаться на курсы.

Вы не знаете, что изучать: мобильные приложения, Android, iOS, веб, операционные системы, искусственный интеллект, машинное обучение.

Вы не знаете как учиться: читать книги, чужой код, взять кого-нибудь для совместного обучения, программировать соревновательно, на пару, устроиться стажером?

# Недостатки программирования

Очень много этих вопросов: «какой, где, что, как, …?».

Вы советуетесь с друзьями, слушаете профессионалов, спрашиваете на форумах. Но их ответы ещё больше запутывают вас.

Проблема при изучении программирования в том, что по теме слишком много информации. И вам надо научиться пробираться через дебри этого шума. Выбирать только то, что необходимо, очень непросто, но от этого навыка зависит ваше будущее.

Будьте самообучаемыми. Программная разработка - одна из самых динамично развивающихся отраслей во всём мире. Если фундаментальные принципы меняются редко, то инструменты — чуть ли не каждый день. Важно следить за всеми новинками и уметь самостоятельно осваивать необходимые для вас.

Учитесь общаться и сотрудничать. Если вы умеете что-то хорошо делать, то для вашей компании вы полезная единица (= 1). Но если при этом вы поддерживаете и вдохновляете ещё 10 человек, то вы превращаетесь в глазах руководства в одиннадцать (= 11).

## Невозможно всё знать

Каждый раз, когда ты в идеале овладел какими-то навыками, ты узнаешь, что появилось что-то новое, намного лучше. И возникает парадокс Сократа: «Я знаю, что ничего не знаю». Постоянно нужно тратить много времени на изучение нового, а так как невозможно знать всё и быть специалистом во всём, то постоянно надо выбирать приоритеты — что для тебя в приоритете в данный момент, какая технология, какой подход.

# Эффект Даннинга-Крюгера

Есть такое знаменитое правило Даннинга-Крюгера, суть которого заключается в том, что люди, имеющие низкий уровень квалификации, делают ошибочные выводы, принимают неудачные решения и при этом неспособны осознавать свои ошибки в силу низкого уровня своей квалификации.

**Эффект Даннинга-Крюгера - это когда дураки не могут понять, что они дураки. Потому что они дураки.**

# Эффект Даннинга-Крюгера

Для справки, из википедии.

Эффект Даннинга — Крюгера заключается в том, что люди, имеющие низкий уровень квалификации, делают ошибочные выводы, принимают неудачные решения и при этом неспособны осознавать свои ошибки в силу низкого уровня своей квалификации. Это приводит к возникновению у них завышенных представлений о собственных способностях, в то время как действительно высококвалифицированные люди, наоборот, склонны занижать оценку своих способностей и страдать недостаточной уверенностью в своих силах, считая других более компетентными. Таким образом, менее компетентные люди в целом имеют более высокое мнение о собственных способностях, чем это свойственно людям компетентным, которые к тому же склонны предполагать, что окружающие оценивают их способности так же низко, как и они сами.

# Эффект Даннинга-Крюгера

Почему невежды считают себя экспертами?

# Эффект Даннинга-Крюгера

# Эффект Даннинга-Крюгера

Довольно показательным примером эффекта Даннинга-Крюгера был Наполеон. Он был очень хорошим полководцем, однако не безупречным. Его ошибка заключалась в том, что он не дооценил способности своего соперника, то есть войск России, в силу своей завышенной самооценки и большого количества предыдущих побед.

В следствии этого он потерпел сокрушительное поражение в войне. К тому же, узурпатор так и не признал свою вину в проигрыше, перекладывая ответственность на случайную череду событий.

Следует отметить, что далеко не только эффект ДК послужил триггером в фиаско Бонапарта. В процессе наступления на Россию, он проживал стадии принятия расставания в отношениях с Жозефиной, что также смутило его рассудок и помешало взвешенному принятию решений. Но, все-таки, основной причиной была именно

# Эффект Даннинга-Крюгера

Тем, кто заинтересовался этой темой и хочет проверить себя на наличие эффекта ДК, даю ссылку:

**https://prostudio.ru/journal/dunning-kruger-effect/#top**

# Пример

**Же не манж па сис жур…**

Задача про автоматы, которые едят.

Эксперимент состоял в том, что на черных полях шахматной доски располагались автоматы, которые «умели» распознавать количество «пищи», а на белых – эта самая пища в разных условных количествах. Автомат из максимум четырех доступных полей был способен выбрать поле с наибольшим количеством условных единиц «корма». Введение автомату «второго уровня рефлексии» позволяло ему просчитать на 1 ход поведение других автоматов и выбрать оптимальный вариант – например, поле с 3 единицами «корма», к которому имел доступ он один, а не с 5-ю, куда могли попасть еще три конкурента. Уровень рефлексии у автоматов в эксперименте повышался до способности «просчитывать» поведение конкурентов» на 8 - 9 ходов.

# Пример

Очень забавный результат получился в эксперименте, когда на «шахматной доске» присутствовали автоматы с разным уровнем рефлексии. Лучше всех питались «середнячки» с 4 – 6-м уровнем, дальше шли «тупые» - с 1 – 2-м, а последними оказались интеллектуалы с 8 – 9-м, которые считали окружающих умнее, чем они были на самом деле, и каждый раз выбирали гарантированный минимум… Ничего не напоминает?

# Самое сложное в программировании

Вы можете выбрать какую-то одну вещь и стать специалистом в ней, но тогда вы очень рискуете, что в это же время появилось что-то новое, многократно превосходящее вашу технологию и это нечто завоюет рынок, в то время как вы будете продолжать держаться за старьё обеими руками. Поэтому, если вы любите учиться и постоянно узнавать что-то новое, то выбрав программирование, не будете разочарованы ни на секунду. В реальной жизни не всё так идеально, как в учёбе.

# Самое сложное в программировании

Во время учёбы вы играете с лёгкими программами из нескольких сотен, максимум — тысячи строк кода. Даже в университете, на факультете компьютерных наук.

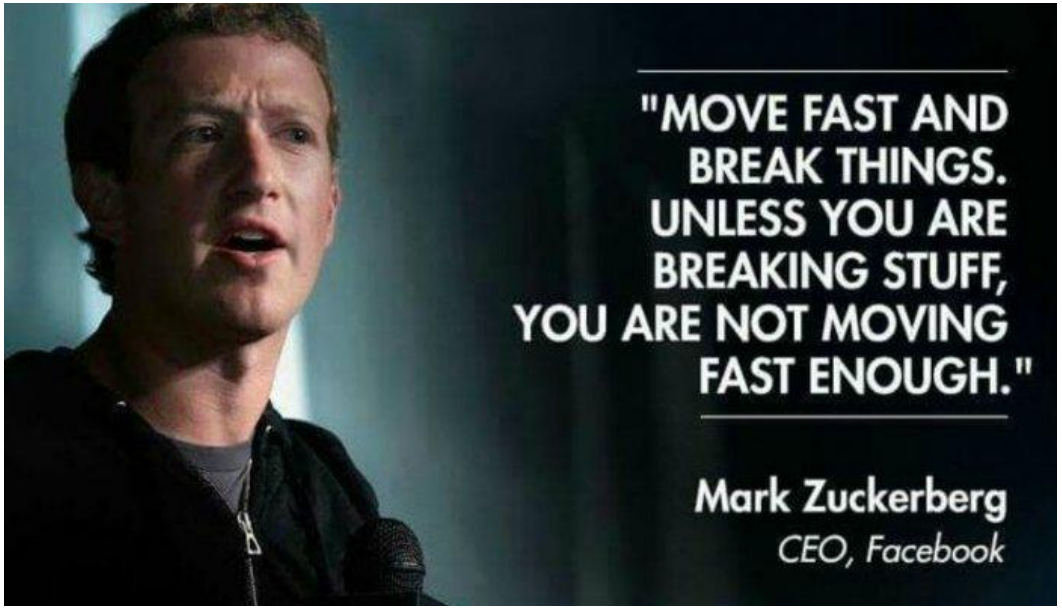
Когда вы приходите на предприятие, вы можете столкнуться с базой кода в сотни тысяч строк и даже миллионы. Там много ошибок, нелепые названия переменных, мудреные подпрограммы без документации, используются разные проектировочные шаблоны, многоуровневое кэширование и т.д.

Когда всё это надо понять и изучить за сжатые сроки — вы получаете самую вертикальную кривую обучения, с которой сталкиваются многие программные инженеры.

## ПРОЦЕСС ПОЛУЧЕНИЯ ЗНАНИЙ

Идти быстро и ломать вещи

Нужно развивать в себе особый **склад характера**, когда вы **не боитесь идти вперед**, не будучи заранее готовыми к этому. Старый девиз Фейсбука: «Идите быстро, ломайте вещи. Если вы ничего не ломаете, значит движетесь медленно».



## Балансирование между теорией и практикой

С одной стороны, можно много изучать теорию, годами читать что-то и думать, что ты мало знаешь, и ничего не делать. Это надоедает и перестаёт приносить пользу в какой-то момент. С другой стороны, можно начать делать что-то, без знания теории, и быстро застрять или заблудиться в своём коде и его ошибках. Многие начинают делать что-то, опираясь только на обрывистые ответы с форумов, не понимая целой картины своего приложения и того, куда их приведет работа в конечном итоге (например, к неподдерживаемому, необновляемому коду).

Так вот, очень важно прочувствовать этот баланс минимальной теории и последующей за ней практики. Тогда и то, что вы пишете, будет грамотным, и теория будет усваиваться в разы быстрее и интереснее, и вы будете гармонично обучаться во время работы.

## Борьба с багами

Баги (жуки) - это ошибки в программе. Если продолжить метафору с жуками и человеком, то для новичков это скорее что-то подкожное, зудящее, вызывающее ужас, потому что невидимо и трудно устранимо.

Самое обидное, что они появляются, когда вы вроде бы всё сделали правильно, и можно приступать к дальнейшим свершениям. Но вдруг программа перестает работать без видимых причин или работает не так, как задумано. И приходится всё бросить и тратить несколько часов, а то и дней на поиск этой ошибки. Кажется, будто это время тратится впустую (ведь вы не занимаетесь созданием «нового», а ковыряетесь в «старом»). Чтобы пережить этот период, нужно титаническое терпение.

Вы должны понимать, что, на самом деле, за это время вы узнаёте очень много нового, и делаете это с большей мотивацией и степенью запоминаемости, чем в спокойных условиях изучения теории. Исправление каждого бага — это, в первую очередь, устранение своего невежества во многих вопросах, о существовании которых вы раньше и не задумывались. **Происходит переход от неосознанного незнания к осознанному незнанию и перевод незнания в знание.** Со временем количество допущенных вами багов неизменно будет уменьшаться и наловчитесь работать с инструментами по их устранению.