SOFTWARE ENGINEERING

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики
Кафедра технологий программирования

Software Engineering

Минск 2015

Краткое содержание

Подробное содержание

Глава 1. IT-проект

IT-проект и его структура

Структура проекта

Проект – это временное действие, которое выполняется для создания уникального продукта или услуги. Временное обозначает, что каждый проект имеет свои определенные начало и конец. Уникальный обозначает, что продукт или услуга принципиально отличается от других аналогичных продуктов или услуг, так как принципиально отличаются условия создания этих продукта или услуги в каждом проекте.

Участниками проекта являются:

- собственник, заказчик, инвестор;
- менеджеры проекта;
- исполнители работ проекта;
- «окружающая» организация;
- службы контроля (технического, финансового и пр.)
- финансирующие организации, банки.

Несмотря на все многообразие существующих проектов, в команде можно выделить ряд более или менее стандартных *ролей*.

В первую очередь, это *менеджер (руководитель) проекта* — физическое лицо, несущее личную ответственность за успех проекта и осуществляющее оперативное руководство.

Как правило, в компаниях назначают *куратора проекта* — представителя высшего руководства, который хоть и не вникает в тонкости текущего положения дел в проекте, но контролирует его ход, следит, чтобы проект соответствовал стратегическим целям компании, а если у *менеджера проекта* не хватает полномочий, — помогает ему своим авторитетом.

Проектный комитет создается в компаниях, в которых бизнес построен по проектному типу. Это орган, задачи которого — отбирать проекты и контролировать их выполнение на высшем уровне, принимать ключевые решения.

В технически сложных проектах важна роль *главного инженера проекта (ГИП, командный лидер)*, который порой по статусу равен менеджеру проекта.

В крупных проектах могут выделяться *менеджеры по различным функциональным областям*, например, по управлению финансами, персоналом, рисками и т. п.

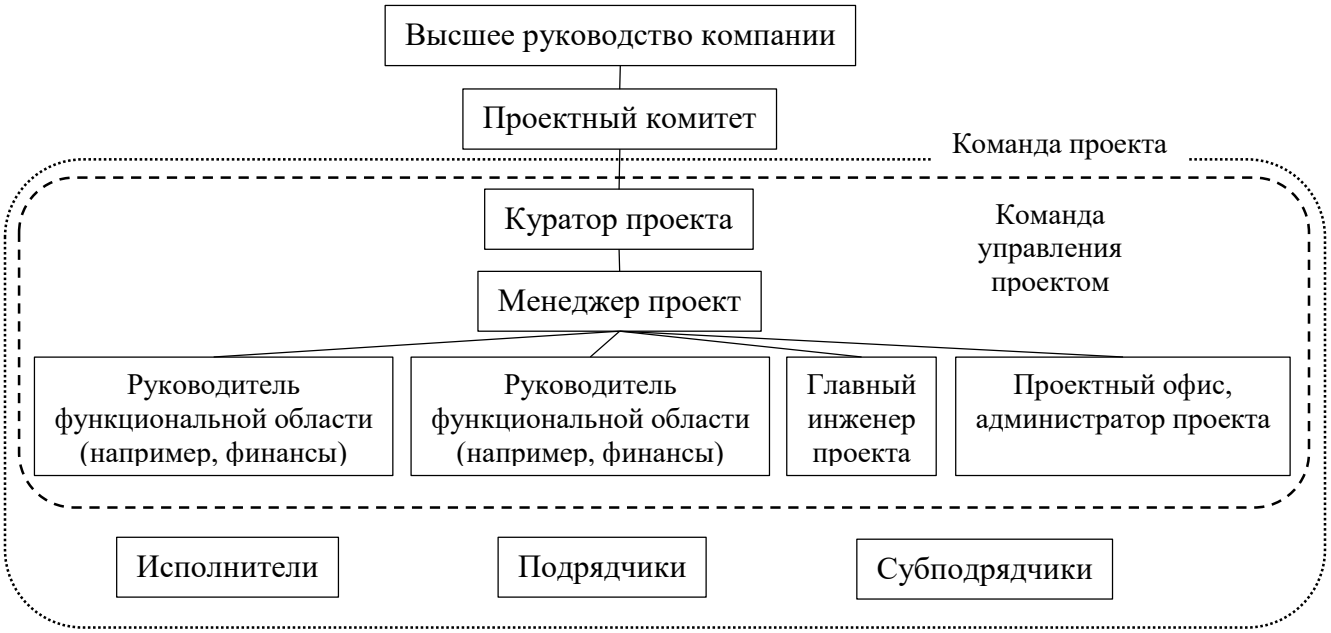
Все вышеперечисленные роли образуют *команду управления проектом*, которая входит в *команду проекта*. Также участниками команды проекта являются *исполнители* как из числа штатных сотрудников компании, так и нанятые специально для реализации конкретного проекта. Иногда в нее включают *подрядчиков* и *субподрядчиков*.

Отдельно стоит выделить *проектный офис*. В простейшем случае это своего рода секретариат, в котором хранится вся документация по проекту. Он может состоять как из одного, так и из

нескольких сотрудников. В более продвинутых компаниях *проектный офис* также играет роль методологического центра, обслуживающего все проекты организации.

Основная команда формируется исходя из потребностей проекта. Должны ли сотрудники:

- работать полный или неполный рабочий день?
- отчитываться менеджеру проекта?
- тратить на проект 100% своего рабочего времени?



Эффективная деятельность

Показатели *эффективной деятельности* команды:

- Ясное понимание цели проекта и нацеленность на конечный результат;
- Четкое распределение функций и ответственности;
- Наличие плана развития команды;
- Командная солидарность;
- Взаимопонимание и бесконфликтность;
- Посещаемость рабочих совещаний и активное участие в решении проблем.

Для *обеспечения эффективного руководства* командой проектный менеджер должен:

- Определить организационную структуру команды, подобрать ее состав, распределить функции и обязанности;
- Назначить руководителей и ответственных за отдельные направления;
- Своевременно спланировать, распределить и скоординировать работу;
- Четко объяснить цели и задачи;
- Преодолевать препятствия и избегать конфликтов;

- Обеспечить трудовую активность команды силой личного авторитета, заинтересовать каждого члена команды, оказывать им помощь и проявлять участие, поддержать перспективу команды;
- Привлекать всех к решению задач;
- Обеспечить поддержку проекта со стороны руководства и регулирование отношений с окружением команды, создавать привлекательный имидж команды.

Менеджер проекта – лицо, отвечающее за успех проекта, а также за подбор и работу своей команды и завершение проекта. Это происходит в рамках ограничений, наложенных партнерскими и другими организациями, внешними по отношению к проектной команде.

Менеджер проекта должен быть назначен как можно раньше. Обычно менеджера проекта назначает *Заказчик*.

Искусство управления человеческими ресурсами и координация их в проекте реализуется менеджером посредством применения административных и поведенческих знаний для достижения определенных проектных целей в содержании, затратах, времени, качестве и удовлетворении участников проекта.

Менеджер проекта является ключевой фигурой в команде проекта. От его лидерских качеств, организационных способностей, харизмы, умения вовремя принимать решения и сглаживать конфликты напрямую зависит успех проекта.

Определение термина IT-проект

Термин "**ИТ-проект**" обычно используется для обозначения деятельности, связанной с использованием или созданием некоторой информационной технологии. Это приводит к тому, что ИТ-проекты охватывают очень разнообразные сферы деятельности: разработку программных приложений, создание информационных систем, развертывание ИТ-инфраструктуры и пр.

С одной стороны, эти работы соответствуют классическому определению проекта "Проект – это комплекс усилий, предпринимаемых с целью получения конкретных уникальных результатов в рамках отведенного времени и в пределах утвержденного бюджета, который выделяется на оплату ресурсов, используемых или потребляемых в ходе проекта". С другой стороны, они обладают известными отличительными особенностями:

- разделение на уровне идеологии заказчика и исполнителя: заказчиком, как правило, является бизнес, а исполнителем – ИТ-специалисты, и есть трудности в выявлении требований, ожиданий от проекта, в формировании технического задания. Существует также проблема эффективных коммуникаций;
- ответственность за результат проекта имеет "солидарный" характер. То есть здесь нельзя возложить ответственность за успех проекта только на исполнителя, точно так же, как нельзя говорить, что исключительно заказчик виновен в том, что проект не удался. В ИТ-проекте должны создаваться определенные условия для взаимодействия сторон, и стороны, участвующие в нем, несут равную ответственность за результаты проекта;
- зачастую реализация ИТ-проекта предусматривает изменение существующих организационных структур на предприятии;

- обычно в ИТ-проект вовлечено множество подразделений организации;
- существует высокая вероятность конфликтов между руководителем проекта, высшим руководством, руководителями подразделений и персоналом организации;
- многие ИТ-проекты имеют колоссальные бюджеты. В крупных компаниях масштабы проектной деятельности в области информационных технологий (ИТ) измеряются миллионами долларов, причем реализация новых проектов происходит постоянно. Если, например, промышленное предприятие достаточно один раз построить – и оно будет работать, не требуя регулярных инвестиций, то развитие ИТ-инфраструктуры в растущих компаниях требует больших и регулярных вложений. Большие бюджеты, в свою очередь, подразумевают больший уровень ответственности и, соответственно, больший уровень компетенции тех людей, которые этими проектами управляют.

Планирование бюджета

Планирование бюджета на IT-проекту можно разделить на три этапа:

I. Сбор информации

Для составления (и обоснования) бюджета на ИТ нужно знать:

1. Что компания уже купила в прошлом и за что продолжает платить в настоящем, а именно:
    - используемое и находящееся в резерве оборудование,
    - лицензии на программное обеспечение,
    - сервисные контракты,
    - операторские услуги,
    - расходные материалы и затраты на них.
2. Какие информационные технологии используются в компании и для чего. Полный список услуг (ИТ-сервисов), которые использует бизнес в своей работе, начиная от банальных «электронная почта», «печать документов», «телефонная связь», заканчивая системами управления, безопасности и специфическими бизнес-приложениями.
3. Какие цели, планы и задачи у компании на ближайший финансовый период, какие проблемы должны быть решены? Реорганизация подразделений, увеличение персонала, появление новых задач, изменение требований по производительности, надежности и безопасности работы информационных систем. На данном этапе полезно пообщаться как с руководством компании, так и с руководителями структурных подразделений, чтобы понять их потребности по изменению качества обслуживания, надежности, удобства работы с системами, а также уточнить список используемых ими ИТ-сервисов. Дополнительно можно провести анкетирование пользователей, собрать полученные за последнее время обращения.

II. Анализ

Задача данного этапа – найти те точки, которые мешают в текущий момент достичь поставленных целей или могут помешать в будущем. Для этого рекомендуется проанализировать следующие показатели:

1. **Производительность**

   Хватает ли производительности систем в настоящий момент всем пользователям? Соответствует ли текущий уровень производительности систем ожиданиям бизнеса? Что сейчас является слабым звеном? Хватит ли производительности, если нагрузка вырастет в соответствии с планами по развитию?

2. **Надежность**

   Достаточные ли меры приняты для обеспечения сохранности данных? Допустимо ли менять оборудование по мере выхода из строя или стоит его обновить заранее? Сможете ли вы в случае сбоя восстановить работоспособность в требуемые сроки или требуется заранее приобрести дополнительное оборудование или программное обеспечение для этого? Есть ли какие-то известные проблемы, которые влияют на безотказность работы ИТ-систем?

3. **Функциональность**

   Решают ли существующие приложения задачи пользователей и бизнеса? Решают ли они их эффективно? Что компания недополучает сейчас? Что нужно изменить, чтобы соответствовать будущим требованиям? Актуальны ли существующие бизнес-приложения вообще?

4. **Безопасность**

   Защищены ли данные компании от внешних угроз? А от внутренних? Соответствует ли система защиты уровню угроз? Как изменятся требования к информационной безопасности в обозримом будущем?

5. **Удобство**

   Создает ли что-нибудь дискомфорт в работе пользователей с компьютерной техникой? Удобно ли расположены принтеры в офисе, сильно ли шумят компьютеры, все ли интерфейсы и системы понятны для пользователей, жалуются ли они еще на что-то? Можно ли это улучшить?

6. **Операционные расходы**

   Оптимальны ли существующие операционные расходы? Соответствуют ли они рыночной стоимости? В какие суммы ежемесячно обходится компании тот или иной сервис? Из чего состоят эти расходы? Можно ли их уменьшить без ущерба для компании?

7. **Запасы**

   Есть ли необходимые расходные материалы? Сколько нужно будет дополнительного оборудования и лицензий в случае расширения? Нужны ли будут дополнительные разовые или постоянные услуги в случае планируемого развития бизнеса?

III. Формирование бюджета и обоснование

Собственно, вся основная работа выполнена на прошлых этапах. Последняя задача — представить полученные выводы руководству в понятном и удобном для принятия решений виде. Обычно разделяют все расходы по следующим категориям:

1. Операционные расходы на поддержание деятельности: расходные материалы, сервисные контракты, услуги, оплата труда специалистов.

2. Необходимые капитальные вложения, в случае отсутствия которых возможны серьезные потери для бизнеса. Сюда относятся расходы, которых компании не избежать и вопрос лишь в том, будет она инвестировать деньги в это заранее, или, когда уже понесет обозначенные потери.
3. Рекомендуемые инвестиции – в сочетании с необходимыми позволяют значительно повысить показатели работы, а также устранить риски, которые могут оказать негативное влияние на бизнес.
4. Расходы, связанные с развитием – необходимый объем инвестиций для обеспечения работы систем и поддержания качественных показателей в случае реализации планов по росту бизнеса.
5. Возможные инвестиции, позволяющие улучшить функциональные возможности и/или удобство работы сотрудников с ИТ-системами. Данный пункт является красной тряпкой для финансистов, позволяя им при согласовании бюджета отказать вам в этой части и тем самым с честью выполнить свой долг, не опасаясь за последствия.
6. Обоснование каждой из статей бюджета. Это самый важный подпункт. Бизнес, к сожалению, не оперирует понятиями «шестилетний сервер» и ничего не понимает в технике — он оперирует только категориями потребностей в ИТ-сервисах, возможностями, рисками и их стоимостью для бизнеса. Каталог услуг (ИТ-сервисов), который вы делали в самом начале, нужен вам для того, чтобы общаться с руководством компании на одном языке – это ваша точка взаимопонимания. Выявив потребность в оборудовании, программном обеспечении, персонале и пр. обосновывайте необходимость в них конкретными показателями работы конечных ИТ-сервисов (риски, качество, скорость реакции и пр.), которые получает или получит бизнес.

Реализация IT-проекта

В **реализации ИТ-проектов** следует обратить внимание на следующие особенности:

- зачастую в компании заказчика одновременно выполняются несколько ИТ-проектов;
- приоритеты выполнения проектов постоянно корректируются;
- по мере реализации проектов выполняется уточнение и корректировка требований и содержания проектов;
- велико влияние человеческого фактора: сроки и качество выполнения проекта в основном зависят от непосредственных исполнителей и коммуникации между ними;
- каждый исполнитель может принимать участие в нескольких проектах;
- налицо трудности планирования творческой деятельности, отсутствуют единые нормативы и стандарты;
- сохраняется повышенный уровень риска, вплоть до непредсказуемости результатов;
- происходит постоянное совершенствование технологии выполнения работ.

Прибыль в Open Source проектах

Зачастую в ИТ проектах заказчик оплачивает всю его стоимость, однако следует отметить, что в **Open Source** проектах *финансирование* может происходить и другими способами:

- Продажа профессиональных услуг

  Финансовая отдача от затрат на программное обеспечение с открытым исходным кодом может исходить от продажи услуг, таких как обучение, техническая поддержка, или консультации, а не самого программного обеспечения.

  Другая возможность предлагает open source программное обеспечение только в виде исходного кода, при этом предоставляя исполняемые бинарные файлы только платным клиентам, предлагая коммерческие услуги по компиляции и созданию инсталляционных пакетов программного обеспечения. Кроме того, предоставление open source как коммерческий товар на физическом носителе (например, DVD).

  Успешные Open source компании, использующие эту бизнес-модель: RedHat и IBM; более специализированным примером является Revolution Analytics.

- Продажа фирменных товаров

  Некоторые FOSS организации, например, Mozilla Foundation или Wikimedia Foundation, пытаются продавать фирменные товары: футболки, кофейные кружки. Это может рассматриваться в качестве дополнительной услуги для сообщества пользователей.

- Продажа программного обеспечения как услуги

  Платная подписка на онлайн-аккаунты и доступ к серверу для клиентов является способом получения прибыли на базе программного обеспечения с открытым исходным кодом. Кроме того, комбинация настольных ПК с сервисом, называется программное обеспечение плюс услуги. Предоставление услуг облачных вычислений и программного обеспечения как услуги (SaaS) без предоставления самого программного обеспечения с открытым исходным кодом, ни в двоичной ни в исходной форме соответствует большинстве лицензий с открытым исходным кодом (за исключением AGPL).

- Партнерство с финансирующими организациями

  Прочие финансовые ситуации включают партнерские отношения с другими компаниями. Правительства, университеты, компании или другие неправительственные организации могут разрабатывать у себя или нанять подрядчика для внутренних пользовательских модификаций программного обеспечения, а затем выпустить этот код под открытой open source лицензией. Некоторые организации поддерживают разработку программного обеспечения с открытым исходным кодом грантами или стипендиями, например, Google Summer of Code initiative основанную в 2005.

- Добровольные пожертвования

  Появление систем Интернет микроплатежей в 2000-х годах таких, как PayPal, Flattr и Bitcoin помогает этому.

- Денежное вознаграждение за выполнение задачи (bounty)

Пользователи конкретного программного обеспечения могут объединиться вместе и собрать деньги для open source проекта для разработки желаемого функционала.

- Предварительный заказ / Crowdfunding / модель обратная bounty

  Новая возможность финансирования проектов СПО – Crowdfunding, модель похожа на пред-заказ, а также на перевернутую модель bounty. Как правило, организуется на базе веб-платформ, таких как Kickstarter, Indiegogo, Catincan или Bountysource. Пример успешного финансирования: австралийской программист Timothy Arceri, который предложил за $2500 реализовать в течение двух недель расширение OpenGL 4.3 для библиотеки Mesa.

- Программное обеспечение, содержащее рекламу

  С целью коммерциализации FOSS, многие компании (в том числе Google, Mozilla и Canonical) перешли к экономической модели заработка на рекламе в программном обеспечении.

  Например, приложение с открытым исходным кодом AdBlock Plus получает деньги от Google за расширение белого списка разрешенных рекламных блоков в обход блокировщика рекламы в браузере.

- Продажа дополнительных проприетарных расширений

  Некоторые компании продают собственные дополнительные расширения, модули, плагины к open source программному продукту. Это может соответствовать свободным лицензиям, если сделано технически достаточно тщательно.

- Продажа необходимых проприетарных частей программного продукта

  Вариант подхода выше, заключается в хранении нужного контента (например, аудио, видео для игр, графику или другие художественные активы) в закрытом программном продукте, выпуская сам исходный код под открытой лицензией. Хотя такой подход вполне совместим с большинством open source лицензий, клиенты должны купить контент, чтобы иметь полную и работающую версию программного продукта.

  Похожий на этот прием привязывания open source программного продукта к проприетарной аппаратной части называется «*тивоизация*» и проходит с большинством open source лицензий за исключением GPLv3, которая прямо запрещает подобное использование.

- Пере-лицензирование под проприетарной лицензией

  Если программный продукт использует только собственное программное обеспечение и программное обеспечение с открытым исходным кодом под разрешительной свободной лицензией, то компания может повторно лицензировать конечный программный продукт под проприетарной лицензией и продать продукт без исходного кода и софтверных свобод. Например, Apple Inc. является активным эксплуататором этого подхода, используя исходный код и программное обеспечение из различных open source проектов, например,

ядро операционной системы BSD Unix под лицензией BSD было использовано в компьютерах Mac, которые продаются как патентованных продукты.

- Обфускация исходного кода

  Подход состоящий в запутывании исходного кода, для коммерциализации с некоторыми открытыми лицензиями, защищая при этом важные коммерческие тайны, интеллектуальную собственность и технические ноу-хау. Этот подход был использован в ряде случаев, например, Nvidia в своих драйверах для графических карт.

- Задержка с выпуском open-source software

  Некоторые компании предоставляют самую свежую версию только для платных клиентов. Далее вендор делает ответвление от программного проекта без авторского права, добавляет к нему дополнения с закрытым кодом и продает конечный программный продукт. После некоторого периода времени патчи интегрируются обратно в приложение под той же лицензией, что и остальной часть кода. Эта бизнес-модель называется версия с отставанием.

  Экстремальный вариант такой модели является бизнес-практика, которую популяризировали Id Software и 3D Realms, которые выпустили несколько своих программных продуктов под свободной лицензией после долгого коммерческого периода, в течение которого произошел возврат инвестиций.

- FOSS и экономика

  По правовым исследованиям предпринимательства в Гарвардской школе права, свободное программное обеспечение является наиболее видимой частью новой экономики на основе общего равного производства информации, знаний и культуры. В качестве примеров они приводит ряд проектов FOSS, которые включают как free, так и open source ПО.

  Эта новая экономика уже на стадии разработки. С целью коммерциализации FOSS, многие компании, Google является самым успешным, движутся в направлении экономической модели программного обеспечения, содержащего рекламу (AdWare).

Аналоги среди IT-проектами

Анализ статистики показывает, что примерно 90 процентов ИТ-проектов аналогичны уже выполненным. У руководителя проекта имеется опыт реализации таких задач и понимание возможных проблем. В этих случаях иерархическая структура проекта и работ (ИСП/ИСР) формируется с применением подхода Top-down (сверху вниз), используется типовая структура проектной команды, планы проекта (план управления рисками, план коммуникаций и пр.) аналогичны планам предыдущих проектов. Однако 10 процентов проектов – инновационные, реализуемые "с нуля" и требующие творчества, нестандартных решений и управленческой смелости. Принятие решений в таких проектах характеризуется высокими рисками, что требует от руководителя глубоких знаний методики проектного управления и понимания особенностей её применения в сфере информационных технологий.

Управление IT-проектом

Применение **методологии управления проектами** позволяет зафиксировать цели и результаты проекта, дать им количественные характеристики, определить временные, стоимостные и качественные параметры проекта, создать реальный план выполнения проекта, выделить, оценить риски и предотвратить возможные негативные последствия во время реализации проекта.

Для эффективного управления проект должен быть хорошо структурирован. Суть этого процесса сводится к выделению следующих основных элементов:

- фазы жизненного цикла проекта, этапов, работ и отдельных задач;
- организационная структура исполнителей проекта;
- структура распределения ответственности.

Жизненный цикл IT-проекта

Жизненный цикл – это последовательность фаз проекта, через которые он должен пройти для гарантированного достижения целей проекта, в нашем случае – для реализации некоторой информационной технологии.

Организационная структура подразумевает выделение ролей исполнителей, которые необходимы для реализации проекта, определение взаимоотношений между ними и распределение ответственности за выполнение задач.

Жизненный цикл IT-проекта

Жизненный цикл проекта

Любой проект имеет ограниченный отрезок времени существования. Наличие этого отрезка времени означает, что у проектов есть жизненный цикл. Жизненный цикл последовательно проходить через четыре стадии:

1. Определение
2. Планирование
3. Выполнение
4. Сдача

Все начинается, когда появляется идея проекта. Происходит его определение. Здесь поднимается вопрос о цели проекта, о его целесообразности. Далее проект переходит в стадию планирования, где планируется бюджет, принимаются ключевые решения. Бюджет проекта имеет большое значение. Так, например, на воплощение проекта с большим бюджетом, в среднем затрачивается меньше времени, чем на проект с меньшим бюджетом. К этой стадии необходимо относится ответственно, потому что если на этом этапе совершается архитектурная ошибка, то в случае, если ошибку невозможно исправить, проект либо меняет свою структуру (а это тянет за собой дополнительные расходы ресурсов), либо прекращает свое существование. На стадии выполнения происходит непосредственное воплощение проекта. Здесь проект может претерпевать значительные изменения. После стадии выполнения проект сдается. Под сдачей подразумевается введение проекта в эксплуатацию.

Жизненный цикл IT-проекта

В IT-проектах можно выделить следующие особенности:

- по мере реализации проектов выполняется большое количество уточнение и корректировка требований и содержания;
- крайне велико влияние человеческого фактора на сроки выполнения;
- из-за трудности планирования творческой деятельности, отсутствуют единые нормативы и стандарты;
- сохраняется повышенный уровень риска, вплоть до непредсказуемости результатов;
- происходит постоянное совершенствование технологии выполнения работ, что тянет за собой постоянное изменение структуры проекта.

Что касается стандартизации, то современные стандарты не предписывают четких и однозначных схем построения структуры жизненного цикла проекта. Это сделано намеренно, поскольку достаточно жесткие схемы препятствуют использованию более прогрессивных технологий разработки, которых появилось очень много и которые продолжают интенсивно развиваться.

Исходя из этих особенностей, можно построить структуру жизненного цикла it-проекта:

- **Начальная стадия** - цель**:** определить границы системы и собрать требования высокого уровня;
- **Стадия уточнения -** цель**:** создать архитектурную основу системы;
- **Стадия конструирования** - цель**:** создание финального продукта;
- **Стадия передачи и сопровождения** - цель**:** внедрение продукта на предприятии заказчика, обучение персонала, сопровождение и обновление установленной информационной системы.

Начальная стадия жизненного цикла it-проекта не отличается по структуре от любого другого проекта. На стадии уточнения производится выбор технологий, определение необходимых ресурсов (как денежных, так и человеческих), построение команды. Также здесь происходит подписание необходимых документов. Таких как техническое задание и документов, которые призваны защитить компанию-разработчика от внезапного отказа заказчика от проекта. Уже на стадии уточнения может начинаться тестирование. Такой подход называется *test-driven development* (TDD, Разработка через тестирование). Он заключается в написании тестов раньше, чем написание кода. Далее следует **стадия конструирования**, которую можно условно разделить на фазы части:

- **Проектирование** - определение характеристик архитектуры системы, компонентов, составляющих, интерфейсов и других частей. Созданное описание, в свою очередь, является фундаментом для реализации;
- **Реализация** - создание программного продукта, исходя из созданного на этапе проектирования программного проекта;
- **Интеграция** - объединение программных компонентов и интегрирование их в среду.

Основная часть стадии конструирования, **реализация**, может быть поделена еще на 6 частей:

- **Пре-Альфа** - период времени со старта разработки до выхода стадии Альфа (или до любой другой, если стадии Альфа нет). Также так называются программы, не вышедшие еще в стадию альфа или бета, но прошедшие стадию разработки, для первичной оценки функциональных возможностей в действии. В отличие от альфа- и бета-версий, пре-альфа может включать в себя не весь спектр функциональных возможностей программы. В этом случае, подразумеваются все действия, выполняемые во время проектирования и разработки программы вплоть до тестирования. К таким действиям относятся — разработка дизайна, анализ требований, собственно разработка приложения, а также отладка отдельных модулей.
- **Альфа** - внутреннее тестирование — стадия начала тестирования программы в целом специалистами-тестерами, обычно не разработчиками программного продукта, но, как правило, внутри организации или сообществе разрабатывающих продукт. Также это может быть стадия добавления новых функциональных возможностей. Программы на данной стадии могут применяться только для ознакомления с будущими возможностями.
- **Бета** - публичное тестирование — Стадия активного бета-тестирования и отладки программы. Программы этого уровня могут быть использованы другими разработчиками программного обеспечения для испытания совместимости. Тем не менее, программы этого этапа могут содержать достаточно большое количество ошибок.
- **Релиз-кандидат** - иногда «гамма-версия» — стадия-кандидат на то, чтобы стать стабильной. Программы этой стадии прошли комплексное тестирование, благодаря чему были исправлены все найденные критические ошибки. Но в то же время существует вероятность выявления ещё некоторого числа ошибок, не замеченных при тестировании.
- **Релиз** - издание продукта, готового к тиражированию. Это стабильная версия программы, прошедшая все предыдущие стадии, в которых исправлены основные ошибки, но существует вероятность появления новых, ранее не замеченных, ошибок.
- **Пост-релиз** - издание продукта, у которого есть несколько отличий от релизного и помечается как самая первая стадия разработки следующего продукта. Такие релизы не выпускаются на продажу, а раздаются бета-тестировщикам. Эта стадия встречается редко и присуща проектам, которые делятся на отдельно реализуемые версии.

Третья по счету фаза конструирования - **интеграция**. На этой фазе происходит интеграция разрабатываемого проекта с уже созданным окружением, с которым этот проект будет функционировать.

И последняя, четвертая, стадия жизненного цикла it-проекта - **стадия передачи и сопровождения**. Передача проекта заказчику - очень сложный процесс и требует тщательного подхода и ответственности. Какие здесь возникают проблемы? Часто возникают спорные вопросы из-за того, что требования к системе были сформулированы абстрактно либо недостаточно хорошо. Качественное техническое задание - оружие компании-разработчика. Однако наличие технического задания не является ни необходимым, ни достаточным основанием для полноценного закрытия работ. Если исполнитель придерживается ГОСТов, к приемочным испытаниям на основании технического задания должен быть разработан и согласован с заказчиком дополнительный документ "Программа и методика испытаний". В нем должны быть

прописаны принципы оценки реализации требований технического задания. Также на этой стадии производится поддержка и сопровождение проекта. Длительность сопровождения зависит от сложности, целесообразности, выгодности и области применения проекта. Так, например, социальные сети поддерживаются и сопровождаются на протяжении всего времени использования, а проект, который перестал приносить прибыль, быстро теряет поддержку со стороны производителя.

Особенности open-source проектов

У open-source проектов выделяется ряд специфических особенностей:

- техническое задание либо размытое, либо и вовсе отсутствует;
- большое количество исполнителей (contributors);
- отсутствие четко сформированного будущего проекта;
- непредсказуемость продолжительности жизни.

Моделирование жизненного цикла

Мозговой штурм

Управление жизненным циклом

Хотя управление и соответствие нормативным требованиям встречаются на протяжении всего жизненного цикла, однако их представление, область действия и цели зависят от конкретного этапа. Например, действия по управлению изменениями на этапах «Планирование» и «Эксплуатация» будут иметь другую значимость и отличаться по составу участников и используемым факторам.

Внутренние меры контроля

Процедуры и меры контроля следует разделить между несколькими людьми, каждый из которых будет выполнять свою часть. В этом случае внутренние меры контроля должны обеспечить надлежащее объединение результатов, полученных разными людьми, и гарантировать, что никому не удалось уклониться от выполнения. В финансовых вопросах проблемы контроля являются еще более важными. Отсутствие эффективного контроля может привести к ошибкам в бухгалтерском учете или даже мошенничеству и хищению.

Внутренние меры контроля представлены во всех областях, с которыми работает IT-подразделение. Одни меры контроля предназначены для физической среды, в которой находится инфраструктура центров данных, а другие используются непосредственно для технологий (например, определяют конфигурацию и перечень лиц, которым предоставлен доступ к административным функциям). Некоторые меры контроля используются при доступе к данным и применяются на различных этапах жизненного цикла данных — от шифрования до авторизации, восстановления и защиты данных.

Подтверждением того, что IT-услуга фактически контролируется на протяжении всего жизненного цикла, является следующее:

- Определение общих целей для каждого этапа жизненного цикла
- Определение рисков, связанных с достижением этих целей
- Определение методов управления рисками в виде мер внутреннего контроля по смягчению последствий рисков

Руководство несет ответственность за выработку целей, оценку хода работ и достижение результатов. В частности, управление включает процессы принятия решений (меры контроля), помогающие руководству выполнять эти требования. Каждый этап жизненного цикла IT-услуги содержит одну или несколько процедур управленческого анализа, функционирующих как управленческие меры контроля. Это означает, что нужные люди будут собраны вместе в надлежащее время и обеспечены информацией, необходимой для принятия управленческих решений.

Контроль исполнения проекта - процесс сравнения показателей плановых и фактических показателей выполнения проекта, анализ отклонений и их причин, оценка возможных альтернатив и принятие, в случае необходимости, решений о корректирующих действиях для ликвидации нежелательных отклонений.

Контроль проекта может включать следующие процедуры:

- Сбор отчетности о ходе работ по проекту
- Анализ текущего состояния проекта относительно основных базовых показателей (результаты, стоимость, время)
- Прогнозирование достижения целей проекта
- Подготовка и анализ последствий корректирующих воздействий
- Принятие решений о воздействиях и изменениях

Организация контроля может следить за:

- Качеством работ
- Ходом и темпом работ
- Стоимостью и сроками

Классификация программного обеспечения

Глава 2. Менеджмент IT-проекта

Менеджмент IT-проектов

Роли в команде и их функции

Для чего нужно разделение ролей

Каждый IT-специалист идет по пути наименьшего сопротивления, как правило, пытаясь сбросить часть работы на коллегу. Для избегания возможных конфликтов в команде нужно четко разграничить участок работ каждого участника, в том числе заказчика. Для целенаправленного выполнения проекта должен быть выполнен ряд работ, различных как по своему назначению, так и по квалификационным требованиям, предъявляемым к *разработчикам*. Иными словами, в ходе развития проекта командой *разработчиков* выполняются те или иные функции.

Распределение функций между исполнителями

Функции, выполняемые разработчиками*,* — понятие неформализованное. В разных проектах оно может обретать свое содержание. Заметим, что в рамках деятельности менеджера любого проекта необходимо организовать *распределение функций* проекта между исполнителями. В результате ее

выполнения члены команды, выполняющей проект, начинают играть соответствующие роли. Обычно роль объединяет родственные функции. Также принято обозначать роли их главными функциями. Продолжая только что приведенную иллюстрацию функций, выполняемых *разработчиками* проекта, укажем на следующие роли: кодировщик — действующее лицо, главной функцией которого является кодирование, аналитик — тот, кто занимается анализом требований. Подобную характеристику можно дать и *тестировщику*. Что же касается функции отладки, то в реальных проектах она обычно подразделяется на несколько видов: отладка компонентов, которой занимаются *разработчики* компонентов (например, те же кодировщики) и комплексная отладка, которая может поручаться специально выделенным сотрудникам или рассматриваться в качестве задачи группы *разработчиков*. Часто выполняются такие работы, как отладка спецификаций, декомпозиция проекта и др. Иными словами, функция отладки обычно не рассматривается как образующая роль, а распределяется по нескольким ролям в соответствии с принятой стратегией развития проекта. Роли назначаются на начальной стадии жизненного цикла проекта.

Не следует путать функции, которые предписано выполнять *разработчику* как исполнителю определенной роли, с поручениями в проекте. Поручения — это разовые или систематические задания, из которых обычно и складываются действия, необходимые для выполнения той или иной функции. Если для функции определен регламент выполнения поручений, т. е. последовательность выполнения составляющих ее поручений не требует дополнительных разъяснений для исполнителя, то такая функция называется технологической. В случае нетехнологической функции сотруднику, который выполняет соответствующую роль, приходится самому выстраивать нужную последовательность. Иными словами, технология здесь уступает место ремеслу.

При разработке любых проектов естественно стремление к повышению их технологичности, к установлению регламента для как можно большего числа функций. И одним из способов достижения этого для менеджера является использование работников с нужной квалификацией, для которых поручаемые им роли оказываются технологичными, т.е. состоящими только из *технологических функций*. К сожалению, реальность такова, что менеджеру приходится работать в условиях ограниченных возможностей в подборе кадров, а потому уровень технологичности выполнения проекта снижается по сравнению с идеальной ситуацией. Таким образом, в рамках любого проекта возникает задача повышения квалификации сотрудников. Для различных схем ведения проектов эта задача решается по-разному. Крайнюю точку зрения на проблему соответствия квалификации работников требованиям проекта отражает идея *экстремального программирования*, когда используется неформальная организация группы исполнителей проекта без четкого распределения ролей, а значит, и обязательств сотрудников. В этом случае провозглашается принцип, когда каждый в группе делает то, что он умеет делать лучше всего. И хотя все функции, которые должны выполняться, остаются, создается впечатление, что в группе исполнителей проекта исчезают роли. В результате возможны пробелы в разработке, в частности при анализе и декомпозиции проектируемой системы. Чтобы этого избежать, *разработчики* должны понимать, какую абстрактную роль они исполняют в каждый конкретный момент, выполнение каких проектных функций необходимо сейчас, как связаны между собой работы, как должны распределяться ресурсы. Словом, они должны обращать внимание на выполнение распределенных по группе менеджерских функций. И даже в этом случае схему *экстремального программирования* можно рекомендовать лишь для слаженных групп исполнителей с высоким уровнем коллективной ответственности.

Функции, выполняемые *разработчиками* в проекте, подразделяются на:

- Организационные
- Производственные

Первые создают условия для выполнения проектных заданий, вторые непосредственно связаны с этими заданиями. Часто неудачи проекта возникают из-за того, что менеджер не учитывает важность выполнения *организационных функций*. Так, обычно проектное задание фиксирует лишь то, что нужно предъявлять *заказчику* в качестве результатов. С точки зрения результатов просто не требуется знать, например, как организована передача проектных материалов между *разработчиками*, какая процедура отчетности предусматривается, но игнорирование задач реализации информационных потоков в проекте может привести к хаосу, что в конечном итоге отразится и на результатах.

Понятно, что как состав, так и значимость ролей *разработчиков* и тех, кто с ними связан, различаются в зависимости от выполняемого проекта, от коллектива исполнителей, принятой технологии, от других факторов. Неоднозначность выбора ролей приводит к тому, что их список часто становится непомерно велик. Чрезмерное увеличение числа есть следствие отождествления роли и функции и, соответственно, игнорирования понятия родственности функций. В то же время, если ролей выбрано недостаточно, есть опасность, что не все нужные в проекте функции будут охвачены планированием и управлением.

Также при выборе состава нужно учитывать направление того или иного человека. В случае крупных проектов необходимо нанимать узкопрофильным специалистов, так как они лучше знают свою сферу. Если у проекта небольшой бюджет, то можно сэкономить, наняв многопрофильных специалистов. К слову, есть такой класс разработчиков, как full-stack разработчики — это еще одна попытка «работодателей» получить задешево, то, что никогда дешевым быть не могло. Это такой же психологический прием, как и «вы же профессионал! Вы же профессионал?»

Цепочку распределений ролей можно описать в следующем виде:

- спонсор (куратор) проекта (это сотрудник (как правило, руководитель высшего звена) организации, реализующей проект, который курирует проект со стороны организации (владельца проекта), обеспечивает общий контроль и поддержку проекта) назначает менеджера проекта и обеспечивает ему необходимую поддержку.
- менеджер проекта выбирает команду управления проектом, среди которых есть командный лидер (team leader).
- командный лидер назначает разработчиков.

Подход компании Microsoft к распределению ролей

Как конкретный *разработчик* может получать одновременно несколько ролей, так и роль может быть распределена между несколькими исполнителями. Когда менеджеру в конкретных условиях руководства коллективом придется распределять роли, он неизбежно столкнется с тем, что эта задача зависит и от специфики проекта, и от контингента исполнителей. В связи с этим уместно упомянуть об одном из ее решений, которое предлагается специалистами Microsoft в качестве универсального подхода.

Предлагается образовывать небольшие мобильные коллективы как атомарные производственные единицы с общей ответственностью за выполняемые задания — так называемые проектные группы. Такие группы строятся как многопрофильные команды, члены которых распределяют между собой ответственность и дополняют *области компетентности* друг друга. Группа состоит не более чем из 10 человек. Все они считаются обладающими сходным уровнем профессиональной подготовки, хотя и в разных областях индивидуальной специализации. Провозглашается равноправие членов группы и коллективная ответственность за выполняемые задания: проектная группа — команда равных. Все это позволяет сохранять внутри группы неформальные отношения. Вместо понятия роли для группы в целом определяются ***ролевые кластеры***, которые заполняются точно так же, как происходит распределение ролей. В то время как за успех проекта ответственна вся команда, каждый из ее *ролевых кластеров*, определяемых моделью, ассоциирован с одной из проектных целей и работает над ее достижением. В данной модели именно эти цели задают роли *разработчиков*, которые определяются кластерами. В терминологии используется понятие *области компетенции*, или *области функциональной специализации* (functional area), обозначающее ту или иную роль, которую выполняет кластер группы в проекте. Принципиальное отличие распределения исполнителей по *ролевым кластерам* от распределения ролей заключается лишь в том, что ответственность за это несет не *менеджер проекта*, а сама группа. Менеджер проекта выдает задания и контролирует их выполнение лишь в целом для группы, не вмешиваясь в ее работу.

Определено шесть **ролевых кластеров**, которые соответствующим образом структурируют проектные функции разработчиков:

- **Управление продуктом (product management)**. Ключевая цель кластера — обеспечивать удовлетворение интересов заказчика. Для ее достижения кластер должен содержать следующие области компетенции:
    - планирование продукта;
    - планирование доходов;
    - представление интересов заказчика;
    - маркетинг.
- **Управление программой (program management)**. Задача — обеспечить реализацию решения в рамках ограничений проекта, что может рассматриваться как удовлетворение требований к бюджету проекта и к его результату. Области компетенции кластера:
    - управление проектом;
    - выработка архитектуры решения;
    - контроль производственного процесса;
    - административные службы.
- **Разработка (development)**. Первостепенной задачей кластера является построение решения в соответствии со спецификацией. Области компетенции кластера:
    - технологическое консультирование;
    - проектирование и осуществление реализации;
    - разработка приложений;
    - разработка инфраструктуры.
- **Тестирование (test)**. Задача кластера — одобрение выпуска продукта только после того, как все дефекты выявлены и устранены. Области компетенции кластера:

    - разработка тестов;
    - отчетность о тестах;
    - планирование тестов.
- **Удовлетворение потребителя (user experience)**. Цель кластера — повышение эффективности использования продукта. Области компетенции кластера:
    - общедоступность (возможности работы для людей с недостатками зрения, слуха и др.);
    - интернационализация (эксплуатация в иноязычных средах);
    - обеспечение технической поддержки;
    - обучение пользователей;
    - удобство эксплуатации (эргономика);
    - графический дизайн.
- **Управление выпуском (release management)**. Задача кластера — беспрепятственное внедрение и сопровождение продукта. Области компетенции кластера:
    - инфраструктура (infrastructure);
    - сопровождение (support);
    - бизнес-процессы (operations);
    - управление выпуском готового продукта (commercial release management).

Основные роли, встречающиеся на проекте и их обязанности

Существуют следующие роли на IT-проектах:

- **Заказчик (Customer)** — отвечает за:
    - своевременный просмотр спецификаций и других присылаемых документов (с целью утвердить документ, дать комментарии, исправить неточности и т.п.);
    - внесение замечаний, дефектов, пожеланий в багтрекинговую систему;
    - своевременный просмотр каждого выпуска и предоставление комментариев.
- **Планировщик ресурсов (Planner)**:
    - выдвигает и координирует требования к проектам в организации, осуществляющей данную разработку;
    - развивает и направляет план выполнения проекта с точки зрения организации.
- **Менеджер проекта (Project Manager)** — отвечает за:
    - проектная документация;
    - составление плана проекта;
    - согласование сроков;
    - анализ возможных рисков;
    - участие в подборе и утверждении проектной команды;
    - разбивка продукта на компоненты и раздача их исполнителям;
    - определение требуемых ресурсов и рабочей среды, их распределение внутри команды;
    - постановка рабочего процесса в команде (разработка, тестирование, работа с требованиями);
    - определение приоритетности задач;
    - организация работы команды вокруг требуемой задачи;
    - отслеживание состояния проекта, хода выполнения задач;

- отслеживание должной приоритетности выполнения задач;
- отслеживание нагрузки задачами и прогресса по задачам каждого разработчика;
- отслеживание сроков выполнения задач;
- удерживание команды в рабочем состоянии, мотивация команды;
- создание прозрачной среды общения между всеми участниками процесса;
- отслеживание удовлетворенности проектом со стороны команды;
- решение всевозможных конфликтных ситуаций внутри команды и в связке заказчик-команда;
- общение с заказчиком, управление его ожиданиями;
- предоставление заказчику отчетности о ходе выполнения задач и проекта в целом;
- презентация заказчику готовых решений, демоверсий, прототипов;
- интервьюирование новых членов команды.

- **Руководитель команды (Team Leader)**

  Руководитель команды — это нечто среднее между проектным менеджером и квалифицированным разработчиком. Командный лидер обязан перевести бизнес-задачу в понятную техническую для разработчиков и сказать не только то, что нужно сделать, но и зачем это нужно.

  На проектах есть две роли: менеджерская — PM, и техническая — System Architect. Командный лидер отчасти выполняет обе роли, но акцент его обязанностей направлен на менеджмент (акцент на техническую часть — это tech lead).

  Под управленческую роль TL попадают такие обязанности, как:

  - менеджмент;
  - распределение и делегирование задач:
  - всевозможные оценки и составление рабочего графика;
  - контроль состояния проекта;
  - проведение митингов;
  - коммуникации с заказчиком, руководством и всеми членами команды (разработчиками, архитекторами, тестировщиками, менеджерами).

  Под техническую роль TL попадают**:**

  - участие в написании технической документации;
  - выбор технологий для проекта;
  - разработка архитектуры;
  - обзор и анализ кода (code review);
  - контроль и наставление молодых разработчиков;
  - проведение технических собеседований;
  - грамотное вовлечение новых членов команды в рабочий процесс;
  - ответственность за техническую часть проекта.
- **Системный аналитик (Technical Leader) —** отвечает за:
  - координацию и контроль деятельности по дизайну, архитектуре и кодированию;
  - поддержку контроля версий;
  - настройку скрипта для авто-билдера и своевременную сборку версий.

- **Архитектор (Architect)** — отвечает за:
    - проектирование архитектуры системы;
    - согласование развития работ, связанных с проектом.

  Архитектор — это человек, который решает, как в конечном итоге будет выглядеть информационная система организации в целом и в деталях. Основная цель архитектора в компании заключается в том, чтобы обеспечить решение задач бизнеса при помощи информационных технологий. Причем, он должен не только сформировать решение, но и контролировать правильность его реализации.

  Внутри профессии существуют специализации: *функциональная* и *техническая*. В первом случае архитектор в большей степени отвечает за общение с бизнесом и по результатам контактов определяет конструкцию системы, которая нужна заказчику. Во втором ИТ-архитектор в основном общается с разработчиками и конструирует систему изнутри.

  Не всякой компании нужен ИТ-архитектор. На небольших предприятиях или там, где информационные проекты не слишком масштабны, функции ИТ-архитектора может выполнять опытный менеджер проекта, разработчик или иной технический специалист в сфере ИТ.

  Иметь собственного ИТ-архитектора необходимо, в первую очередь, крупным компаниям с развитой функциональностью унаследованных систем, разветвленной региональной оргструктурой и имеющим согласованные руководством планы развития ИТ.

- **Эксперт предметной области (Domain Expert)** — отвечает за:
    - изучение сферы приложения;
    - поддержку направленности проекта на решение задач данной области.
- **Разработчик (Developer)**:
    - разработку качественного кода;
    - проведение модульного тестирования;
    - поддержку контроля версий;
    - написание пользовательской документации, относящейся к инсталляции и администрированию.

  Это широкое понятие, которое может подразделяться на специальные роли (например, разработчик классов). В зависимости от сложности проекта команда может включать различное число разработчиков.

- **Бизнес аналитик (Business Analyst)** отвечает за:
    - выяснение и анализ всех требований заказчика;
    - фиксирование всех требований заказчика (в багтрекинговой системе и в функциональных спецификациях), прослеживание всех изменений в требованиях;
    - написание и поддержка спецификаций.
- **Разработчик информационной поддержки (Information Developer**):
    - создает документацию, сопровождающую продукт, когда выпускается версия. Включаемые в нее инсталляционные материалы, равно как ссылочные и учебные, а также материалы помощи предоставляются на бумажных и машинных носителях.

    - Для сложных проектов возможно распределение этих задач между несколькими разработчиками информационной поддержки.
- **Специалист по пользовательскому интерфейсу (Human Factors Engineer)**:
    - отвечает за удобство применения системы;
    - работает с заказчиком, чтобы удостовериться, что пользовательский интерфейс удовлетворяет требованиям.
- **QA менеджер (Quality Assurance manager) —** отвечает за:
    - организацию и контроль процесса тестирования в проекте;
    - планирование тестирования;
    - участие в адаптации процесса разработки под проект, анализ его качества;
    - анализ результатов тестирования и качества продукта;
    - участие в управлении требованиями;
    - участие в настройке багтрекинговой системы, полное прослеживание багов;
    - контроль готовности нового выпуска для QA.
- **QA аналитик (QA Analyst)** — отвечает за:
    - подготовку тест дизайна;
    - написание тест кейс спецификаций;
    - проведение тестирования;
    - регистрацию багов;
    - прослеживание и проверку багов;
    - написание документации пользователя.

Возможность совмещения ролей

Роли	**Характеристика совмещения ролей**
Менеджер и архитектор	Желательно
Менеджер и руководитель команды	Противоречиво
Руководитель команды и архитектор	Возможно
Руководитель команды и проектировщик подсистемы	Нежелательно
Менеджер и разработчик	Не допускается
Для различных разработчиков	Эффективно с ограничениями
Создание документации (все сотрудники)	Успешно распределяется
Специалист по интерфейсу и менеджер	Разумно
Эксперт предметной области и менеджер	Зачастую разумно
Специалист по интерфейсу и эксперт предметной области	Редко бывает эффективно
Эксперт предметной области и разработчик	Бывает полезно
Специалист по интерфейсу и разработчик	Часто полезно
Библиотекарь и один из разработчиков	Допустимо
Тестировщики и другие члены команды	Перекрестно
Эксперт предметной области, тестировщик	Оправданно

Распределение ролей посредством RACI-матрицы

Модель RACI — средство для выявления активностей и распределения их по ролям и зонам ответственности. Использование матрицы RACI позволяет избежать непонимания в том, кого необходимо привлечь к проекту, а также кто и что должен делать.

RACI — сокращение от основных ролей участников проекта:

- **Responsible (Исполнитель)**: Тот кому назначена эта роль отвечает за выполнение работы и достижение целей проекта. На каждом этапе может быть несколько исполнителей.
- **Accountable (Ответственный)**: Исполнитель этой роли отвечает за качество и результаты процесса. Обладатель этой роли обеспечивается полномочиями для обратной связи с исполнителями. На каждом этапе может быть только один ответственный.
- **Consulted (Консультант, Эксперт)**: Тот кому назначена эта роль привлекается, как носитель уникальных знаний или информации. Часто в этой роли выступают эксперты в предметной области.
- **Informed (Информируемый)**: Это лицо, которого необходимо держать в курсе о ходе и результатах процесса, чаще всего в одностороннем порядке, т.к. у него нет полномочий напрямую влиять на ход проекта.

Иногда в эту модель добавляются и другие роли, например, S — supported (Оказывающий поддержку).

	Director Service Management	Service Level Manager	Problem Manager	Security Manager	Procurement Manager
Activity 1	AR	C	I	I	C
Activity 2	A	R	C	C	C
Activity 3	I	A	R	I	C
Activity 4	I	A	R	I	
Activity 5	I	I	A	C	I

Для того, чтобы понимать, по какому принципу такая табличка должна рисоваться, а также как ее использовать на практике (в реальных проектах), рекомендуется уделить должное внимание следующему порядку действий при построении матрицы:

- Определяется список необходимых активностей/процессов в поставленной задаче (проекте)
- Определяется и указывается функциональные роли (людей, которые заинтересованы или которых тем или иным образом касается данная задача)
- Собирается митинг и назначаются RACI коды (собственно — буквы) конкретным ролям, непосредственно разграничиваются ответственности
- Определяются несоответствия (например, слишком много ответственных либо отсутствие таковых)

- Описывается таблица и собираются отзывы
- Контролируется выполнение назначенных ролей

По функциональным ролям, анализировать можем, отвечая на такие вопросы:

- *Много «A»* — правильно ли распределены обязанности? Есть ли в наличии «узкие места»?
- *Много «R»* — не многовато ли ответственности для одной роли?
- *Отсутствие пустых ячеек в таблице* – действительно ли эта роль должна быть вовлечена в такое количество задач?

Также, проводится анализ по выполняемым активностям:

- *Более одного «A»* — только одна роль должна быть подотчетной
- *Отсутствие «A»* — необходимо найти подотчетного
- *Более одного «R» или отсутствие такового* – кто-то должен быть ответственный, однако нетребуется, чтобы ответственность была широко распределена – есть риск того, что задача не будет выполнена
- *Много «C»* — стоит ли консультироваться с многими ролями и будет ли это эффективно?
- *Отсутствие «C» и «I»* — правильно ли установлены коммуникации?

Пример использования RACI-матрицы

Допустим, есть авиакомпания, которая на своем сайте собирается внедрить систему online check-in. Глобальные активности, необходимы к выполнению в контексте задачи, будут приблизительно следующие: сбор требований к системе; дизайн решения; непосредственная разработка решения (development); внедрение; собственно – стадия “production”; оптимизация решения.

Далее — определяется список функциональных ролей, в данной задачи возможны такие варианты: внутренний сервис провайдер (IT отдел авиакомпании) или же внешний сервис провайдер в случае отсутствия первого; ISP – компания предоставляющая хостинг для сайта авиакомпании; бизнес подразделение авиакомпании (представляющее интересы заказчика); финансовое подразделение (бухгалтерия); сервис менеджер (в зависимости от размеров организации, может входить во внутренний IT отдел); команда разработчиков (в зависимости от размеров организации, может входить во внутренний IT отдел).

Попробуем расставить RACI коды соответственно ролям и выполняемым ими активностям (ясно, что данный процесс проходит при участии всех сторон).

	IT Отдел	ISP	Бизнес подразделение	Бухгалтерия	Сервис менеджер	Разработчики
Требования	R	I	C	C	A	I
Дизайн	C	C	I	I	AR	C
Разработка	C	I	I	I	C	AR
Внедрение	AR	R	C	I	C	C
Production	AR	R	I	I	I	I
Оптимизация	R	C	C	C	A	R

Глава 3. Архитектура компьютера и мобильных устройств

Глава 4. Языки программирования

Общие понятия

Транслятор — программа или техническое средство, выполняющее *трансляцию программы*.

Трансляция программы — преобразование программы, представленной на одном из языков программирования, в программу на другом языке и, в определённом смысле, равносильную первой.

Транслятор обычно выполняет также диагностику ошибок, формирует словари идентификаторов, выдаёт для печати текст программы и т. д.

Язык, на котором представлена входная программа, называется *исходным языком*, а сама программа — *исходным кодом*. Выходной язык называется *целевым языком, а выходная (результирующая) программа — объектным кодом*.

В общем случае, понятие трансляции относится не только к языкам программирования, но и к другим языкам — как формальным компьютерным (вроде языков разметки типа HTML), так и естественным (русскому, английскому и т. п.)

Виды трансляторов

Существует несколько видов трансляторов:

- *Диалоговый* транслятор — транслятор, обеспечивающий использование языка программирования в режиме разделения времени.
- *Синтаксически-ориентированный* (*синтаксически-управляемый)* транслятор — транслятор, получающий на вход описание синтаксиса и семантики языка, текст на описанном языке и выполняющий трансляцию в соответствии с заданным описанием.
- *Однопроходной* транслятор — транслятор, создающий объектный модуль при однократном последовательном чтении исходного кода (за один проход).

- *Многопроходной* транслятор — транслятор, создающий объектный модуль после нескольких чтений исходного кода (за несколько проходов).
- *Оптимизирующий* транслятор — транслятор, выполняющий оптимизацию создаваемого кода перед записью в объектный файл.
- *Тестовый* транслятор — транслятор, получающий на вход исходный код и выдающий на выходе изменённый исходный код. Запускается перед основным транслятором для добавления в исходный код отладочных процедур. Например, транслятор с языка ассемблера может выполнять замену макрокоманд на код.
- *Обратный* транслятор — транслятор, выполняющий преобразование машинного кода в текст на каком-либо языке программирования.

Реализации

Цель трансляции — преобразование текста с одного языка на язык, понятный адресату. При трансляции компьютерной программы адресатом может быть:

- устройство — процессор (трансляция называется *компиляцией*);
- программа — интерпретатор (трансляция называется *интерпретацией*).

Виды трансляции

- компиляция;
- интерпретация;
- динамическая компиляция.

Компилятор

Компиляция — трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду (абсолютный код, объектный модуль, иногда на язык ассемблера). Входной информацией для компилятора (исходный код) является описание алгоритма или программа на объектно-ориентированном языке, а на выходе компилятора — эквивалентное описание алгоритма на машинно-ориентированном языке (объектный код).

Виды компиляторов

- *Векторизующий*. Транслирует исходный код в машинный код компьютеров, оснащённых векторным процессором.
- *Гибкий*. Сконструирован по модульному принципу, управляется таблицами и запрограммирован на языке высокого уровня или реализован с помощью компилятора компиляторов.
- *Диалоговый*.
- *Инкрементальный*. Повторно транслирует фрагменты программы и дополнения к ней без перекомпиляции всей программы.
- *Интерпретирующий (пошаговый)*. Последовательно выполняет независимую компиляцию каждого отдельного оператора (команды) исходной программы.
- *Компилятор компиляторов*. Транслятор, воспринимающий формальное описание языка программирования и генерирующий компилятор для этого языка.

Синтаксис выражается в виде Форма Бэкуса — Наура (формальная система описания синтаксиса, в которой одни синтаксические категории последовательно определяются через другие категории. БНФ используется для описания контекстно-свободных формальных грамматик, Пример: от пример БНФ-конструкции, описывающей правильные скобочные последовательности: <правпосл>::=<пусто> | (<правпосл>) | <правпосл><правпосл>) или её производной и должен удовлетворять правилам того метода синтаксического анализа, который будет использоваться в генерируемом компиляторе.

Семантика языка обычно описывается путём ассоциирования процедуры генерации кода с каждой синтаксической конструкцией, причём необходимая процедура вызывается всякий раз, когда соответствующая конструкция распознаётся программой синтаксического анализа. Таким образом, пользователю компилятора компиляторов в любом случае нужно разработать исполняющие структуры и выбрать способ преобразования каждой входной синтаксической конструкции в операции выходного языка или в машинные операции, после чего нужно написать собственно процедуры генерации кода. Следовательно, компилятор компиляторов — это полезное средство, помогающее писать компиляторы, но не более того.

Компилятор компиляторов Bison

Bison – это GNU-выпуск известной программы YACC, предназначенной для порождения компиляторов по описанной пользователем КС-грамматике.

- *Отладочный*. Устраняет отдельные виды синтаксических ошибок.
- *Резидентный*. Постоянно находится в оперативной памяти и доступен для повторного использования многими задачами.
- *Самокомпилируемый*. Написан на том же языке, с которого осуществляется трансляция. Метод создания транслятора для некоторого языка программирования, при котором транслятор пишется на том же языке программирования; создание транслятором исполняемых файлов из исходного кода самого транслятора. Используется для переноса трансляторов на новые платформы. Появился в середине 1950-х годов. Позволяет создать транслятор, который генерирует сам себя. Применялся для создания трансляторов многих языков программирования, включая языки BASIC, Алгол, Си, Паскаль, ПЛ/1, Factor, Haskell, Modula-2, Oberon, OCaml, Common Lisp, Scheme, Java, Python, Scala, Nemerle и другие.
- *Универсальный*. Основан на формальном описании синтаксиса и семантики входного языка. Составными частями такого компилятора являются: ядро, синтаксический и семантический загрузчики

Виды компиляции

- *Пакетная*. Компиляция нескольких исходных модулей в одном пункте задания.
- *Построчная*. То же, что и интерпретация.
- *Условная*. Компиляция, при которой транслируемый текст зависит от условий, заданных в исходной программе директивами компилятора. Так, в зависимости от значения некоторой константы, можно включать или выключать трансляцию части текста программы.

Процесс компиляции

Процесс компиляции состоит из следующих этапов:

- Лексический анализ. На этом этапе последовательность символов исходного файла преобразуется в последовательность лексем.
- Синтаксический (грамматический) анализ. Последовательность лексем преобразуется в дерево разбора.
- Семантический анализ. Дерево разбора обрабатывается с целью установления его семантики (смысла) — например, привязка идентификаторов к их декларациям, типам, проверка совместимости, определение типов выражений и т. д. Результат обычно называется «промежуточным представлением/кодом», и может быть дополненным деревом разбора, новым деревом, абстрактным набором команд или чем-то ещё, удобным для дальнейшей обработки.
- Оптимизация. Выполняется удаление излишних конструкций и упрощение кода с сохранением его смысла. Оптимизация может быть на разных уровнях и этапах — например, над промежуточным кодом или над конечным машинным кодом.
- Генерация кода. Из промежуточного представления порождается код на целевом языке.

В конкретных реализациях компиляторов эти этапы могут быть разделены или, наоборот, совмещены в том или ином виде.

Генерация кода

Генерация машинного кода

Большинство компиляторов переводит программу с некоторого высокоуровневого языка программирования в машинный код, который может быть непосредственно выполнен процессором. Как правило, этот код также ориентирован на исполнение в среде конкретной операционной системы, поскольку использует предоставляемые ею возможности (системные вызовы, библиотеки функций). Архитектура (набор программно-аппаратных средств), для которой производится компиляция, называется *целевой машиной*.

Результат компиляции — исполнимый модуль — обладает максимальной возможной производительностью, однако привязан к определённой операционной системе и процессору (и не будет работать на других).

Для каждой целевой машины (IBM, Apple, Sun и т. д.) и каждой операционной системы или семейства операционных систем, работающих на целевой машине, требуется написание своего компилятора. Существуют также так называемые *кросс-компиляторы*, позволяющие на одной машине и в среде одной ОС генерировать код, предназначенный для выполнения на другой целевой машине и/или в среде другой ОС. Кроме того, компиляторы могут оптимизировать код под разные модели из одного семейства процессоров (путём поддержки специфичных для этих моделей особенностей или расширений наборов инструкций). Например, код, скомпилированный под процессоры семейства Pentium, может учитывать особенности распараллеливания инструкций и использовать их специфичные расширения — MMX, SSE и т. п.

Некоторые компиляторы переводят программу с языка высокого уровня не прямо в машинный код, а на язык ассемблера (примером может служить PureBasic, транслирующий бейсик-код в ассемблер FASM). Это делается для упрощения части компилятора, отвечающей за кодогенерацию, и

повышения его переносимости (задача окончательной генерации кода и привязки его к требуемой целевой платформе перекладывается на ассемблер), либо для возможности контроля и исправления результата компиляции программистом.

Генерация байт-кода

Результатом работы компилятора может быть программа на специально созданном низкоуровневом языке, подлежащем интерпретации *виртуальной машиной*. Такой язык называется псевдокодом или байт-кодом. Как правило, он не является машинным кодом какого-либо компьютера и программы на нём могут исполняться на различных архитектурах, где имеется соответствующая виртуальная машина, но в некоторых случаях создаются аппаратные платформы, напрямую поддерживающие псевдокод какого-либо языка. Например, псевдокод языка Java называется байт-кодом Java и выполняется в Java Virtual Machine, для его прямого исполнения была создана спецификация процессора picoJava. Для платформы .NET Framework псевдокод называется Common Intermediate Language (CIL), а среда исполнения — Common Language Runtime (CLR).

Некоторые реализации интерпретируемых языков высокого уровня (например, Perl) используют байт-код для оптимизации исполнения: затратные этапы синтаксического анализа и преобразование текста программы в байт-код выполняются один раз при загрузке, затем соответствующий код может многократно использоваться без промежуточных этапов.

Динамическая компиляция

Из-за необходимости интерпретации байт-код выполняется значительно медленнее машинного кода сравнимой функциональности, однако он более переносим (не зависит от операционной системы и модели процессора). Чтобы ускорить выполнение байт-кода, используется *динамическая компиляция*, когда виртуальная машина транслирует псевдокод в машинный код непосредственно перед его первым исполнением (и при повторных обращениях к коду исполняется уже скомпилированный вариант).

CIL-код также компилируется в код целевой машины JIT-компилятором, а библиотеки .NET Framework компилируются заранее.

Декомпиляция

Существуют программы, которые решают обратную задачу — перевод программы с низкоуровневого языка на высокоуровневый. Этот процесс называют декомпиляцией, а такие программы — декомпиляторами. Но поскольку компиляция — это процесс с потерями, точно восстановить исходный код, скажем, на C++, в общем случае невозможно. Более эффективно декомпилируются программы в байт-кодах — например, существует довольно надёжный декомпилятор для Flash. Разновидностью декомпилирования является дизассемблирование машинного кода в код на языке ассемблера, который почти всегда выполняется успешно (при этом сложность может представлять самомодифицирующийся код или код, в котором собственно код и данные не разделены). Связано это с тем, что между кодами машинных команд и командами ассемблера имеется практически взаимно-однозначное соответствие.

Раздельная компиляция

Трансляция частей программы по отдельности с последующим объединением их компоновщиком в единый загрузочный модуль.

Исторически особенностью компилятора, отражённой в его названии (англ. *compile* — собирать вместе, составлять), являлось то, что он производил как трансляцию, так и компоновку, при этом компилятор мог порождать сразу абсолютный код. Однако позже, с ростом сложности и размера программ (и увеличением времени, затрачиваемого на перекомпиляцию), возникла необходимость разделять программы на части и выделять библиотеки, которые можно компилировать независимо друг от друга. При трансляции каждой части программы компилятор порождает объектный модуль, содержащий дополнительную информацию, которая потом, при компоновке частей в исполнимый модуль, используется для связывания и разрешения ссылок между частями.

Появление раздельной компиляции и выделение компоновки как отдельной стадии произошло значительно позже создания компиляторов. В связи с этим вместо термина «компилятор» иногда используют термин «транслятор» как его синоним: либо в старой литературе, либо когда хотят подчеркнуть его способность переводить программу в машинный код (и наоборот, используют термин «компилятор» для подчёркивания способности собирать из многих файлов один).

Интерпретатор

Интерпретатор — программа (разновидность транслятора), выполняющая *интерпретацию*.

Интерпретация — пооператорный (покомандный, построчный) анализ, обработка и тут же выполнение исходной программы или запроса (в отличие от компиляции, при которой программа транслируется без её выполнения).

Типы интерпретаторов

- **Простой интерпретатор** анализирует и тут же выполняет (собственно интерпретация) программу покомандно (или построчно), по мере поступления её исходного кода на вход интерпретатора. Достоинством такого подхода является мгновенная реакция. Недостаток — такой интерпретатор обнаруживает ошибки в тексте программы только при попытке выполнения команды (или строки) с ошибкой.
- **Интерпретатор компилирующего типа** — это система из компилятора, переводящего исходный код программы в промежуточное представление, например, в байт-код, и собственно интерпретатора, который выполняет полученный промежуточный код (так называемая виртуальная машина). Достоинством таких систем является большее быстродействие выполнения программ (за счёт выноса анализа исходного кода в отдельный, разовый проход, и минимизации этого анализа в интерпретаторе). Недостатки — большее требование к ресурсам и требование на корректность исходного кода. Применяется в таких языках, как Java, PHP, Tcl, Perl, REXX (сохраняется результат парсинга исходного кода[j]), а также в различных СУБД.

В случае разделения интерпретатора компилирующего типа на компоненты получаются компилятор языка и простой интерпретатор с минимизированным анализом исходного кода. Причём исходный код для такого интерпретатора не обязательно должен иметь текстовый формат или быть байт-кодом, который понимает только данный интерпретатор, это может быть машинный код какой-то существующей аппаратной платформы. К примеру, виртуальные машины вроде QEMU, Bochs, VMware включают в себя интерпретаторы машинного кода процессоров семейства x86.

Некоторые интерпретаторы (например, для языков Лисп, Scheme, Python, Бейсик и других) могут работать в режиме диалога или так называемого цикла чтения-вычисления-печати (англ. *read-eval-print loop, REPL*). В таком режиме интерпретатор считывает законченную конструкцию языка (например, s-expression в языке Лисп), выполняет её, печатает результаты, после чего переходит к ожиданию ввода пользователем следующей конструкции.

Уникальным является язык Forth, который способен работать как в режиме интерпретации, так и компиляции входных данных, позволяя переключаться между этими режимами в произвольный момент, как во время трансляции исходного кода, так и во время работы программ.

Следует также отметить, что режимы интерпретации можно найти не только в программном, но и аппаратном обеспечении. Так, многие микропроцессоры интерпретируют машинный код с помощью встроенных микропрограмм, а процессоры семейства x86, начиная с Pentium (например, на архитектуре Intel P6), во время исполнения машинного кода предварительно транслируют его во внутренний формат (в последовательность микроопераций).

Алгоритм работы простого интерпретатора

1. прочитать инструкцию;
2. проанализировать инструкцию и определить соответствующие действия;
3. выполнить соответствующие действия;
4. если не достигнуто условие завершения программы, прочитать следующую инструкцию и перейти к пункту 2.

Достоинства и недостатки

Достоинства

- Бóльшая переносимость интерпретируемых программ — программа будет работать на любой платформе, на которой есть соответствующий интерпретатор.
- Как правило, более совершенные и наглядные средства диагностики ошибок в исходных кодах.
- Меньшие размеры кода по сравнению с машинным кодом, полученным после обычных компиляторов.

Недостатки

- Интерпретируемая программа не может выполняться отдельно без программы-интерпретатора. Сам интерпретатор при этом может быть очень компактным.
- Интерпретируемая программа выполняется медленнее, поскольку промежуточный анализ исходного кода и планирование его выполнения требуют дополнительного времени в сравнении с непосредственным исполнением машинного кода, в который мог бы быть скомпилирован исходный код.
- Практически отсутствует оптимизация кода, что приводит к дополнительным потерям в скорости работы интерпретируемых программ.

Динамический компилятор

JIT-компиляция (англ. *Just-in-time compilation*, компиляция «на лету»), **динамическая компиляция** (англ. *dynamic translation*) — технология увеличения производительности программных систем, использующих байт-код, путём компиляции байт-кода в машинный код или в другой формат

непосредственно во время работы программы. Таким образом достигается высокая скорость выполнения по сравнению с интерпретируемым байт-кодом (сравнимая с компилируемыми языками) за счёт увеличения потребления памяти (для хранения результатов компиляции) и затрат времени на компиляцию. JIT базируется на двух более ранних идеях, касающихся среды исполнения: *компиляции байт-кода* и *динамической компиляции*.

Так как JIT-компиляция является, по сути, одной из форм динамической компиляции, она позволяет применять такие технологии, как адаптивная оптимизация и динамическая рекомпиляция. Из-за этого JIT-компиляция может показывать лучшие результаты в плане производительности, чем статическая компиляция. Интерпретация и JIT-компиляция особенно хорошо подходят для динамических языков программирования, при этом среда исполнения справляется с поздним связыванием типов и гарантирует безопасность исполнения.

Проекты LLVM, GNU Lightning, libJIT (часть проекта DotGNU) и RPython (часть проекта PyPy) могут быть использованы для создания JIT интерпретаторов любого скриптового языка.

Особенности реализации

JIT-компиляция может быть применена как ко всей программе, так и к её отдельным частям. Например, текстовый редактор может на лету компилировать регулярные выражения для более быстрого поиска по тексту. С AOT-компиляции (компиляция перед исполнением) такое сделать не представляется возможным, так как данные предоставляются во время исполнения программы, а не во время компиляции. JIT используется в реализациях Java, JavaScript, .NET Framework, в одной из реализаций Python — PyPy. Существующие наиболее распространённые интерпретаторы языков PHP, Ruby, Perl, Python и им подобных, которые имеют ограниченные или неполные JIT.

Большинство реализаций JIT имеют последовательную структуру: сначала (AOT) приложение компилируется в байт-код виртуальной машины среды исполнения, а потом JIT компилирует байт-код непосредственно в машинный код. В итоге приложение подтормаживает при запуске, что, впоследствии, компенсируется более быстрой его работой.

Описание

В языках, таких как Java, PHP, C#, Lua, Perl, GNU CLISP, исходный код транслируется в одно из промежуточных представлений, называемое байт-кодом. Байт-код не является машинным кодом какого-либо конкретного компьютера и может переноситься на различные компьютерные архитектуры и исполнятся точно так же. JIT читает байт-код из некоторых секторов (редко сразу из всех) и компилирует их в машинный код. Этим сектором может быть файл, функция или любой фрагмент кода. Однажды скомпилированный код может кэшироваться и в дальнейшем повторно использоваться без перекомпиляции.

Динамически компилируемая среда — это среда, в которой компилятор может вызываться приложением во время выполнения. Например, большинство реализаций Common Lisp содержат функцию `compile`, которая может создать функцию во время выполнения; в Python это функция `eval`. Это удобно для программиста, так как он может контролировать, какие части кода действительно подлежат компиляции. Также с помощью этого приёма можно компилировать динамически сгенерированный код, что в некоторых случаях приводит даже к лучшей производительности, чем реализация в статически скомпилированном коде. Однако стоит

помнить, что подобные функции могут быть опасны, особенно когда данные передаются из недоверенных источников.

Основная цель использования JIT — достичь и превзойти производительность статической компиляции, сохраняя при этом преимущества динамической компиляции:

- Большинство тяжеловесных операций, таких как парсинг исходного кода и выполнение базовых оптимизаций происходит во время компиляции (до развёртывания), в то время как компиляция в машинный код из байт-кода происходит быстрее, чем из исходного кода
- Байт-код более переносим (в отличие от машинного кода)
- Среда может контролировать выполнение байт-кода после компиляции, поэтому приложение может быть запущено в песочнице (стоит отметить, что для нативных программ такая возможность тоже существует, но реализация данной технологии сложнее)
- Компиляторы из байт-кода в машинный код легче в реализации, так как большинство работы по оптимизации уже было проделано компилятором

JIT, как правило, эффективней, чем интерпретация кода. К тому же в некоторых случаях JIT может показывать большую производительность по сравнению со статической компиляцией за счёт оптимизаций, возможных только во время исполнения:

1. Компиляция может осуществляться непосредственно для целевого CPU и операционной системы, на которой запущено приложение. Например, JIT может использовать векторные SSE2 расширения процессора, если он обнаружит их поддержку. Однако, до сих пор нет основных реализаций JIT, где этот подход бы использовался, ведь чтобы обеспечить подобный уровень оптимизации, сравнимый со статическими компиляторами, потребовалось бы либо поддерживать бинарный файл под каждую платформу, либо включать в одну библиотеку оптимизаторы под каждую платформу.
2. Среда может собирать статистику о работающей программе и производить оптимизации с учётом этой информации. Некоторые статические компиляторы также могут принимать на вход информацию о предыдущих запусках приложения.
3. Среда может делать глобальные оптимизации кода (например, встраивание библиотечных функций в код) без потери преимуществ динамической компиляции и без накладных расходов, присущим статических компиляторам и линкерам.
4. Более простое пристраивание кода для лучшего использования кэша

Задержка при запуске, средства борьбы с ней

Типичная причина задержки при запуске JIT-компилятора — расходы на загрузку среды и компиляцию приложения в байт-код. В общем случае, чем лучше и чем больше оптимизаций выполняет JIT, тем дольше получается задержка. Поэтому разработчикам JIT приходится искать компромисс между качеством генерируемого кода и временем запуска. Однако, часто оказывается так, что узким местом в процессе компиляции оказывается не сам процесс компиляции, а задержки системы ввода вывода (так, например, *rt.jar* в Java Virtual Machine (JVM) имеет размер 40MB, и поиск метаданных в нём занимает достаточно большое количество времени).

Ещё одно средство оптимизации — компилировать только те участки приложения, которые используются чаще всего. Этот подход реализован в PyPy и Sun’s HotSpot Java Virtual Machine.

В качестве эвристики может использоваться счётчик запусков участков приложения, размер байт-кода или детектор циклов.

Порой достаточно сложно найти правильный компромисс. Так, например, Sun's Java Virtual Machine имеет два режима работы — клиент и сервер. В режиме клиента количество компиляций и оптимизаций минимально для более быстрого запуска, в то время как в режиме сервера достигается максимальная производительность, но из-за этого увеличивается время запуска.

Ещё одна техника, называемая pre-JIT, компилирует код до запуска. Преимуществом данной техники является ускоренное время запуска, в то время недостатком является плохое качество скомпилированного кода по сравнению с runtime JIT.

История

Самую первую реализацию JIT можно отнести к LISP, написанную McCarthy in 1960[5]. В его книге *Recursive functions of symbolic expressions and their computation by machine, Part I,* он упоминает функции, компилируемые во время выполнения, тем самым избавив от надобности вывода работы компилятора на перфокарты.

Другой ранний пример упоминания JIT можно отнести к Кену Томпсону, который в 1968 году впервые применил регулярные выражения для поиска подстрок в текстовом редакторе QED. Для ускорения алгоритма Томпсон реализовал компиляцию регулярных выражений в машинный код IBM 7094.

Важный метод получения скомпилированного кода был предложен Митчелом в 1970 году, когда он реализовал экспериментальный язык *LC*^{2}.

Smalltalk (1983) был пионером в области JIT-технологий. Трансляция в машинный код выполнялась по требованию и кэшировалась для дальнейшего использования. Когда память кончалась, система могла удалить некоторую часть закэшированного кода из оперативной памяти и восстановить его, когда он снова потребуется. Язык программирования Self некоторое время был самой быстрой реализацией Smalltalk-а и работал всего-лишь в два раза медленней C, будучи полностью объектно-ориентированным.

Self был заброшен Sun, но исследования продолжились в рамках языка Java. Термин «Just-in-time компиляция» был заимствован из производственного термина «Точно в срок» и популяризован Джеймсом Гослингом, использовавшим этот термин в 1993. В данный момент JIT используется почти во всех реализациях Java Virtual Machine.

Также большой интерес представляет диссертация, защищённая в 1994 году в Университете ETH (Швейцария, Цюрих) Михаэлем Францем «Динамическая кодогенерация — ключ к переносимому программному обеспечению» и реализованная им система Juice динамической кодогенерации из переносимого семантического дерева для языка Оберон. Система Juice предлагалась как плагин для интернет-браузеров.

Безопасность

Так как JIT составляет исполняемый код из данных, возникает вопрос безопасности и возможных уязвимостей.

JIT компиляция включает в себя компиляцию исходного кода или байт-кода в машинный код и его выполнение. Как правило, результат записывается в память и исполняется сразу же, не используя диск и не вызывая код как отдельную программу. В современных архитектурах для повышения безопасности произвольные участки памяти не могут быть исполнены. Для корректной работы память должна быть помечена, как исполняемая (NX bit); для большей безопасности флаг должен быть поставлен *после* загрузки кода в память, а эта память должна быть помечена как доступная только для чтения, так как перезаписываемая и исполняемая память есть ничто иное, как дыра в безопасности.

Глава 5. Базы данных

Реляционные базы данных. SQL

Нереляционные базы данных

Глава 6. Разработка программного обеспечения

Разница между разработкой и производством программного обеспечения

Парадигмы программирования

Стили программирования

Паттерны проектирования

Разработка мобильных приложений

Тестирование

Стандартизация

Глава 7. Активно развивающиеся технологии

Искусственный интеллект

Виртуальная реальность

Нейронные сети

Облачные вычисления

Глава 8. Компьютерная и информационная безопасность

> Только атаки дилетантов нацелены на машины.
> Атаки профессионалов нацелены на людей.
>
> Б. Шнайер

Компьютерная и информационная безопасность

Компьютерная и информационная безопасность

Компьютерная безопасность — меры безопасности, применяемые для защиты вычислительных устройств (компьютеры, смартфоны и другие), а также компьютерных сетей (частных и публичных сетей, включая Интернет). Поле деятельности системных администраторов охватывает все

процессы и механизмы, с помощью которых цифровое оборудование, информационное поле и услуги защищаются от случайного или несанкционированного доступа, изменения или уничтожения данных, и приобретает всё большее значение в связи с растущей зависимостью от компьютерных систем в развитом сообществе.

Информационная безопасность — это процесс обеспечения конфиденциальности, целостности и доступности информации. Конфиденциальность: обеспечение доступа к информации только авторизованным пользователям. Целостность: Обеспечение достоверности и полноты информации и методов ее обработки. Доступность: Обеспечение доступа к информации и связанным с ней активам авторизованных пользователей по мере необходимости.

Зачастую, люди путают эти два понятия. Внесём ясность.

Компьютерная безопасность - это безопасность компьютера, его программного обеспечения, информации, заложенной в нём. Она связана конкретно с компьютерами и техникой. А ***информационная безопасность*** - это защищенность информационных ресурсов (документов и массивов документов какой-либо организации, например), а также защита прав личности и государства в информационной сфере (например, от разглашения гос. тайн или от пиратства) и много чего другого.

На самом деле, если рассуждать строго, ***компьютерная безопасность*** - это меры, направленные на исключение нарушения работоспособности компьютера (или компьютеров) или управления компьютером (или компьютерами) сторонними лицами. Доступ к компьютеру злоумышленника ещё не означает кражу информации. Информация может быть зашифрована или находиться на недоступном для злоумышленника носителе. ***Специалист по информационной безопасности*** работает в конкретном направлении, составляя нечто подобное плану защиты информации, воплощают это в жизнь как правило уже другие люди, в зависимости от распределения обязанностей организации. ***Специалист по компьютерной безопасности*** это частный случай ***специалиста по информационной безопасности***. Чаще всего тот, кто занимается вопросами компьютерной безопасности, он же их и воплощает в жизнь, поэтому ***специалист по компьютерной безопасности*** - больше теоретик, ***по информационной*** – практик.

Информация является одним из наиболее ценных ресурсов любой компании, поэтому обеспечение защиты информации является одной из важнейших и приоритетных задач.

В силу развитых технологий не стоит безответственно относиться к своим личным данным, цифровым данным, которые при определённых обстоятельствах оказывают влияние на нашу судьбу. Уже ни для кого не секрет, что электронные девайсы, которыми пользуется человек в современном мире для передачи информации, подвержены многим уязвимостям, а также и они сами и информация представляют огромный интерес для злоумышленников и правительств.

Аргумент "мне нечего скрывать" не считается весомым, так как даже с первого взгляда безобидная информация может поведать о каждом конкретном индивидууме многое: личные предпочтения, круг общения, перемещения и прочие вещи, которые в умелых руках могут стать отличным инструментом для достижения тех или иных целей в отношении этого индивидуума.

Ну и по поводу размещения информации в сети интернет. Не выкладывайте свои личные данные, никаких фотографий, никакой привязки аккаунтов к реальному вашему номеру телефона (для этого

существуют сервисы подобные этим: **http://receivesmsonline.com/ http://www.vsms24.com/web/index.php**), никакой установки игровых клиентов на компьютер с важными данными и никакой серьёзной переписки в чатах этих онлайн-игр. (Но следует помнить и о том, что спецслужбы при необходимости могут сохранять коммуникации на том лишь основании, что они зашифрованы, а затем хранить зашифрованное сообщение вечно или по крайней мере до тех пор, пока его не расшифруют. О незашифрованных данных известно, что все логи посещений сайтов сохраняются, а система СОРМ может записывать ваш незашифрованный трафик.)

Уничтожение и шифрование данных

Также необходимо помнить о безвозвратном удалении следов информации с жёсткого диска, для чего целесообразнее применять CCleaner, PrivaZer, wipefile. И вот ещё список того, что желательно уничтожать:

Информация на компьютере:

- **Неиспользуемые пароли и данные к аккаунтам.** Удалить невосстанавливаемым программным методом.
- **Закладки в браузере.** Внимательно просмотреть и выявить все ненужное.
- **Браузеры.** Очистить всю историю и любые другие данные, сохраненные сайтами.
- **Неиспользуемые контакты.** Просмотреть и удалить неиспользуемые контакты: icq, jabber, skype, другие im-приложения, почтовые программы и сервисы.
- **Связка gpg-ключей.** Просмотреть все связки и удалить неиспользуемые, забытые и неизвестные публичные и секретные ключи.
- **Сообщения.** Удалить ненужные и старые сообщения в почтовых программах, icq, jabber-клиентах, skype и прочих программах обмена сообщениями.
- **Документы.** Просмотреть папки хранения документов — старые, неиспользуемые и ненужные удалить невосстанавливаемым программным методом.
- **Бекапы.** Просмотреть места хранения бекапов и удалить старые невосстанавливаемым программным методом.
- **Скачанное.** Очистить локальную папку для скачивания файлов.
- **Неиспользуемое свободное место на дисках.** Для очистки использовать программы Ccleaner и Bleachbit.
- **Пароли.** Решить, где лучше сменить пароли.

Интернет:

- "**Личка".** Просмотреть все посещаемые форумы и остальные социальные сервисы и удалить все личные сообщения.
- **Сообщения.** Удалить старые и ненужные сообщения в почтовых сервисах.

Средства общения, мобильные компьютеры

- **Контакты.** Пройтись по контактам в телефонах и удалить ненужные.
- **Сообщения.** Удалить все СМС и другие сообщения на смартфоне/планшете.
- **Документы.** Найти и удалить ненужные и забытые документы и книги на смартфоне/планшете.

- **Закладки в браузере.** Внимательно просмотреть и выявить все ненужное.
- **Браузеры.** Очистить всю историю и любые другие данные, сохраненные сайтами.
- **Неиспользуемые контакты в im-приложениях**. Просмотреть и удалить неиспользуемые контакты: icq, jabber, skype, другие im-приложения, почтовые программы и сервисы.
- **Приложения.** Выявить и удалить старые и ненужные приложения, а также информацию, оставленную ими.
- **Пароли.** Решить, где лучше сменить пароли.

Материальное

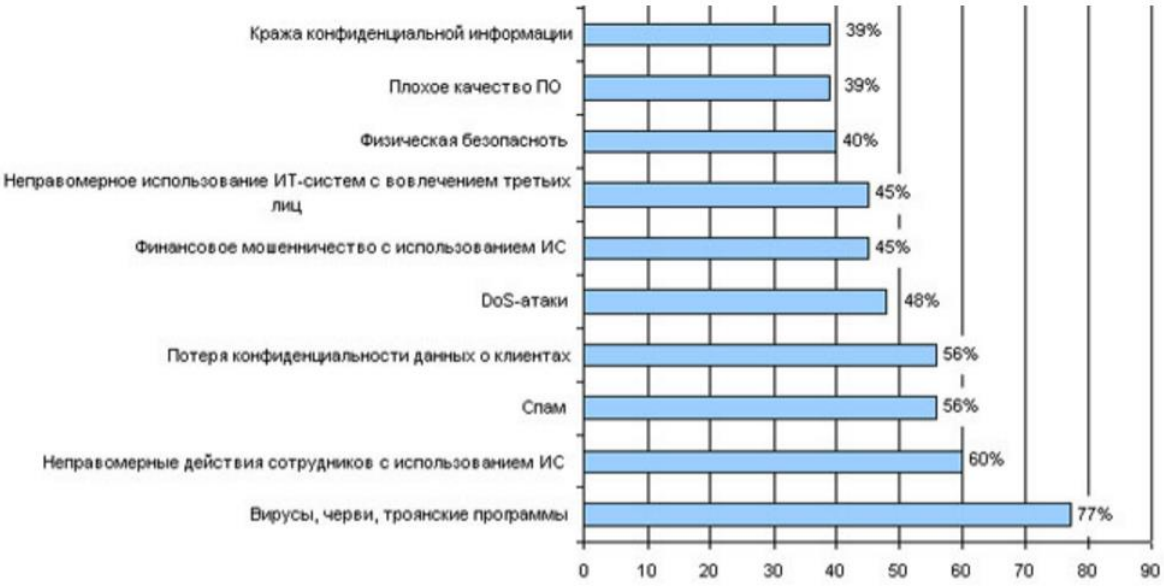
- **Флешки и другие носители информации.** Проверить все носители информации в пределах досягаемости и уничтожить (разломать, сжечь и т.п.) информацию на них по своему усмотрению.
- **Телефоны, модемы, Wi-Fi модули и другие средства коммуникации.** Использованные, старые, ненужные и по-другому "засвеченные" средства коммуникации физически сломать и выбросить минимум в трех кварталах (километрах) от их местоположения.
- **СИМ-карты.** Выявить и уничтожить (разломать, сжечь и т.п.) все забытые и неиспользуемые СИМ-карты и другие карты доступа.
- **Документы.** Выявить ненужные и забытые документы, записки, заметки, записные книжки и уничтожить (лучше сжечь).

Желательно "не оставлять следов": использование кредитных карт, мобильных телефонов даёт возможность отследить ваше передвижение, покупки, контакты. Нужно иметь в виду, что каждое использование компьютера также оставляет следы.

Также следует упомянуть о необходимости шифрования данных. Ведь оно защищает нашу частную информацию и анонимность. Эта защита важна для каждого. Более того, в последние годы вопрос конфиденциальности в сети становится очень актуальным, учитывая сканирование наших электронных писем. Сегодня самым известным и простым алгоритмом, который используется в системах защиты информации организаций и предприятий, является DES (Data Encryption Standart). Он необходим для защиты от незаконного доступа к важной информации в организациях. Особенности алгоритма DES — используется ключ длиной 56 бит; зашифровать можно сообщение с помощью одной программы, а расшифровать — используя любую другую программу, соответствующую DES; высокая скорость обработки достигается за счёт несложного алгоритма и высокой стойкости. Data Encryption Standart подходит для шифрования и аутентификации данных.

Угрозы безопасности информационных систем

Известны следующие источники угроз безопасности информационных систем:



Средства защита информации от несанкционированного доступа

Получение доступа к ресурсам информационной системы предусматривает выполнение трех процедур: *идентификация, аутентификация* и *авторизация*.

Идентификация - присвоение пользователю (объекту или субъекту ресурсов) уникальных имен и кодов (идентификаторов).

Аутентификация - установление подлинности пользователя, представившего идентификатор или проверка того, что лицо или устройство, сообщившее идентификатор является действительно тем, за кого оно себя выдает. Наиболее распространенным способом аутентификации является присвоение пользователю пароля и хранение его в компьютере.

Авторизация - проверка полномочий или проверка права пользователя на доступ к конкретным ресурсам и выполнение определенных операций над ними. Авторизация проводится с целью разграничения прав доступа к сетевым и компьютерным ресурсам.

Защита информации в компьютерных сетях

Локальные сети предприятий очень часто подключаются к сети Интернет. Для защиты локальных сетей компаний, как правило, применяются межсетевые экраны - брандмауэры (firewalls). Экран (firewall) - это средство разграничения доступа, которое позволяет разделить сеть на две части (граница проходит между локальной сетью и сетью Интернет) и сформировать набор правил, определяющих условия прохождения пакетов из одной части в другую. Экраны могут быть реализованы как аппаратными средствами (различные устройства, системы, платформы), так и программными.

Межсетевой экран, сетевой экран, файрвол или брандмауэр — это комплекс аппаратных и программных средств в компьютерной сети, осуществляющий контроль и фильтрацию проходящих через него сетевых пакетов в соответствии с заданными правилами.

Основной задачей сетевого экрана является защита сети или отдельных её узлов от несанкционированного доступа. Также сетевые экраны часто называют фильтрами, так как их основная задача — не пропускать (фильтровать) пакеты, не подходящие под критерии, определённые в конфигурации.

Проблемы, не решаемые фаерволом

Межсетевой экран сам по себе не панацея от всех угроз для сети. В частности, он:

- не защищает узлы сети от проникновения через «люки» (англ. back doors) или уязвимости ПО;
- не обеспечивает защиту от многих внутренних угроз, в первую очередь — утечки данных;
- не защищает от загрузки пользователями вредоносных программ, в том числе вирусов;

Для решения последних двух проблем используются соответствующие дополнительные средства, в частности, *антивирусы*.

Но зачем Linux антивирус нужен вообще, ведь она практически не подвержена вирусным атакам? А ответ очень прост: многие линуксоиды пользуются не только Linux, но ставят, или оставляют на компьютере, второй системой Windows. Поэтому антивирус нужен, чтобы сканировать раздел Windows, когда она не работает, на возможность заражения. Многие вирусы могут легко прятаться от антивируса, но их намного легче обнаружить именно тогда, когда Windows спит. Именно так поступаю некоторые антивирусы в Windows: они перезагружают Windows и начинают её сканировать перед загрузкой. Но даже если у вас не стоит второй системой Windows, вам может потребоваться Linux антивирус, чтобы не стать разносчиком вирусов. Простая ситуация: вы взяли у друга флешку, скопировали нужные вам файлы, а в них вирус. Вам хоть бы что, и это здорово. Но вы передаёте эти файлы третьему человеку, у которого Windows, и заражаете его этими вирусами. Ну и третий вариант - это сервер, на котором стоит Linux. Но вот в офисе все работают на Windows, и вам просто необходимо мониторить интернет трафик на наличие вирусов. И в этом случае Linux антивирус хорошее решение, так как он может прекрасно работать в режиме командной строки.

Считается, что Unix-системы намного лучше защищены от компьютерных вирусов, в сравнении с операционными системами Windows. В принципе, до сих пор нет ни одного широко распространенного вируса для Linux, в то время как в среде Windows просто кишит всякой заразой. в Linux очень быстро исправляют уязвимости.

Существует мнение, что вирусов на Linux мало потому, что его доля на рынке операционных систем крайне мала, поэтому и создают вирусы для Windows, из-за его большой популярности и распространенности. Будь Linux популярнее, и для него так же написали бы кучу вирусов. Но это не так. Если взять самое популярное программное обеспечение для web-серверов, то это будет Apache. По состоянию на 2013 год доля Apache на web-серверах составляет почти 45%, а доля IIS от Microsoft — 23.10% рынка. Из этого следует, что атаки хакеров должны быть активнее к большему числу интернет-ресурсов на Apache, и должно наблюдаться больше червей, вирусов и других вредоносных программ, которые будут нацелены на Apache и те операционные системы, под

управлением которых он работает, чем на Windows и IIS. Но в жизни дела обстоят иначе. Уже много времени целью для сетевых червей и всевозможных атак есть IIS от Microsoft.

Основные достоинства Linux, в плане безопасности:

- для *nix-платформ создано крайне мало вирусов;
- существует большое количество разных Linux-дистрибутивов. Одни работают с .deb пакетами, другие с .RPM пакетами. Создание вируса, который будет одинаково хорошо работать во всех из них очень затруднительно;
- в разных дистрибутивах по умолчанию инсталлированы разные наборы, по-разному скомпилированного программного обеспечения, что также уменьшает вероятность массового заражения вирусом;
- программы, которые скачаны из интернета, в Linux по умолчанию не являются исполняемыми. Сперва их нужно сделать таковыми;
- главным источником ПО в Linux являются проверенные (официальные) репозитории. Это дает право утверждать, что вероятность попадания вирусов в эти источники крайне мала.

Криптографическая защита информации

Для обеспечения секретности информации применяется ее шифрование или криптография. Для шифрования используется алгоритм или устройство, которое реализует определенный алгоритм. Управление шифрованием осуществляется с помощью изменяющегося кода ключа.

Извлечь зашифрованную информацию можно только с помощью ключа. Криптография - это очень эффективный метод, который повышает безопасность передачи данных в компьютерных сетях и при обмене информацией между удаленными компьютерами.

Электронная подпись

Для исключения возможности модификации исходного сообщения или подмены этого сообщения другим необходимо передавать сообщение вместе с электронной подписью. Электронная цифровая подпись - это последовательность символов, полученная в результате криптографического преобразования исходного сообщения с использованием закрытого ключа и позволяющая определять целостность сообщения и принадлежность его автору при помощи открытого ключа.

Другими словами, сообщение, зашифрованное с помощью закрытого ключа, называется электронной цифровой подписью. Отправитель передает незашифрованное сообщение в исходном виде вместе с цифровой подписью. Получатель с помощью открытого ключа расшифровывает набор символов сообщения из цифровой подписи и сравнивает их с набором символов незашифрованного сообщения.

При полном совпадении символов можно утверждать, что полученное сообщение не модифицировано и принадлежит его автору.

Целью применения систем цифровой подписи является аутентификация информации - защита участников информационного обмена от навязывания ложной информации, установление факта модификации информации, которая передается или сохраняется, и получения гарантии ее подлинности, а также решение вопроса об авторстве сообщений. Система цифровой подписи предполагает, что каждый пользователь сети имеет свой секретный ключ, который используется

для формирования подписи, а также соответствующий этому секретному ключу открытый ключ, известный некоторому кругу пользователей сети и предназначенный для проверки подписи. Цифровая подпись вычисляется на основе секретного ключа отправителя информации и собственно информационных бит документа (файла). Один из пользователей может быть избран в качестве "нотариуса" и заверять с помощью своего секретного ключа любые документы. Остальные пользователи могут провести верификацию его подписи, то есть убедиться в подлинности полученного документа. Способ вычисления цифровой подписи таков, что знание открытого ключа не может привести к подделке подписи. Проверить подпись может любой пользователь, имеющий открытый ключ, в том числе независимый арбитр, который уполномочен решать возможные споры об авторстве сообщения (документа).

Защита информации от компьютерных вирусов

Компьютерный вирус – это небольшая вредоносная программа, которая самостоятельно может создавать свои копии и внедрять их в программы (исполняемые файлы), документы, загрузочные сектора носителей данных и распространяться по каналам связи.

Существует несколько классификаций компьютерных вирусов:

1. **По среде обитания** различают вирусы сетевые, файловые, загрузочные и файлово-загрузочные.
2. **По способу заражения** выделяют резидентные и нерезидентные вирусы.
3. **По степени воздействия** вирусы бывают неопасные, опасные и очень опасные.
4. **По особенностям алгоритмов** вирусы делят на паразитические, репликаторы, невидимки, мутанты, троянские, макро-вирусы.

Загрузочные вирусы заражают загрузочный сектор винчестера или дискеты и загружаются каждый раз при начальной загрузке операционной системы.

Резидентные вирусы загружается в память компьютера и постоянно там находится до выключения компьютера.

Самомодифицирующиеся вирусы (мутанты) изменяют свое тело таким образом, чтобы антивирусная программа не смогла его идентифицировать.

Стелс-вирусы (невидимки) перехватывает обращения к зараженным файлам и областям и выдают их в незараженном виде.

Троянские вирусы маскируют свои действия под видом выполнения обычных приложений.

Троянская программа — вредоносная программа, распространяемая людьми, в отличие от вирусов и червей, которые распространяются самопроизвольно. В данную категорию входят программы, осуществляющие различные несанкционированные пользователем действия: сбор информации и её передачу злоумышленнику, её разрушение или злонамеренную модификацию, нарушение работоспособности компьютера, использование ресурсов компьютера в неблаговидных целях.

Существуют Password Trojans. Эти трояны находят на вашем компьютере пароли и отправляют их хозяину. Нет разницы, какой пароль, будь то интернет пароль, пароль от почтового ящика, этот троян найдет все эти пароли.

Keyloggers трояны очень просты в использовании. Они записывают все, что было набрано на клавиатуре (включая пароли) в файл и впоследствии отправляют это все по электронной почте хозяину. Keylogger-ы обычно занимают мало места, и могут маскироваться под другие полезные программы, из-за чего их бывает трудно обнаружить. Некоторые трояны этого типа могут выделять и расшифровывать пароли, найденные в специальных полях для ввода паролей. Обычно информация с паролями отсылается хозяину трояна по электронной почте.

Одним из самых эффективных методов защиты от кейлоггеров является использование виртуальной клавиатуры. Не стоит спешить прибегать к помощи виртуальной клавиатуры, встроенной в Windows. Эта программа не предназначена для защиты от кейлоггеров. Подобного рода виртуальные клавиатуры не защищены от считывания информации, набираемой на них. В данном случае нужна специально разработанная виртуальная клавиатура, наподобие той, что входит в состав продуктов Лаборатории Касперского. Наличие виртуальной клавиатуры в программах данной лаборатории подтверждает факт сложности обнаружения кейлоггеров. В настоящее время в большинстве серьезных банков используются виртуальные клавиатуры для защиты своих клиентов от кражи вводимой информации при доступе к интернет-банкинг системам. Новое бесплатное приложение Oxynger KeySheild позволяет защитить данные от передачи хакерам без ведома пользователя. Оно реализовано в форме виртуальной клавиатуры, помогающей безопасно вводить критически важную информацию и таким образом избегать необходимости использовать аппаратную клавиатуру, за которой, возможно, следят кейлоггеры.

Современные кейлоггеры умеют соотносить клавиатурный ввод с текущим окном и элементом ввода. Многие из них умеют отслеживать список работающих приложений, делают “фото” экрана по заданному расписанию или событию, шпионят за содержимым буфера обмена, скрытно следить за пользователем. Вся собранная информация сохраняется на диске, а затем записывается в Log-файл – что-то вроде журнала регистрации, данные могут быть переданы по электронной почте или http/ftp-протоколу. Как заявляет разработчик виртуальной клавиатуры Oxynger KeySheild, программа обезопасит пользователя от шпионского программного обеспечения и троянов за счет запрета регистрации, входа в личный кабинет и тому подобного ввода критически важных данных при помощи клавиатуры, мыши, буфера обмена и экранных действий. Специальная функция блокирования скриншотов дает возможность отменять экранные снимки на таких уязвимых для информации этапах. Еще один интересный факт об Oxynger KeySheild – возможность полностью осуществлять ввод данных без помощи мыши. Для каждой инсталляции своего экземпляра программа использует различные раскладки клавиатуры. Поскольку стандартная раскладка QWERTY давно уже не является серьезным барьером для киберпреступников, Oxynger предоставляет уникальную организацию клавиш для каждой отдельной копии программного обеспечения, что в итоге оборачивается большой головной болью для хакеров.

Разумеется, приложение Oxynger KeySheild не предназначено для ежедневного использования в рамках работы с большими объемами данных на клавиатуре. Программа нацелена на ситуации, когда пользователю требуется ввести какую-либо конфиденциальную информацию: номер карты или счета, к примеру.

Подобрать пароли от различных аккаунтов можно с помощью фишинга — это популярный способ получения информации от беспечных пользователей. Хакеры присылают письма, которые по внешнему виду очень похожи на настоящие ресурсы и просят пользователя якобы заново ввести

логин и пароль к своей учетной записи. Еще один пример фишинговых писем — это маскировка "зловредов" под отправленные документы. Правда никаких документов на самом деле в письме нет, а вместо документов в письмо вложена GIF-картинка со ссылкой на сайт для выманивания паролей.

Открывает список необычных компьютерных вирусов червь P2PShared.U. Этот вредоносный код распространялся с помощью почтовых сообщений с темой: "McDonalds желает Вам счастливого Рождества!". В сообщении речь шла о купоне, дающем право на бесплатный обед в McDonald’s. Но именно купон и являлся носителем вредоносного кода.

BatGen.D, испанский шеф-повар. Этот вредоносный код – специалист в области приготовления вредоносных закусок на любой вкус. Он попадает на компьютеры в виде файла под названием “personalcake.bat”. На деле он представляет собой инструмент для создания вредоносного программного обеспечения, который затем спрашивает у пользователя, каким именем он бы хотел назвать творение.

На седьмом месте расположился вредоносный код Aidreden.A. После заражения компьютера на экран выводится сообщение: "Вы умрете в следующем месяце". Таким образом Aidreden.A пытается убедить пользователя в том, что его компьютер якобы заражен вирусами, после чего предлагает загрузить фальшивый антивирус. Представляет собой обыкновенное окошко с кнопкой ОК. Потом открывает страницу Microsoft Security Center и предлагает скачать антивирус.

Все современные вирусы не пишутся наобум, нужно написать такой, который хорошо прячется, модифицируется, еще лучше если дальше распространяется и умеет выполнять какую-то работу, а не просто спалить видеокарту.

Поэтому при ответе на вопрос, на каком языке написать вирус, правильным ответом будет: на том, котором знаете. Ведь что такое вирус? Это программа, которая работает с определенными тонкостями системы. На каком языке можно сделать программу - теоретически на любом. Есть тонкости: что-то на одном проще, что-то на другом, вот и вся разница. Не рекомендуется писать на Ассемблере, так как тяжело получить доступ к каким-либо API.

DoS-атаки

DoS (от англ. Denial of Service — отказ в обслуживании) — хакерская атака на вычислительную систему с целью довести её до отказа, то есть создание таких условий, при которых легальные пользователи системы не могут получить доступ к предоставляемым системным ресурсам (серверам), либо этот доступ затруднён. Если атака выполняется одновременно с большого числа компьютеров, говорят о DDoS-атаке (от англ. Distributed Denial of Service, распределённая атака типа "отказ в обслуживании"). В настоящее время DoS и DDoS-атаки наиболее популярны, так как позволяют довести до отказа практически любую систему, не оставляя юридически значимых улик. Последняя проводится в том случае, если требуется вызвать отказ в обслуживании хорошо защищённой крупной компании или правительственной организации. Особенностью данного вида компьютерного преступления является то, что злоумышленники не ставят своей целью незаконное проникновение в защищенную компьютерную систему с целью кражи или уничтожения информации. Цель данной атаки - парализовать работу атакуемого веб-узла.

Схематически DDoS-атака выглядит примерно так: на выбранный в качестве жертвы сервер обрушивается огромное количество ложных запросов со множества компьютеров с разных концов света. В результате сервер тратит все свои ресурсы на обслуживание этих запросов и становится практически недоступным для обычных пользователей. Циничность ситуации заключается в том, что пользователи компьютеров, с которых направляются ложные запросы, могут даже не подозревать о том, что их машина используется хакерами. Программы, установленные злоумышленниками на этих компьютерах, принято называть "зомби". Известно множество путей "зомбирования" компьютеров: от проникновения в незащищенные сети, до использования программ-троянцев. Пожалуй, этот подготовительный этап является для злоумышленника наиболее трудоемким. Чаще всего злоумышленники при проведении DDoS-атак используют трехуровневую архитектуру, которую называют "кластер DDoS". Такая иерархическая структура содержит:

- управляющую консоль (их может быть несколько), т.е. именно тот компьютер, с которого злоумышленник подает сигнал о начале атаки;
- главные компьютеры. Это те машины, которые получают сигнал об атаке с управляющей консоли и передают его агентам-"зомби". На одну управляющую консоль в зависимости от масштабности атаки может приходиться до нескольких сотен главных компьютеров;
- агенты - непосредственно сами "зомбированные" компьютеры, своими запросами атакующие узел-мишень.

Другая опасность DDoS заключается в том, что злоумышленникам не нужно обладать какими-то специальными знаниями и ресурсами. Программы для проведения атак свободно распространяются в Сети. Для защиты от сетевых атак применяется ряд фильтров, подключенных к интернет-каналу с большой пропускной способностью. Фильтры действуют таким образом, что последовательно анализируют проходящий трафик, выявляя нестандартную сетевую активность и ошибки. В число анализируемых шаблонов нестандартного трафика входят все известные на сегодняшний день методы атак, в том числе реализуемые и при помощи распределённых бот-сетей.

Защита от основных видов DoS-атак

В основном защита от DoS-атак строится на правильной настройке компьютера. Следующие меры защиты способны защитить лишь от слабых DoS-атак, либо они будут использоваться в качестве снижения её эффективности.

Также есть несколько универсальных советов, которые помогут подготовить систему к DoS-атаке:

- Все серверы, которые имеют доступ во внешнюю сеть, должны быть подготовлены к удаленной аварийной перезагрузке. Также желательно наличие второго сетевого интерфейса, через который по ssh-соединению можно быстро получить доступ к серверу.
- Программное обеспечение, которое установлено на сервере, должно быть в актуальном состоянии, а именно: должно быть установлено последнее ПО, касающееся обеспечения безопасности системы.
- Все сетевые сервисы должны быть защищены брандмауэром.

Полностью защититься от DDoS-атак на сегодняшний день невозможно, так как совершенно надёжных систем не существует. Здесь также большую роль играет человеческий фактор, потому

что любая ошибка системного администратора, неправильно настроившего маршрутизатор, может привести к весьма плачевным последствиям. Однако, несмотря на всё это, на настоящий момент существует масса как аппаратно-программных средств защиты, так и организационных методов противостояния.

Хакерство

Как ни крути, а информационная безопасность у многих прочно ассоциируется с хакерами. В наши дни под хакером понимается злоумышленник, который делает что-то нелегальное, взламывает какие-то системы с материальной выгодой для себя. Но это далеко не всегда было так.

Вернемся на полвека назад, в 1960-е годы, когда ЭВМ постепенно стали проникать в нашу жизнь. Хакерство началось еще тогда, с попыток использовать технику не по назначению. Например, чтобы запустить на ней нами же написанную игру. В те времена понятие "хакер" — это очень увлеченный человек, пытающийся сделать с системой что-то нестандартное. Доступ к компьютерам ведь был в основном у сотрудников университетов, причем процессорного времени каждому выделялось не так уж много. Позволялось поработать с вычислительной машиной всего несколько часов в неделю, по строгому расписанию. Но даже в таких условиях людям удавалось выкраивать время на эксперименты. Хакерам того времени было интересно не просто решить какую-то вычислительную задачу, они хотели понять, как устроена и работает машина. Культура хакерства вышла из очень увлеченных людей.

В 70-е хакерство окончательно сформировалось как попытка "поиграть" с информационной системой, обходя ее ограничения. Интернета тогда еще не было, но уже была телефония. Поэтому появилось такое явление как фрикинг (сленговое выражение, обозначающее взлом телефонных автоматов, телефонных сетей и сетей мобильной связи, с использованием скрытых от пользователя или недокументированных функций. Обычно фрикинг осуществляетс для бесплатных звонков, пополнения личного мобильного счета.). Отцом фрикинга считается Джон Дрейпер, известный под ником "Капитан Кранч". Однажды он нашел в пачке кукурузных хлопьев "Captain Crunch" подарок — свисток. Телефонные линии в то время были аналоговыми и телефонные аппараты общались друг с другом посредством обмена тоновыми сигналами. Оказалось, что тональность обнаруженного Дрейпером свистка совпадает с тоном, используемым телефонным оборудованием для передачи команд. Фрикеры стали при помощи свистков эмулировать систему команд и бесплатно дозванивались из уличных телефонов в соседние города и штаты. Следующим шагом стало создание Стивеном Возняком и Стивом Джобсом "blue box" — аппарата, эмулировавшего все те же тоновые команды. Он позволял не только дозваниваться до нужных номеров, но и пользоваться секретными служебными линиями. Подав сигнал на частоте 2600 Гц можно было перевести телефонные системы в административный режим и дозвониться на недоступные обычному человеку номера, например, в Белый Дом. Развлечение продолжалось до конца 80-х, когда в популярной газете была опубликована большая статься о "blue box", которая привлекла к фрикерам внимание полиции. Многие фрикеры, в том числе и сам Дрейпер, были арестованы. Позже все же удалось разобраться, что делали они это все не ради денег, а скорее из спортивного интереса и баловства. В то время в уголовном кодексе просто не существовало никаких статей, относящихся к мошенничеству с информационными системами, так что вскоре все фрикеры были отпущены на свободу.

В 80-е годы слово "хакер" впервые получило негативный оттенок. В сознании людей уже стал формироваться образ хакера как человека, который может делать что-то незаконное для получения прибыли.

Кевин Митник

Как и его предшественники, Кевин Митник начал заниматься хакерством, вторгаясь в телефонные линии. В 1981 году семнадцатилетний Кевин со своим другом взломал телефонную станцию Computer System for Mainframe Operations (COSMOS), принадлежащую компании Pacific Bell в Лос-Анджелесе. Вторгшись в систему, он переключал телефонные линии и перехватывал все звонки, идущие через эту станцию. Абоненты вскоре начали жаловаться, списывая это на ошибки и розыгрыши операторов. Кевин Митник, разумеется, отвечал на звонки сам, иногда при этом отмачивая бестактные шутки.

Но Митник на этом не остановился: он продолжал влезать в систему компании Pacific Bell и её COSMOS. Хакер сумел войти в базу данных компании и украсть информацию о нескольких абонентах. Он с лёгкостью получил доступ к телефонным счетам, паролям, комбинациям шлюзов и даже к системному руководству. Митник даже использовал переключение телефонных линий для своих личных нужд. В конце концов, один из сотрудников Pacific Bell обнаружил нарушения в системе COSMOS. Было начато расследование, которое быстро привело к телефонной будке, используемой Кевином Митником для звонков и доступа к сети; оставалось только дождаться, когда появится преступник и поймать его с поличным. Приговор суда был мягким: Митника обвинили в порче данных и краже и приговорили к трём месяцам отбывания наказания в исправительном учреждении и дали один год испытательного срока.

Самый большой успех пришёл к Кевину Митнику в 1983 году, когда он совершил поистине впечатляющее деяние. В то время он был студентом южнокалифорнийского университета. Используя один из университетских компьютеров, возможно, TRS-80 с 1,77-МГц процессором Zilog, Митник проник в глобальную сеть ARPANet, являющуюся предшественницей Internet, которая в то время была предназначена для военных целей и объединяла крупные корпорации и университеты. Проникнув в эту сеть, Митник добрался до самых защищённых компьютеров того времени, до компьютеров Пентагона. Он получил доступ ко всем файлам Министерства обороны США. В то время не было никаких следов кражи информации или порчи: Митник действовал просто из любопытства и проверял свои способности. Но один из системных администраторов обнаружил акт вторжения и поднял тревогу. Расследование выявило автора атаки, и Кевина Митника арестовали прямо на территории университета. Он был осуждён и отбыл своё первое настоящее наказание за незаконное вторжение в компьютерную систему, проведя полгода в исправительном центре для молодёжи.

В 1987 году Митник оставил свою незаконную деятельность. Кевин находился на испытательном сроке, в соответствии с последним приговором суда, поэтому он не мог позволить себе никаких правонарушений. Однако вскоре Митник снова был замешан в тёмных делах.

Однажды вечером вместе со своим другом Ленни ДиСикко Митник вторгся во внутреннюю сеть исследовательской лаборатории американской компьютерной компании Digital Equipment Corporation (DEC). Для Митника сделать это было несложно, так как ДиСикко являлся сотрудником данной лаборатории и одновременно был соучастником взлома. EasyNet - внутренняя сеть DEC - не выдержала хакерской атаки, и вскоре сообщники получили доступ ко всей системе.

Как и в случае с предыдущими атаками, вторжение в лабораторию было быстро обнаружено, но на этот раз Митник это предвидел и подготовился. Он зашифровал источник вызовов, сделав бесполезными все попытки выследить его. И на этот раз Митник взломал систему не из простого любопытства или для проверки своих способностей: у него было другая цель. Хакер хотел завладеть исходным кодом операционной системы VMS, разработанной компанией DEC для компьютеров VAX.

Митник предпринял все меры предосторожности, но не учёл одного - своего собственного друга. Кевин любил игры и розыгрыши, поэтому как-то раз он позвонил работодателю своего друга Ленни ДиСикко, притворившись представителем государственной власти. Он сказал, будто у одного из его служащих (ДиСикко) проблемы с налогами. ДиСикко не оценил такую шутку и решил отомстить по-своему.

ДиСикко предал Митника, сообщив своему работодателю о вторжении Митника в сеть компании. Затем он связался с ФБР и сказал, что может сдать хакера, который регулярно проникал в сеть лаборатории. Во время встречи Митника с ДиСикко хакер попал в ловушку, расставленную его собственным другом, который пришёл в сопровождении двух агентов ФБР. Митник был арестован.

Судебное дело длилось недолго: компания DEC обвинила хакера в краже информации и запросила за нанесённый ущерб более $200 000. Митника приговорили к одному году тюремного заключения, кроме того, он должен был посещать шестимесячные курсы для излечения от компьютерной зависимости.

В 1994 году Кевин Митник вернулся к своей незаконной деятельности, его разыскивало ФБР. За свои "подвиги" Митник уже прославился на весь мир. Федералы разослали повсюду его фотографии, чтобы узнавшие его люди могли позвонить властям. В течение этого и следующего года на Кевина Митника была объявлена облава, самая впечатляющая за всю историю поимки хакеров.

Митник решил атаковать другого хакера и эксперта по компьютерной безопасности - Тсутому Шимомуру (Tsutomu Shimomura). Он хорошо подготовил свою атаку и, чтобы убедиться, что никто ему не помешает, запустил её в Рождество, 25 декабря 1994 года. Митник взломал личный компьютер Шимомуры, используя неизвестную в то время методику - подмену IP-адреса, то есть разновидность вторжения, при котором взломщик пытается замаскироваться под другую систему, используя её IP-адрес.

Однако Митника подвёл брандмауэр Шимомуры, который записывал все действия, происходящие на целевой машине. На следующий же день, 26 декабря, Шимомуре позвонил один из его коллег и сообщил, что его компьютер стал жертвой

вторжения. Шимомура быстро вычислил Митника и решил помочь ФБР в поимке хакера, используя свои же хакерские умения.

Для поимки Митника ФБР предоставило Шимомуре полную свободу действий, в том числе разрешило использовать хакерство. Облава превратилась в виртуальную охоту. Однажды Шимомура сообщил, например, что он застал Митника врасплох 17 января 1995 года, когда тот влез в сеть, принадлежащую Motorola, чтобы украсть секретное программное обеспечение компании.

Преследование усиливалось, и спецслужбы начали приближаться к киберпреступнику, отступающему в г. Роли, штат Северная Каролина. Чтобы засечь сотовый телефон, с которого Митник совершал свои атаки, Шимомура в течение двух дней бродил по улицам Роли с детектором связи. 15 февраля 1995 года в 2 часа ночи агенты ФБР вместе с Шимомурой ворвались в квартиру Митника. Когда знаменитый хакер увидел своего соперника, он воскликнул: "Hi, Tsutomu! Мои поздравления!". После почти двухлетнего преследования Митника приговорили к пяти годам тюремного заключения. На то время это было самое суровое наказание для хакера. Ныне является консультантом по компьютерной безопасности.

Роберт Таппан Моррис

Роберт Таппан Моррис (Robert Tappan Morris), который сейчас является штатным профессором в Массачусетском технологическом институте (МТИ) в лаборатории компьютерных наук и искусственного интеллекта, создал первого "интернет-червя".

И опять же, к созданию "червя" Морриса побудило любопытство. По его словам, основной целью его программы было оценить истинные размеры сети Интернет, т.е. определить количество подключённых к ней компьютеров. В то время к Интернету было подключено не так много машин, но вопреки расчётам Морриса, его "червь" причинил гораздо больше вреда, чем ожидалось.

"Червь" Морриса, отправленный с компьютеров МТИ, был запрограммирован на то, чтобы проверить компьютер и скопировать себя в систему, если машина ещё не была инфицирована. Проблемы начались, когда Моррису пришла в голову мысль, что некоторые системные администраторы могут придумать уловку с поддельными копиями, чтобы обмануть "червя" и заставить его думать, что компьютер уже заражён. Тогда он модифицировал код программы, чтобы "червь" оставлял свои копии каждый раз, независимо от того, инфицирован компьютер или нет.

"Червь" Морриса распространился, как пожар, заразив несколько тысяч компьютеров всего за несколько часов. Приблизительно подсчитали, что восстановление каждой инфицированной системы будет стоить от $200 до $53 000, в зависимости от компьютера. Чтобы остановить "червя", были мобилизованы различные команды программистов, для нейтрализации атаки понадобилось несколько дней.

Роберт Таппан Моррис был признан виновным в компьютерном мошенничестве и был приговорён к трём годам условно, 400 часам общественных работ и $10 050 штрафа.

Кевин Поулсен

Кевин Поулсен - это ещё одно имя, которое вошло в анналы ФБР в 80-е гг. Впервые его арестовали в 1989 году, когда ему было 24 года. Тогда его обвинили в нескольких взломах телефонных сетей и компьютерных серверов, против него были собраны разные уличающие доказательства. Но когда пришло время предстать перед судом, Поулсен решил скрыться, агенты ФБР 17 месяцев выслеживали его. Именно в это время он совершил своё самое знаменитое хакерское деяние. В прямом эфире на лос-анджелесской радиостанции KIIS-FM проходила викторина-розыгрыш. Слушателям предлагалось позвонить на радиостанцию и попытаться выиграть автомобиль Porsche 944 S2. Приз должен был достаться 102-му дозвонившемуся радиослушателю. Кевин Поулсен начал действовать: он так заблокировал телефонную сеть, что на радио никто не мог дозвониться, кроме него самого. Следовательно, Кевин стал 102-м дозвонившимся и выиграл приз. Поулсен "утёр нос" ФБР, заставив власти вновь приступить к его поиску, потому что после этого он исчез.

В апреле 1991 года ФБР, наконец, удалось арестовать Поулсена. Поимке поспособствовал анонимный донос: кто-то сообщил властям, что Поулсен делает покупки в супермаркете на окраине Лос-Анджелеса. В 1994 году ему предъявили обвинение и в результате приговорили к четырём годам тюремного заключения. В то время это был самый суровый приговор, назначенный судом за хакерство.

В 1983 году на обложке Newsweek появился один из основателей хакерской группы "414s", ставшей известной благодаря серии взломов серьезных компьютерных систем. У хакеров появляются и собственные журналы, и другие способы обмена информации. В эти же годы власти западных стран начинают формировать законы, связанные с компьютерной безопасностью. Впрочем, масштаб, конечно, не идет ни в какое сравнение с сегодняшним днем. Так, много шума случилось вокруг дела Кевина Поулсена, который, благодаря своим фрикерским навыкам, первым дозвонился в эфир радиостанции KIIS-FM, что позволило ему выиграть автомобиль Porsche в рамках проходящего конкурса. По сравнению с современными атаками, которые позволяют уводить со счетов миллионы долларов — это просто мелочь, но тогда эта история вызвала очень большой резонанс.

Девяностые – это эра активного развития интернета, в это же время за хакерством окончательно закрепляется криминальный оттенок. Персональные компьютеры уже стали относительно доступны простым людям, при этом о безопасности никто особенно не задумывался. Появлялось огромное количество готовых программ, с помощью которых можно было взламывать пользовательские компьютеры, не имея каких-то серьезных технических навыков и способностей. Чего стоит только известная программа winnuke, которая позволяла отправить Windows 95/98 в "синий экран" путем отправки одного-единственного IP-пакета.

Владимир Левин

Люди занимаются хакерством не только из любопытства и азарта, иногда в дело замешаны деньги. Самый яркий пример - ограбление банка с целью получения денег, иногда удавалось украсть миллионы. Так, Владимир Левин (Vladimir Levin) завоевал дурную славу тем, что украл несколько миллионов долларов при странных обстоятельствах.

В 1994 году Левин проник во внутреннюю сеть американского банка Citibank, взломав аналоговое модемное подключение банка и получив доступ к нескольким счетам. Он сумел перевести 10,7 миллиона долларов на счета в США, Финляндию, Германию, Израиль и Нидерланды. Левину помогали трое сообщников, которые должны были вернуть украденные деньги.

Однако его сообщников арестовали, когда они пытались стащить украденные деньги. Их допрос вывел на след Левина, который работал программистом в Санкт-Петербурге. Российского хакера арестовали в марте 1995 года в лондонском аэропорту Хитроу. Судебное разбирательство против него началось только в сентябре 1997 года и закончились в феврале следующего года. Левина приговорили к трём годам лишения свободы.

Вообще, девяностые считаются золотыми годами темных хакеров: возможностей для мошенничества полно, компьютерные системы соединены через интернет, а серьезных механизмов безопасности в массовых ОС в эти годы еще нет. Еще одна отличительная особенность девяностых — огромное количество голливудских фильмов про хакеров как иллюстрация того факта, что взлом компьютерных систем постепенно становится "обыденной диковинкой" для широких масс. Ведь почти в каждом таком фильме обязательно фигурировал какой-нибудь вирус, который взрывал мониторы. Может быть, именно из-за киноиндустрии у пользователей в голове сложился стереотип "безопасность — это вирусы", что, конечно, помогло многим антивирусным компаниям сделать свое состояние.

В 2002 году Билл Гейтс написал своим сотрудникам в компании Microsoft письмо о том, что ситуацию нужно исправлять, и пора начинать разрабатывать программное обеспечение с оглядкой на безопасность. Данная инициатива получила название "Trustworthy computing", и развивается она до сих пор. Начиная с Windows Vista, эта идея начала воплощаться в жизнь. Количество уязвимостей в ОС от Microsoft заметно снизилось, эксплойты (компьютерная программа, фрагмент программного кода или последовательность команд, использующие уязвимости в программном обеспечении и применяемые для проведения атаки на вычислительную систему) под них все реже появляются в публичном доступе. Как ни удивительно это звучит, но с точки зрения подхода к безопасности последние версии Windows гораздо надежнее других распространенных операционных систем. Так, в OS X только в последнее время стали появляться механизмы, которые затрудняют эксплуатацию уязвимостей.

Нулевые годы нашего века. Цифровой криминал выходит на новый уровень. Каналы связи стали толще, стало проще совершать мощные DoS-атаки. Никто уже не занимается взломом компьютерных систем просто ради удовольствия, это многомиллиардный бизнес. Расцветают ботнеты: огромные системы, состоящие из миллионов зараженных компьютеров. Другой характерной особенностью последнего десятилетия является тот факт, что фокус атакующего сместился на пользователя, на его персональный компьютер. Системы становятся сложнее, современный браузер — это уже не просто программа, которая умеет рендерить HTML, показывать текст и картинки. Это очень сложный механизм, полноценное окно в интернет. Почти никто уже не пользуется отдельными мессенджерами и почтовыми клиентами, все взаимодействие с интернетом происходит именно через браузер. Неудивительно, что один из основных методов заражения пользователей в наш дни — drive-by-downloads — происходит как раз при помощи браузера. Конечно, в современных браузерах стали появляться механизмы борьбы с этим,

пользователей стараются предупреждать, что посещаемый сайт может быть опасен. Но браузер по-прежнему остается одним из основных посредников при заражении пользовательских машин. Еще один большой канал распространения вредоносных программ — мобильные устройства. Если в официальных магазинах приложений программы хоть как-то проверяют на вредоносность, то из неофициальных источников можно занести на свой девайс практически все, что угодно. Да и вообще безопасность мобильных устройств сейчас — довольно молодая и немного сумбурная отрасль, что связано с той скоростью, с которой современные мобильные платформы ворвались в нашу жизнь.

Андриан Ламо

Адриан Ламо, безусловно, свёл с ума самое большое количество сетевых администраторов. От его деятельности пострадали такие крупные корпорации, как Microsoft, New York Times, AOL, Sun Microsystems, Yahoo!, MacDonald's и Cingular. Ему приписывают все виды атак и нарушений защиты корпоративных систем безопасности. Ламо обходил защитные системы с обескураживающей простотой. Так, во время эфира ночных новостей телекомпании NBC журналист предложил Адриану доказать свой талант прямо перед объективом камеры, и тогда хакер менее чем за пять минут проник во внутреннюю сеть самой телекомпании. В настоящее время Ламо является специалистом по безопасности и наслаждается полной свободой передвижения после того, как многие годы был под наблюдением властей США.

12 февраля 2004 года был самый обычный день, но в компании Microsoft было объявлено чрезвычайное положение. Кто-то украл исходный код операционной системы Windows 2000, которой до сих пор пользуется большое количество пользователей. И что ещё хуже, неизвестный хакер выложил этот код в Wild. Кража была масштабна: 600 миллионов байт данных, 30 195 файлов и 13,5 миллиона строк кода. Утечка информации коснулась операционной системы Windows 2000 и её "старшей сестры" Windows NT4. Все сотрудники "софтового" гиганта пытались выяснить, что произошло, но никто не мог дать ответ. Данные были украдены прямо из сети Microsoft. Неизвестный хакер вошёл во внутреннюю сеть компании, взломав пароль одного из компьютеров. Исходный код быстро распространился по Интернету, особенно по P2P-сетям. К счастью, несмотря на то, что все опасались худшего, последствия этой грандиозной кражи оказались довольно мягкими.

Ответственность

На сегодняшний день за "хакерство" можно получить наказания по статьям 212 (Хищение путем использования компьютерной техники), 354 (Разработка, использование либо распространение вредоносных программ).

Статья 212. Хищение путем использования компьютерной техники

1. Хищение имущества путем изменения информации, обрабатываемой в компьютерной системе, хранящейся на машинных носителях или передаваемой по сетям передачи данных, либо путем введения в компьютерную систему ложной информации - наказывается штрафом, или лишением права занимать определенные должности или заниматься определенной деятельностью, или арестом на срок до

шести месяцев, или ограничением свободы на срок до трех лет, или лишением свободы на тот же срок.

2. То же деяние, совершенное повторно, либо группой лиц по предварительному сговору, либо сопряженное с несанкционированным доступом к компьютерной информации, - наказывается ограничением свободы на срок от двух до пяти лет или лишением свободы на срок до пяти лет с лишением права занимать определенные должности или заниматься определенной деятельностью или без лишения.

3. Деяния, предусмотренные частями первой или второй настоящей статьи, совершенные в крупном размере, - наказываются лишением свободы на срок от трех до десяти лет с конфискацией имущества или без конфискации и с лишением права занимать определенные должности или заниматься определенной деятельностью или без лишения.

4. Деяния, предусмотренные частями первой, второй или третьей настоящей статьи, совершенные организованной группой либо в особо крупном размере, - наказываются лишением свободы на срок от шести до пятнадцати лет с конфискацией имущества и с лишением права занимать определенные должности или заниматься определенной деятельностью или без лишения.

Статья 354. Разработка, использование либо распространение вредоносных программ

1. Разработка компьютерных программ или внесение изменений в существующие программы с целью несанкционированного уничтожения, блокирования, модификации или копирования информации, хранящейся в компьютерной системе, сети или на машинных носителях, либо разработка специальных вирусных программ, либо заведомое их использование, либо распространение носителей с такими программами - наказываются штрафом, или арестом на срок от трех до шести месяцев, или ограничением свободы на срок до двух лет, или лишением свободы на тот же срок.

2. Те же действия, повлекшие тяжкие последствия, - наказываются лишением свободы на срок от трех до десяти лет.

За всю историю самое жестокое наказание за хакерство было в 1998-1999 годах в Китае, когда два брата подверглись смертной казни за взлом компьютерной системы государственного банка с целью кражи денег. Один из братьев получил отсрочку исполнения приговора за согласие дать свидетельские показания.

Не всегда же хакерство носит негативный оттенок. Известны случаи, когда можно законно зарабатывать, будучи хакером. Ведь некоторые из них помогают защитить наши компьютеры и гаджеты от кражи информации и взломов. Весной 2015 года особо отличился хакер Джун Хун Ли из Южной Кореи. Участвовал в конкурсе Pwn2Own, который спонсирует корпорация Google. В ходе этого мероприятия хакеры со всего мира демонстрируют конкурсному жюри обнаруженные ими баги, уязвимости и эксплойты в различных приложениях и операционных системах. Разумеется, за каждый обнаруженный баг хакер получает награду от организаторов. Джун Хун Ли выявлял одну уязвимость за другой. Сначала он сломал браузер Chrome. Затем продемонстрировал, как с помощью этого же браузера можно организовать широкомасштабную атаку на операционную

систему. Потом взломал браузер Apple. Заработав в общей сложности 225 000 долларов за два дня конкурса, Джун Хун Ли наконец расслабился и перевёл дыхание.

Подведем итог о том, что такое хакерство сегодня, и чего стоит ожидать в ближайшем будущем. Хакерство возникло в 70-е гг. прошлого века, но некоторые из многочисленных группировок, сформировавшихся вокруг этого движения, существуют и по сей день. Всё больше и больше людей получают возможность пользоваться Интернетом, а хакеров и так называемых "script kiddies" (компьютерных хулиганов-подростков) сейчас больше, чем когда-либо. Однако мы не наблюдаем бурного роста крупномасштабных атак против компьютерных систем, который был бы соразмерен увеличению числа хакеров. Движение пошло немного по другому направлению, по сравнению с периодом расцвета хакерства. Некоторые атаки по-прежнему производят фурор, и системным администраторам приходится за это расплачиваться. Кевины Митники и Джоны Дрейперы, которые вошли в историю, уже успокоились и остепенились, а нынешние специалисты по безопасности компьютерных систем сталкиваются с менее опасной, но гораздо более массовой угрозой. В начале двухтысячных, если обнаруживалась какая-нибудь уязвимость в Windows, практически сразу в свободном доступе появлялся эксплойт, который позволял получить контроль над пользовательским компьютером. Тогда над извлечением прибыли от уязвимостей практически не задумывались. Конечно, были программы, которые воровали данные пользователей, уводили компьютеры в ботнеты, но сами уязвимости, приводящие к компрометации компьютера, было относительно просто эксплуатировать, а значит, написать эксплоит. Примерно последние пять лет найти в публичном доступе рабочий эксплоит для недавно обнаруженной уязвимости стало очень непросто. Теперь это огромный бизнес. Ведь написать эксплоит для уязвимости в системе, в которой внедрены механизмы вроде DEP и ASLR, значительно сложнее. Последние годы также показывают, что всех нас ждут проблемы безопасности так называемого "интернета вещей". Компьютеры с доступом в интернет сейчас все чаще в том или ином виде присутствуют в самых разнообразных бытовых и медицинских приборах, а также автомобилях. И уязвимости в них точно такие же, как в обычных компьютерах. Исследования по взлому привычных нам вещей становятся популярной темой на ведущих мировых конференциях по безопасности. Ведь если злоумышленники начнут пользоваться такими уязвимостями, это будет представлять серьезную опасность для здоровья и жизни пользователей. Тем важнее становится роль специалиста по безопасности.

Антивирусы

Приложение

Список вопросов, возникнувших в ходе обсуждения

IT-проект

IT-проект и его структура

- Каким образом заказчик может вносить изменения в проект? (Белый А.А.)
- Чем отличается куратор проекта от менеджера? (Белый А.А.)
- Какими навыками должен обладать менеджер? (Белый А.А.)
- Что обязана делать группа людей, занимающаяся внедрением готового продукта? (Борисевич П.И.)
- Кто контролирует аудит проекта? (Гетьман С.И.)

- Как планируется бюджет проекта, и кто является инвестором? (Гетьман С.И.)
- Что означает "развертывание IT-инфраструктуры"? (Гетьман С.И.)
- Кто и зачем занимается документацией проекта? (Гетьман С.И.)
- Как влияет изменение состава исполнителей на развитие проекта? (Гетьман С.И.)
- Кто такие подрядчики и субподрядчики, их роль и структура? (Григорьев А.В.)
- Если какие-нибудь дополнения к схеме участников проекта? (Грушевский А.А.)
- Отличия open source проекта от коммерческого? (Грушевский А.А.)
- Какая роль на проекте является самой важной, а какая самой незначительной? (Ипатов А.Е.)
- Какие роли требуют высокой квалификации? (Ипатов А.Е.)
- Какие существуют части планирования? (Ипатов А.Е.)
- Какие роли могут совмещаться, а какие нет? (Ипатов А.Е.)
- Как организуется взаимодействие между командой и руководством? (Лебедев Н.А.)
- Бизнес-аналитики в структуре проекта? (Лебедев Н.А.)
- Какова роль менеджера проекта? (Лебедев Н.А.)
- Какова роль руководителей компании в структуре проекта? (Лебедев Н.А.)
- Самая ответственная должность на проекте? (Лебедев Н.А.)
- В чем заключается принципиальное различие в работе куратора и руководителя проекта? (Михальцова А.Ю.)
- Что означает фраза "проект структурирован"? (Михальцова А.Ю.)
- Какими способами можно контролировать выполнение проекта? (Ровдо Д.И.)
- Какие должности в структуре проекта обязательны, а какие нет? (Ровдо Д.И.)
- Насколько важную роль в проекте играют подрядчики? (Ровдо Д.И.)
- Как можно разрешить ситуацию, когда кто-то из структуры (руководство, исполнители) не может выполнять свою работу из-за нехватки знаний, опыта? (Трубач Г.Г.)
- Как происходит общение бизнес-аналитика с заказчиком? (Трубач Г.Г.)
- Как разработчики получают прибыль из open source проектов? (Трубач Г.Г.)
- Какова структура open source и фриланс проектов? (Трубач Г.Г.)
- Есть ли отличия между структурами IT-проектов в Беларуси и проектов других стран? Если есть, то какие? (Ярошевич Я.О.)
- Как осуществляется взаимодействие в команде между собой, если проект является интернациональным? (Ярошевич Я.О.)
- Как происходит выбор подходящей команды для проекта? (Ярошевич Я.О.)
- Как организуется структура проекта, создающийся не в IT –компании? (Ярошевич Я.О.)

Жизненный цикл IT-проекта и фазы разработки ПО

- На каком этапе реализуется поддержка проекта? (Белый А.А.)
- Существуют ли отличия между жизненным циклом обычного проекта и IT-проекта? (Белый А.А.)
- Как осуществляется интеграция проекта? (Белый А.А.)
- С какой целью созданы международные стандарты жизненного цикла и в каких случаях их требуют к выполнению? (Борисевич П.И.)
- Насколько тщательно подходят к планированию жизненного цикла? (Гетьман С.И.)

- Как справиться с постоянным обновлением и улучшением различных платформ разработки? (Гетьман С.И.)
- Как долго длится сопровождение продукта? (Гетьман С.И.)
- Влияет ли бюджет проекта на его жизненный цикл? (Гетьман С.И.)
- Суть и назначение пост-релизного этапа разработки. (Григорьев А.В.)
- Роль планирования в процессе разработки проекта (Григорьев А.В.)
- Почему планирование проекта требует малых затрат? (Лебедев Н.А.)
- На какой стадии жизненного цикла определяется архитектура проекта? (Лебедев Н.А.)
- Что может произойти с проектом, если при планировании допущена архитектурная ошибка? (Лебедев Н.А.)
- Можно ли выпускать проект на альфа-стадии его разработки? (Михальцова А.Ю.)
- Чем отличаются стадии разработки "релиз кандидат" от "публичной реализации"? (Михальцова А.Ю.)
- Означает ли фаза "завершение", что проект уже окончательно готов к использованию? (Михальцова А.Ю.)
- Когда занимаются поддержкой проекта, если в жизненном цикле отсутствует фаза post-release? (Ровдо Д.И)
- На какой стадии лучше всего начинать тестирование продукта? (Ровдо Д.И)
- Какие существуют подэтапы этапа планирования? Какие из них являются наиболее важными? (Ровдо Д.И)
- Каков жизненный цикл open source проектов и стартапов? (Трубач Г.Г.)
- Какие существуют стандарты жизненного цикла? (Трубач Г.Г.)
- Какие существуют варианты выхода из ситуации, когда заказчик отказывается от проекта и возможен ли такой случай? (Трубач Г.Г.)
- Возможно ли возвращение от более позднего этапа жизненного цикла к более раннему? Если возможно, то по каким причинам? (Ярошевич Я.О.)
- Как соотносятся между собой этапы жизненного цикла по времени? (Ярошевич Я.О.)
- Осуществляется ли поддержка проекта после релиза и в каких случаях? (Ярошевич Я.О.)

Моделирование жизненного цикла

- Как в Scrum методологии осуществляется контроль рисков? (Белый А.А.)
- Какая методология наиболее универсальна? (Белый А.А.)
- Можно ли менять методологию в процессе разработки IT-проекта? (Борисевич П.И.)
- Когда используется модель водопада? (Борисевич П.И.)
- Кто выбирает методологию? (Гетьман С.И.)
- Какая методология наименее затратная? (Гетьман С.И.)
- Какие методологии используют наиболее успешные компании? (Гетьман С.И.)
- Какие основные отличия между спиральной и итерационной моделью? (Григорьев А.В.)
- В чем суть экстремального программирования? (Григорьев А.В.)
- Суть Scrum методологии и ее виды. (Григорьев А.В.)

- Какая модель наиболее эффективна? (Ипатов А.Е.)
- Какая методология наиболее универсальна? (Ипатов А.Е.)
- В чем заключается простота Scrum методологии? (Лебедев Н.А.)
- Как происходит переход между спринтами в Scrum методологии? (Лебедев Н.А.)
- В чем преимущества и недостатки каждой из методологий? Какая из них наиболее оптимальна? (Лебедев Н.А.)
- Каковы минусы SCRUM-а, если они есть. (Ровдо Д.И.)
- Какая модель самая оптимальная? (Ровдо Д.И.)
- Почему модель "спираль" довольно известна? Ведь, по сути, она почти не отличается от итерационной. Какие есть особенности этой модели, которые отличают ее от других? (Ровдо Д.И.)
- Какая методология сейчас используется наиболее часто? (Трубач Г.Г.)
- От чего зависит выбор той или иной модели? (Трубач Г.Г.)
- Можно ли изменять спринт при Scrum методологии? (Трубач Г.Г.)
- Чем грозит срыв спринта при Scrum методологии? (Трубач Г.Г.)
- Можно ли изменять сроки спринта при Scrum методологии? (Трубач Г.Г.)
- Каковы основные принципы экстремального программирования? Как происходит работа с заказчиком в данной методологии? (Трубач Г.Г.)
- Какая методология сейчас наиболее популярна? (Щавровский С.А.)
- Каковы особенности Scrum методологии? (Щавровский С.А.)
- Примеры использования тех или иных методологий? (Щавровский С.А.)
- Какая модель используется чаще других? (Ярошевич Я.О.)
- В чем разница между итеративной и спиральной моделью? (Ярошевич Я.О.)
- Как выбрать лучшую модель для конкретного проекта? (Ярошевич Я.О.)
- Проводят ли компании тренинги по изучению той или иной модели? (Ярошевич Я.О.)
- Какие существуют методологии, придуманные самими компаниями? (Ярошевич Я.О.)

Мозговой штурм

- Может ли мозговой штурм проводится удаленно (например, по Skype)? (Гетьман С.И.)
- Что произойдет, если при мозговом штурме, в команде находятся люди, которые друг с другом в плохих отношениях? (Гетьман С.И.)
- Как можно сорвать мозговой штурм? (Гетьман С.И.)
- Проводятся ли мозговые штурмы в крупных компаниях? (Григорьев А.В.)
- Кто оценивает и отсеивает идеи: вся группа или кто-то конкретный? (Григорьев А.В.)
- Почему 6-12 человек при мозговом штурме является оптимальным количеством? Уложится ли команда в отведенное время? (Лебедев Н.А.)
- Можно ли провести мозговой штурм с самим собой? (Лебедев Н.А.)
- Почему для мозговых штурмов так важно отсутствие критики? (Михальцова А.Ю.)
- Кто отвечает за анализ идей/решений? (Михальцова А.Ю.)
- Есть ли у мозговых штурмов чёткие временные ограничения? (Михальцова А.Ю.)
- Какого рода вопросы/проблемы решаются во время мозговых штурмов? (Михальцова А.Ю.)
- Как часто стоит устраивать мозговые штурмы? (Михальцова А.Ю.)
- Почему важнее количество высказанных идей, чем их качество? (Михальцова А.Ю.)

- Существует ли разделение ролей при мозговом штурме? (Трубач Г.Г.)
- Какова продолжительность рабочего дня при мозговом штурме? (Трубач Г.Г.)
- Часто ли используется методика мозгового штурма? (Щавровский С.А.)
- Какие известные команды используют методику мозгового штурма? (Щавровский С.А.)
- Будете ли вы продвигать методику мозгового штурма в своем рабочем окружении? (Щавровский С.А.)
- Где мозговой штурм может быть применен? (Ярошевич Я.О.)
- Как предотвратить хаос во время мозгового штурма (вечные споры, ругательство и т.д.)? (Ярошевич Я.О.)
- Как из практик мозгового штурма наиболее эффективна? (Ярошевич Я.О.)

Классификация программного обеспечения

- Что мешает использовать свободное по с GPL, для своих приложений, вместо того, чтобы платить за "снятие лицензии"? (Как именно снимается лицензия) (Белый А.А.)
- Какие альтернативен свободные ОС есть для обеспечения независимости, а также удобства для пользователя (Белый А.А.)
- Какая типовая лицензия для Open Source приложений подходит для использования в коммерческих целях? (Борисевич П.И.)
- Что могут сделать нарушителю GPL? (Борисевич П.И.)
- Как выбирать лицензию для свободного софта? (Борисевич П.И.)
- В чём особенность MIT License? В чём её основные преимущества и недостатки в сравнении с GPL? (Гетьман С.И.)
- В чём особенности лицензирования GitHub проектов? (Гетьман С.И.)
- Как свободное ПО влияет на рынок и на зарплаты программистов, менеджеров? (Гетьман С.И.)
- На какие хлеба живёт Ричард Столман? (Гетьман С.И.)
- Как Open Source / free лицензии борются с вредителями? (Гетьман С.И.)
- Насколько важен выбор лицензии для разработки ПО? Бывали ли случаи судебных тяжб из-за неправильно выбранной лицензии? (Гетьман С.И.)
- Какие существуют типы лицензий Open Source? (Ипатов А.Е.)
- На твой взгляд, есть ли перспектива развития направления Open Source? (Ипатов А.Е.)
- Как происходит тестирование Open Source проектов? (Ипатов А.Е.)
- Назови какие-нибудь популярные Open Source проекты. (Ипатов А.Е.)
- Существует ли свободное ПО, которые нарушает 4 свободы? (Лебедев Н.А.)
- В чем выгода для организации, которая работает под Open Source? (Лебедев Н.А.)
- Как сложилось, что образовались такие кластеры как Freeware, FreeSoft, ComSoft? (Лебедев Н.А.)
- Как заработать на рынке Open Source? (Лебедев Н.А.)
- На ваш взгляд, какое будущее у Open Source? (Лебедев Н.А.)
- Существуют другие лицензии, кроме GPL? Если есть, то почему нет единого стандарта? (Лебедев Н.А.)
- Какие проблемы с законом могут возникнуть при использовании Open Source? (Михальцова А.Ю.)

- Любой ли желающий может использовать Open Source? (Михальцова А.Ю.)
- Считаешь ли ты эффективным использование Open Source и почему? (Михальцова А.Ю.)
- Возможно ли в дальнейшем переведение всего ПО на открытое? (Михальцова А.Ю.)
- Как именно владельцы свободного ПО с лицензией GPL обходят ее и продают возможность пользоваться своим продуктом, не "заражая" покупателей? (Ровдо Д.И.)
- Почему все вокруг так увлечены Open Source проектами? (Ровдо Д.И.)
- В чем смысл подхода владельцев софта с открытым исходным кодом, но запретом его использовать? (Ровдо Д.И.)
- Как в проектах отслеживается тип лицензии? (Трубач Г.Г.)
- Возможно ли в открытых приложениях использовать проприетарный софт и наоборот? (Трубач Г.Г.)
- Какие самые известные Open Source проекты существуют? (Трубач Г.Г.)
- Какая ответственность предполагается за использование Open Source проектов в коммерческих целях? (Трубач Г.Г.)
- Примеры свободного программного обеспечения? (Ярошевич Я.О.)
- Можно ли отойти от одного или нескольких пунктов из десяти пунктов в определении Open Source программного обеспечения, и всё равно считать его таковым? (Ярошевич Я.О.)
- Примеры Open Source программного обеспечения? (Ярошевич Я.О.)

Менеджмент IT-проекта

Менеджмент IT-проектов

- Имеет ли место в практике, что один менеджер контролирует несколько проектов? (Белый А.А.)
- Какие существую меры контроля рисками? (Белый А.А.)
- Чем отличается работа менеджера на крупном проекте и на стартапе? (Белый А.А.)
- Какой человек может стать менеджером? (Белый А.А.)
- Можно ли рассчитывать на успех команды менеджмента проекта в новом проекте, если она хорошо справилась с предыдущим? (Борисевич П.И.)
- Как можно оценить деятельность команды менеджмента проекта? (Борисевич П.И.)
- Насколько глубокими должны быть технические знания менеджера в IT-проекте? (Григорьев А.В.)
- Может ли менеджер быть одновременно разработчиком в том же проекте? (Григорьев А.В.)
- Может ли в проекте быть одновременно несколько менеджеров? Если да, то могут ли пересекаться их сферы деятельности? (Григорьев А.В.)
- Чем отличаются полномочия менеджера от его полномочий его помощников? Где они пересекаются и в чем друг друга дополняют? (Грушевский А.А.)
- Должен ли менеджер знать некоторую техническую составляющую проекта? (Ипатов А.Е.)
- Может ли менеджер принимать решение об отказе от того или иного заказа? (Ипатов А.Е.)
- Может ли менеджер выполнять еще какие-либо роли или ему стоит сосредоточиться на своей роли? (Ипатов А.Е.)
- С помощью каких программных или аналитических средств менеджер строит бизнес-модель? (Лебедев Н.А.)

- Направление развития проекта определяется только бизнес-моделью менеджера или существуют другие определяющие факторы? (Лебедев Н.А.)
- Почему менеджер не учитывает тенденций рынка на этапе анализа? (Лебедев Н.А.)
- В чем заключается работа менеджера? (Михальцова А.Ю.)
- Разработка проекта считается непрерывным процессом? (Михальцова А.Ю.)
- Что включает check-лист? (Михальцова А.Ю.)
- Может ли заказчик выступать в качестве менеджера? (Михальцова А.Ю.)
- Какая группа успешнее выполняет проект, поставленную задачу? (Михальцова А.Ю.)
- Может ли менеджер устанавливать сроки проекта? (Михальцова А.Ю.)
- Менеджер не обязательно должен владеть техническими данными, но в его обязанности входит пункт разбиения задачи на подзадачи. Тогда получается, что у менеджера должны быть помощники? (Ровдо Д.И.)
- Какие инструменты влияния есть для обычных работников (не фрилансеров)? (Ровдо Д.И.)
- Должен ли менеджер интересоваться ситуацией и разговаривать с командой или достаточно действий командного лидера? (Ровдо Д.И.)
- Является ли проблема разногласий в команде проблемой менеджера? (Ровдо Д.И.)
- Каким образом определяются и строятся бизнес-модели проекта? (Трубач Г.Г.)
- Как происходит обсуждение вопросов менеджмента и бизнес-части проекта? (Трубач Г.Г.)
- Каковы особенности метода менеджмента проекта — делегирования? (Трубач Г.Г.)
- Чем характерен метод менеджмента — «разделяй и властвуй»? (Трубач Г.Г.)
- Полезно ли записывать (логирование) все события менеджмента (разногласия, соглашения, задачи и т.д.)? (Трубач Г.Г.)
- Почему среди перечисленных ресурсов не было ресурса времени? (Щавровский С.А.)
- Как происходит менеджмент проекта в open-source проектах? (Щавровский С.А.)
- Какие есть решения по автоматизации менеджмента проекта? (Щавровский С.А.)
- Какое образование должен иметь менеджер и его команда? (Ярошевич Я.О.)
- Кто несет наказание в случае нарушения сроков? Что произойдет, если не учтутся некоторые факторы и сроки затянутся? (Ярошевич Я.О.)
- Общается ли менеджер непосредственно с заказчиком? (Ярошевич Я.О.)
- Может ли менеджер работать удаленно? (Ярошевич Я.О.)

Роли в команде и их функции

- Кто распределяет роли в команде? (Белый А.А.)
- Кто такой проектный архитектор? (Белый А.А.)
- Могут ли совмещаться роли разработчика и QA? Применимо ли это на практике? (Белый А.А.)
- Как происходит повышение должности работника? (Белый А.А.)
- Почему нежелательно, чтобы командный лидер и менеджер проекта был одним человеком? (Борисевич П.И.)
- На каком этапе жизненного цикла распределяются роли в проекте? (Борисевич П.И.)
- Может ли заказчик сам назначать роли? (Гетьман С.И.)
- Сколько ролей оптимально использовать на проекте? (Гетьман С.И.)
- Сколько ролей может иметь один человек в команде? (Гетьман С.И.)

- С чем может быть связано изменение ролей? (Гетьман С.И.)
- Сколько ролей оптимально использовать на проекте? (Гетьман С.И.)
- В чем разница бизнес компетенции в кластере управление продуктом и кластере управления выпуском? (Гетьман С.И.)
- Кого лучше взять в команду: узкопрофильного или многопрофильного специалиста? (Гетьман С.И.)
- Может ли меняться роль работника на протяжении проекта? (Григорьев А.В.)
- Может ли один человек выполнять сразу несколько ролей? (Григорьев А.В.)
- Является ли выделение родственных функций в роли обязательным или оно носит формальный характер? (Григорьев А.В.)
- Зависит ли наличие определенных ролей от выбранной модели разработки? И от чего зависит наличие той или иной роли? (Григорьев А.В.)
- Кто занимается консультациями при разработке? (Грушевский А.А.)
- Кто помогает заказчику понимать документацию по отчетности разработки, технических возможностей проекта и т.д.? (Грушевский А.А.)
- Кому подчиняется архитектор? (Грушевский А.А.)
- Что входит в документацию пользователя? (Грушевский А.А.)
- На каком этапе происходит распределение ролей? (Ипатов А.Е.)
- На всех ли проектах у одной роли одинаковые функции? (Ипатов А.Е.)
- Какие роли являются ключевыми, а какие нет? (Ипатов А.Е.)
- Чем отличается кластер управление программой от кластера управление продуктом? (Лебедев Н.А.)
- Как командный лидер оценивает компетентность своей команды? (Лебедев Н.А.)
- Какие роли можно совмещать? (Лебедев Н.А.)
- Какая ситуация наиболее приемлема: сотрудник работает по конкретному плану или сотрудник сам определяет свои обязанности? (Лебедев Н.А.)
- Самые важные роли, с которыми можно начинать разработку? (Лебедев Н.А.)
- Кто распределяет роли в команде? (Михальцова А.Ю.)
- Четко ли соблюдается распределение ролей в команде или возможны совмещения? (Михальцова А.Ю.)
- По каким принципам распределяются роли в различных моделях жизненного цикла? (Михальцова А.Ю.)
- Участвует ли командный лидер в непосредственной разработке проекта? (Михальцова А.Ю.)
- Если командный лидер разъясняет требования заказчика разработчикам, то каковы обязанности бизнес аналитика? (Михальцова А.Ю.)
- Какие роли совмещать нежелательно? (Михальцова А.Ю.)
- Какая из ролей находится выше, а как ниже в структуре проекта? (Ровдо Д.И.)
- Некоторые роли имеют некоторые схожие функции. Получается, что некоторые функции пересекаются между разными ролями? (Ровдо Д.И.)
- Кто такой product owner? (Ровдо Д.И.)
- Как можно сопоставить качества человека и роли? (Щавровский С.А.)
- Возможно ли создать команду без разделения ролей? (Щавровский С.А.)

- Кто такие full-stack разработчики? (Щавровский С.А.)
- Какие роли лучше всего сочетать? (Щавровский С.А.)
- Кто занимается обучением сотрудников в IT-компаниях? (Ярошевич Я.О.)
- Какие навыки нужны для роли командного лидера? (Ярошевич Я.О.)
- Бывают ли такие проекты, на которых отсутствует QA-менеджер? (Ярошевич Я.О.)
- Какими ролями можно пренебречь? (Ярошевич Я.О.)

Стратегии развития крупнейших и наиболее известных IT-компаний

- Какие есть аналоги кремниевой долины в мире, какие компании там представлены? (Борисевич П.И.)
- Какое место в рейтинге самых дорогих компаний занимает Tesla Motors? Верите ли вы что эта компания через 10 лет обгонит Apple как обещает Элон Маск? (Борисевич П.И.)
- Какие не очень большие компании имеют самые высокие темпы роста в последнее время? (Борисевич П.И.)
- Расскажи, пожалуйста, историю надкусанного яблока? (Григорьев А.В.)
- Почему продукция Apple была и остаётся такой популярной? (Григорьев А.В.)
- Возможно ли в Беларуси основать бизнес, который станет настолько крупным, как упомянутые? Почему нет? (Григорьев А.В.)
- Что из себя представляли табуляторы? (Григорьев А.В.)
- В чём заслуга IBM PC? (Григорьев А.В.)
- Какое место в рейтингах занимает Oracle? (Грушевский А.А.)
- Какая компания для тебя является самой успешной и почему? (Грушевский А.А.)
- Какие компании в нашей стране активно набирают обороты? (Ипатов А.Е.)
- Лучше делать упор на какую-то узкую сферу или же наоборот стараться внедрить свой продукт во все сферы? (Ипатов А.Е.)
- Были ли примеры быстрого старта компаний, а затем такого же быстрого падения? С чем это связано? (Ипатов А.Е.)
- Сможет ли в ближайшее время приблизиться к уровню Epam какая-нибудь белорусская компания? (Ипатов А.Е.)
- В каких из белорусских компаний ты бы предпочла работать? (Ипатов А.Е.)
- Можно ли считать, что компания Apple Inc достигла своего успеха только благодаря Стиву Джобсу? (Михальцова А.Ю.)
- Есть ли смысл создавать свою компанию, если уже так много имеющихся крупных и сильно развитых? (Михальцова А.Ю.)
- Останется ли рейтинг IT-компаний таким же, если в качестве критерия оценки добавить еще качество производимой продукции? (Михальцова А.Ю.)
- Почему компания Sony прекратила производство ноутбуков? (Михальцова А.Ю.)
- А как так вышло, что Google так далеко от вершины? Ведь он гораздо более на слуху, чем тот же HP. (Ровдо Д.И.)
- Как ты думаешь, кто из перечисленных компаний останется в топе через 10 лет? (Ровдо Д.И.)

- У Amazon имеется своя БД – DynamoDB. Получается, это не только интернет магазин. А какие продукты у них еще имеются? (Ровдо Д.И.)
- Каковы технические характеристики Apple 1 и Apple 2? (Трубач Г.Г.)
- Почему Apple и Samsung постоянно копируют что-нибудь друг у друга? И как они разбираются с другими компаниями, которые копируют их разработки (китайцы, например)? (Трубач Г.Г.)
- Из-за чего произошел упадок компании IBM? (Трубач Г.Г.)
- Расскажи про компанию Sony поподробнее. И почему у них маленькие доходы (иногда вообще убытки) от мобильной отрасли (отделение ноутбуков уже вообще продали)? (Трубач Г.Г.)
- Какие еще есть известные IT компании, созданные белорусами? (Трубач Г.Г.)
- Есть ли какие-то собственные продукты у компании EPAM? (Щавровский С.А.)
- Почему среди первых 9 компаний из вашего списка, нету ни одной чисто софтверной? (Щавровский С.А.)
- Amazon занимается чем-нибудь еще, кроме магазина? (Щавровский С.А.)

Архитектура компьютера и мобильных устройств

Архитектура компьютера

- Какие существуют способы для ускорения передачи данных между ПЗУ (ОЗУ) и регистрами? (Белый А.А.)
- В чем существенное различие между поколениями ОЗУ (DDR2, DDR3, DDR4)? (Белый А.А.)
- Почему не используют закрытую архитектуру? (Гетьман С.И.)
- Какова максимальная пропускная способность системной шины? (Гетьман С.И.)
- Что означает разрядность процессора? (Григорьев А.В.)
- Может ли CD-диск выступать в качестве системного диска? (Григорьев А.В.)
- Существуют и используются ли компьютеры с закрытой архитектурой? (Григорьев А.В.)
- Все ли компьютеры используют BIOS? (Григорьев А.В.)
- В чем различия распределения ресурсов в одно- и мультипроцессорных системах? (Грушевский А.А.)
- Как разрешаются конфликты доступа к ресурсам мультипользовательских UNIX-системах? (Грушевский А.А.)
- Какие существуют способы расширения возможностей компьютера? (Ипатов А.Е.)
- Особенности cash-памяти и ее уровни (L1, L2 и т.д.)? (Ипатов А.Е.)
- В чем отличия системной и адресной шин? (Ипатов А.Е.)
- Какие существуют аналоги центральной системной шины? Почему остановились на использовании магистралей? (Лебедев Н.А.)
- Можно ли жесткий диск отнести к модулям? (Михальцова А.Ю.)
- От каких параметров зависит частота импульсов и что можно сделать, чтобы ее увеличить? (Михальцова А.Ю.)
- Без каких компонентов архитектуры компьютер может работать (запускаться)? (Ровдо Д.И.)
- В чем заключается разница между HDD и SSD дисками? (Трубач Г.Г.)
- Как производится обмен данными внутри компьютера? (Трубач Г.Г.)

- Как устроен и какие функции выполняет сопроцессор? (Трубач Г.Г.)
- Существуют ли процессоры с разрядностью большей чем x64? (Трубач Г.Г.)
- Что произойдет, если вставить в нужный разъем на материнской плате планку оперативной памяти большего объема, чем это допустимо платой? (Трубач Г.Г.)
- Какие бывают типы жестких дисков и в чем их различия? (Щавровский С.А.)
- В чем структурны различия между графическим процессором и центральным процессором? (Щавровский С.А.)
- Каковы структурные различия ядра UNIX-систем и GNU/Linux? (Щавровский С.А.)
- Какие есть примеры универсальных компьютерных шин? (Ярошевич Я.О.)
- А какие процессы выполняются, когда компьютер выключается? (Ярошевич Я.О.)

Операционные и вычислительные системы

- Какие ОС преимущественно ставят на суперкомпьютеры и почему? (Белый А.А.)
- Как понять, что ОС построена некорректно? (Гетьман С.И.)
- Как обеспечивается на аппаратном уровне возможность работы на одном компьютере с несколькими ОС? (Гетьман С.И.)
- В чем заключается ненадежность ОС Windows? (Гетьман С.И.)
- В чем заключаются недостатки ОС X? (Гетьман С.И.)
- Есть ли будущее у индивидуальных вычислительных систем? (Гетьман С.И.)
- В чем недостатки тех или иных ОС? (Ипатов А.Е.)
- Как ОС наиболее популярна? (Ипатов А.Е.)
- Какие ОС лучше использовать при решении определенных задач и почему? (Лебедев Н.А.)
- Что следует учитывать при разработке ОС? (Лебедев Н.А.)
- Какая ОС наиболее популярна? (Михальцова А.Ю.)
- Каковы преимущества и недостатки наиболее известных и используемых ОС? (Михальцова А.Ю.)
- Что относят к системному ПО? (Михальцова А.Ю.)
- Какая ОС используется в сложных системах, где требуется сильная и стабильная безопасность? (Ровдо Д.И.)
- Какие существуют нестандартные ОС? (Ровдо Д.И.)
- Какие ОС используются в управлении самолетами? (Трубач Г.Г.)
- В чем заключаются проблемы ОС Windows? (Трубач Г.Г.)
- Каков прогноз на разработку игр на свободные ОС? (Трубач Г.Г.)
- Какие ОС входят в ТОП-5 по различным показателям (количеству пользователей, дружественности GUI и т.д.)? (Ярошевич Я.О.)
- Какой есть самый простой путь, чтобы написать свою ОС? (Ярошевич Я.О.)

Мобильные операционные системы

- Какие главные недостатки мобильных операционных систем Android, IOS? (Борисевич П.И.)
- Может ли мобильное устройство использовать несколько операционных систем? (Борисевич П.И.)
- Какая операционная система наименее энергозатратна? (Борисевич П.И.)

- Почему не делают общий магазин для различных мобильных ОС? (Гетьман С.И.)
- Почему Windows Phone не имеет такой популярности среди мобильных ОС, как Windows для ПК? (Гетьман С.И.)
- Что можно сказать о безопасности Android, iOS, а также их взлома (перепрошивки)? (Гетьман С.И.)
- На кого рассчитана Android? IOS? BlackBerry? Windows Phone? Sailfish? Ubuntu Touch? (Гетьман С.И.)
- Зачем нужны альтернативные магазины для Android? Одного мало? (Гетьман С.И.)
- Вероятно ли появление на рынке и обретение популярности мобильной ОС, созданной не гигантом вроде Microsoft, Google, Apple? (Гетьман С.И.)
- А создаются и используются ли ОС, которые обеспечивают максимально минималистичный интерфейс и функционал (чтобы сделать телефон прочным телефоном вроде Nokia 3310)? (Гетьман С.И.)
- Почему не производят мобильные устройства, на которые можно поставить и Windows Phone, IOS или Android? Как скоро нас ждут подобные решения? (Гетьман С.И.)
- Какие ещё существуют мобильные операционные системы и какие из них способны стать одними из лидеров? (Григорьев А.В.)
- Перспективно ли создание собственной мобильной операционной системы? (Григорьев А.В.)
- Отличия и преимущество открытых и закрытых операционных систем. (Григорьев А.В.)
- Для какой из озвученных операционных систем легче/выгоднее создавать свои приложения? (Григорьев А.В.)
- Возможность и способы заработка на своих приложениях для мобильных операционных систем. (Григорьев А.В.)
- Какие существуют альтернативные разработчики ОС Android (кроме официальных)? (Грушевский А.А.)
- Какие есть отличия и нововведения в разных версиях мобильных ОС? (Грушевский А.А.)
- В каких еще сферах возможно внедрение той или иной мобильной ОС? (Ипатов А.Е.)
- С каждым годом IOS все больше и больше набирает обороты, а Android наоборот постепенно начинает уступать. С чем это связано? (Ипатов А.Е.)
- Есть ли смысл создания единой ОС, включающей все преимущества той или иной ОС? Возможно ли это вообще? (Ипатов А.Е.)
- Какие существуют глобальные проблемы современных ОС помимо энергосбережения? (Ипатов А.Е.)
- За счет чего добиваются эффективного энергоснабжения? Чем приходится жертвовать? (Ипатов А.Е.)
- Что прежде всего стоит учесть при создании мобильного приложения? Есть ли отличия от разработки desktop приложений? (Лебедев Н.А.)
- В чем секрет успеха Android? (Лебедев Н.А.)
- Какие отличительные черты у каждой из мобильных ОС? (Лебедев Н.А.)
- Известно, что любая мобильная ОС часто обновляется. С чем это связано? (Лебедев Н.А.)

- Если ли отличия в качестве и производительности открытой от коммерческой мобильной ОС? (Лебедев Н.А.)
- Microsoft в свое время упустили мобильный рынок. Правда, что мобильный Windows — качественный продукт или это хороший маркетинг Microsoft? (Лебедев Н.А.)
- Сильно ли схожа функционально Мобильных ОС с ОС для ПК? (Михальцова А.Ю.)
- Если все Мобильные ОС функционально схожи, зачем тогда их такое большое количество? (Михальцова А.Ю.)
- Что можно предпринять для улучшения энергопотребления? (Михальцова А.Ю.)
- Бесплатна ли Мобильная ОС? (Михальцова А.Ю.)
- Удобна ли синхронизация (данных) мобильного телефона с ПК? (Михальцова А.Ю.)
- Преимущества Android для управления автомобилем. (Михальцова А.Ю.)
- Как ОС влияет на производительность мобильного устройства? (Трубач Г.Г.)
- В чем преимущество ОС IOS? Зачем люди скупают продукцию Apple, которая стоит неоправданно дорого? (Трубач Г.Г.)
- Что из себя представляет ОС Firefox? Это ОС, или какой-нибудь веб-браузер, работающий на движке Gecko? Если второе, то как работает ОС? (Трубач Г.Г.)
- Какая ОС самая (менее) энергосберегательная? (Трубач Г.Г.)
- Есть ли будущее у Windows Phone? (Щавровский С.А.)
- Почему у Android такой большой отрыв в количестве пользователей? (Щавровский С.А.)
- Почему Windows Phone не может завоевать рынок? (Щавровский С.А.)
- Что является основным критерием для вас при выборе мобильной ОС? (Щавровский С.А.)
- Каким образом мобильная ОС "понимает", что следует перейти в другой режим для энергосбережения? Какие есть режимы, и как это организовано? (Ярошевич Я.О.)
- Есть ли возможность эмуляции мобильной ОС? К примеру, специальный софт, который позволяет на устройстве с Android отображать ОС, подобную iOS? И для других аналогично? (Ярошевич Я.О.)
- Какую последовательность действий нужно совершить, чтобы выложить свое приложение в маркет на каждой из мобильных ОС, чтобы пользоваться им в дальнейшем? (Ярошевич Я.О.)
- Есть такое наблюдение, что одни и те же приложения на разных ОС стоят по-разному. Как правило, на iOS дороже. С чем это связано? В пример можно привести приложение maps.me, которое является бесплатным на ОС Android, но платным на ОС iOS. (Ярошевич Я.О.)

Энергосбережение

- Может ли частое использование спящего режима ноутбука чем-то испортить батарею? (Борисевич П.И.)
- Как происходит переход в спящий режим жёстких дисков, когда они не используются? В каких операционных системах это можно сделать? (Борисевич П.И.)
- Может ли длительное использование компьютера от зарядки испортить его батарею? (Борисевич П.И.)
- Как влияет на энергосбережение мобильных устройств режим полета (fly mode)? (Гетьман С.И.)
- Что такое троттлинг? (Гетьман С.И.)

- Полезно ли для сохранения аккумулятора разряжать его и перезаражать чаще, чем держать его включенным в сеть? (Гетьман С.И.)
- Одинаково ли реагируют новые и старые физические составляющие компьютера на одни и те же алгоритмы энергосбережения? (Гетьман С.И.)
- Стоит ли доверять различным программам-уборщикам мусора и т.п.? (Гетьман С.И.)
- Как в случае короткого замыкания ведут себя батарея и аккумулятор? Как строится сберегательный софт с учётом такой ситуации? (Гетьман С.И.)
- Какие программы и алгоритмы энергосбережения есть для суперкомпьютеров? (Гетьман С.И.)
- Если вдруг на Земле не останется электричества, то как можно будет подзарядить устройства? (Гетьман С.И.)
- Много ли внимания уделяется энергосбережению персональных компьютеров? (Григорьев А.В.)
- Допустим есть ноутбук, который используется в качестве домашнего компьютера. Как лучше поступать: держать его постоянно в сети или ждать разрядки и потом заряжать? (Григорьев А.В.)
- Что должен учитывать разработчик мобильных приложений при написании кода и выборе технологий для лучшего энергосбережения? (Григорьев А.В.)
- Насколько распространены сейчас беспроводные способы зарядки тех или иных устройств? Какие вообще способы существуют? (Григорьев А.В.)
- Допустим есть разряженное мобильное устройство и нет возможности зарядить его стандартным способом. Можно ли зарядить его нестандартным способом, используя подручные средства? (Григорьев А.В.)
- (Грушевский А.А.)
- Какое из 4 состояний системы оптимально для работы? (Лебедев Н.А.)
- По каким критериям было сделано такое разделение системы на 4 состояния? (Лебедев Н.А.)
- Как можно увеличить время работы ПК? (Лебедев Н.А.)
- Понятно, что ограничение работы приложений снижает затратность, а как же батарея? Какие батареи самые энергозатратные? Энергосберегающие? (Лебедев Н.А.)
- На данный момент существуют ли инновационные идеи или решения проблемы энергосбережения? (Лебедев Н.А.)
- Какой режим энергосбережения наиболее эффективен: ждущий режим или глубокий сон? И в чем их различия? (Михальцова А.Ю.)
- Есть ли какая-нибудь зависимость между толщиной мобильного телефона и уровнем энергосбережения? (Михальцова А.Ю.)
- Для мобильных телефонов на платформе Android в магазине (Play Store) много приложений для энергосбережения. Эффективны ли они и почему? (Михальцова А.Ю.)
- Какова сущность алгоритмов энергосбережения? (Михальцова А.Ю.)
- Почему не стоит использовать калибровку батареи? (Михальцова А.Ю.)
- Что представляет из себя алгоритм interactive? (Ровдо Д.И.)
- Что такое калибровка батареи? (Ровдо Д.И.)

- Как часто нужно менять батарею и почему? (Ровдо Д.И.)
- Сейчас распространены powerbank. Значит ли это, что производителям мобильных устройств можно придумывать меньше хитрых способов уменьшить потребление батареи? (Ровдо Д.И.)
- Что такое режим гибернации и как он работает? Каковы его отличия от спящего режима? (Трубач Г.Г.)
- Какие существуют режимы энергосбережения в OS Android? (Трубач Г.Г.)
- Какие приложения потребляют больше всего электроэнергии в Android/iOS? (Трубач Г.Г.)
- Как работают режимы Stamina и Ultra Stamina в телефонах Sony? (Трубач Г.Г.)
- Каким образом реализован процесс энергосбережения (и реализован ли) на компьютерах, в которых критически важна производительность? (бортовые компьютеры самолета и т.д.) (Щавровский С.А.)
- Какой элемент операционной системы затрачивает наибольшее количество энергии в среднем? (Щавровский С.А.)
- Сколько экономится электроэнергии в мире при помощи алгоритмов энергосбережения? (Щавровский С.А.)
- Какие из мобильных ОС наиболее энергобережливая (топ5)? (Ярошевич Я.О.)
- Энергосбережение происходит не только на уровне ОС, но и на техническом: какой бренд ноутбука / телефона / планшета наиболее бережлив? (Ярошевич Я.О.)
- Известно, что при включенном Bluetooth устройство разряжается в разы быстрее. Существуют ли алгоритмы, чтобы это максимально обходить? Особенно это важно для умных часов и прочих устройств, которые постоянно работают на Bluetooth. (Ярошевич Я.О.)
- Как на батарею устройства влияет частота подзарядки? (Ярошевич Я.О.)

Языки программирования

Языки программирования. Обзор языков программирования

- Чем конкретно "вреден" оператор goto по Дейкстре? (Белый А.А.)
- Какие основные принципы функциональных языков программирования? (Какие уникальные возможности есть?) (Белый А.А.)
- Какие тенденции развития языков программирования? (Что в дальнейшем будет популярнее: языки типа Java или языки типа Python?) (Белый А.А.)
- Для каких программ использовали первые языки программирования? (Борисевич П.И.)
- Какие есть известные языки программирования без обязательной типизации данных? (Борисевич П.И.)
- Какие устройства могли запускать программы на первых языках программирования? (Борисевич П.И.)
- Почему интерпретатор выполняет код программ медленнее компилятора? (Борисевич П.И.)
- Есть ли среди языков программирования "мёртвые"? Какие атрибуты попадают под это определение? (Гетьман С.И.)
- Возможно было бы определение и создание "структурного программирования" раньше, чем в конце 60-х с выходом статьи Дейкстры? (Гетьман С.И.)

- Какие языки лучше и удобнее для разработки: со строгой или динамической типизацией? А для учёбы? (Гетьман С.И.)
- Как происходили развитие и эволюция компиляторов и интерпретаторов? (Гетьман С.И.)
- Что должен иметь язык программирования, чтобы стать используемым и популярным для разработки? (Гетьман С.И.)
- В чём смысл создания таких экзотических языков как Brainfuck (код посредством рисунка из слешей), Cook (код состоит из одной инструкции, повторяющейся различное число раз в зависимости от вызова необходимой функции)? (Гетьман С.И.)
- Нужны ли языки программирования, в которых даже структур нет (я уже молчу об классах и основах ООП)? (Гетьман С.И.)
- Нужны ли визуальные языки программирования? В чём их преимущества? (Гетьман С.И.)
- Что влияет на рост/падение популярности языка программирования? (Григорьев А.В.)
- Перспективно ли создание собственно языка программирования? И что для этого необходимо? (Григорьев А.В.)
- Где именно применяются низкоуровневые языки программирования? (Григорьев А.В.)
- Что относят к "нормальным алгоритмам"? (Михальцова А.Ю.)
- Если использовать оператор goto вредно, почему тогда на языке Assembler он один из основных при условном переходе? (Михальцова А.Ю.)
- Какие языки программирования относят к функциональным? (Михальцова А.Ю.)
- Что из себя представляют функциональные языки программирования? (Михальцова А.Ю.)
- Язык C++ к какому способу реализации относят? (Михальцова А.Ю.)
- Чем отличаются языки программирования низкого и высокого уровня? (Михальцова А.Ю.)
- Какие языки программирования относят к языкам программирования высокого уровня? (Михальцова А.Ю.)
- Если Java такая "тяжелая", почему она так распространена? Ведь ее используют не только для программирования под Android, но и в веб-приложениях. (Ровдо Д.И.)
- В чем плюсы таких языков как Python и Ruby, раз на них переходят? (Ровдо Д.И.)
- С чем вообще связана популярность тех или иных языков программирования? Что случается такого, что заставляет людей переходить на другие? (Ровдо Д.И.)
- Какова зависимость языков программирования от ОС? (Ровдо Д.И.)
- Присутствует ли оператор goto в языке Java? Почему этот оператор нежелательно использовать? (Трубач Г.Г.)
- Каковы плюсы и минусы ООП? Чем плохи "висячие" методы в коде? (Трубач Г.Г.)
- Приложения под ОС Android пишутся только на Java или еще на каких-нибудь языках? (Трубач Г.Г.)
- Что из себя представляет язык программирования Swift? (Трубач Г.Г.)
- На каких языках программирования можно разрабатывать desktop-приложения? (Трубач Г.Г.)
- Что означает встраиваемый язык программирования? (Щавровский С.А.)

- В какой сфере наиболее применимы функциональные языки программирования? (Щавровский С.А.)
- Чего на ваш взгляд не хватает современным языкам программирования? (Щавровский С.А.)
- Есть такая байка: все бородатые программисты в свободное время пишут свой язык программирования (в частности так появился новый язык компании Apple «Swift»). Вопрос: собираетесь ли вы отращивать бороду и писать свой язык программирования? (Щавровский С.А.)
- Какие существуют языки программирования, написанные не на английском языке? (Команды на каком-либо другом языке) (Ярошевич Я.О.)
- Какие есть языки с динамической типизацией? (Ярошевич Я.О.)
- Почему программы, написанные на интерпретируемом языке, работают медленнее, чем программы, написанные на компилируемом языке? (Ярошевич Я.О.)

JavaScript

- В чем заключается неработоспособность функции isNan? (Белый А.А.)
- Что является самой интересной возможностью сейчас в JavaScript, и в будущих версиях, что обещают разработчики? (Белый А.А.)
- Что может вызвать утечку памяти в JavaScript? (Борисевич П.И.)
- Какие есть преимущества и недостатки TypeScript в сравнении с JavaScript, или кроме типизации никаких отличий между ними нет? (Борисевич П.И.)
- Является ли JavaScript удобным языком в разработке приложений для мобильных устройств? (Борисевич П.И.)
- В каком состоянии CEnvy на 2015 год? И кем он и для чего использовался? (Гетьман С.И.)
- Как осуществляется связь между JavaScript и основным языком разработки приложения (к примеру, C++)? (Гетьман С.И.)
- Лучше ли CoffeeScript чем JavaScript? (Гетьман С.И.)
- Как обходить null в JavaScript? (Гетьман С.И.)
- Как с удобством проводить отладку JavaScript кода без помощи браузера? А как отлаживать библиотеки JavaScript, вроде jQuery? (Гетьман С.И.)
- Используется ли JavaScript в сферах кроме web-разработки? (Григорьев А.В.)
- С чем связан рост популярности JavaScript? (Григорьев А.В.)
- Какие конкуренты существуют у JavaScript? (Григорьев А.В.)
- Существуют ли языки альтернативные JavaScript? (Ипатов А.Е.)
- Какой главный недостаток и главный существенный плюс JavaScript? (Ипатов А.Е.)
- Каковы дальнейшие перспективы развития данного языка? Не превратится ли он в "мертвый" язык? (Ипатов А.Е.)
- Говорят, что JavaScript очень похож на Python, Ruby, но все же он сам по себе. В чем его уникальность? (Ипатов А.Е.)
- Какие недостатки есть у JavaScript? (Лебедев Н.А.)
- Как устроен интерпретатор JavaScript? (Лебедев Н.А.)
- Лучшая IDE для разработки на JavaScript? (Лебедев Н.А.)
- Эффективно было бы сделать JavaScript типизированным? (Михальцова А.Ю.)

- Какой сценарный язык может конкурировать с JS при разработке веб-приложений на стороне клиента? (Михальцова А.Ю.)
- Какие библиотеки JavaScript наиболее распространены? (Михальцова А.Ю.)
- Что представляет собой отладчик в JavaScript? (Михальцова А.Ю.)
- Как происходит тестирование страниц, написанных на JavaScript? (Михальцова А.Ю.)
- В каких случаях имеет смысл писать сервер на JavaScript, а, например, не на Java? (Ровдо Д.И.)
- Какой самый популярный JavaScript фреймворк и почему? (Ровдо Д.И.)
- В чем преимущество нетипизированного языка программирования, как JavaScript? Всего лишь в том, что программист может не думать о типах? То есть, для программистов, которые не хотят слишком много думать? (Ровдо Д.И.)
- Существует ли возможность разработки desktop-приложений на языке JavaScript? (Трубач Г.Г.)
- Как объяснить следующее "3" -+-+-+ "1" + "1" / "3" * "6" + "2" = "42"? (Трубач Г.Г.)
- Возможна ли разработка приложения на JavaScript в связке с C++? (Трубач Г.Г.)

Трансляторы

- В настоящее время имеет ли место разработка новых трансляторов? Если да, то с какой целью? (Белый А.А.)
- Как происходит дизассемблирование кода? (Белый А.А.)
- Можно ли как-то запутать/сломать транслятор? (Белый А.А.)
- Как происходит оптимизация выходного кода транслятора? (Борисевич П.И.)
- Трансляторы используют разные алгоритмы оптимизации? Какие из них наиболее эффективные? (Борисевич П.И.)
- С какой целью могут использовать декомпиляторы? Можно ли получить исходный код какого-нибудь приложения декомпиляцией? (Борисевич П.И.)
- Как разрабатывают компилятор? Можно ли его написать в одиночку? (Гетьман С.И.)
- Как интерпретатор и компилятор обходят ошибки? (Гетьман С.И.)
- Какие самые популярные ошибки, возникающие во время компиляции? (Гетьман С.И.)
- Где мне найти и увидеть то, что из себя представляет компилятор? Как мне увидеть воочию каждую стадию перевода исходного кода в целевой? (Гетьман С.И.)
- Перспективно ли создание своего транслятора? (Григорьев А.В.)
- Существуют ли трансляторы, которые транслируют языки в более высокоуровневые? (Григорьев А.В.)
- Возможна ли декомпиляция кода? (Григорьев А.В.)
- Какие существуют исполнимые модули? В чём их отличия? (Григорьев А.В.)
- В чем преимущество трансляторов языка, написанных на этом же языке? (Грушевский А.А.)
- В каких случаях гораздо эффективнее преобразовывать язык сначала в Assembler, а потом уже в машинный код при компиляции? (Грушевский А.А.)
- Есть ли какие-нибудь трансляторы, которые преобразуют код из одного языка высокого уровня в другой? (Грушевский А.А.)
- Что представляет из себя "дерево разбора"? (Ипатов А.Е.)
- Возможна ли компиляция без перевода в машинный язык? (Ипатов А.Е.)

- Какие существуют псевдокоды? (Ипатов А.Е.)
- Какие существуют аппаратные подходы? (Ипатов А.Е.)
- Где применяется JIT? (Ипатов А.Е.)
- В чем преимущество однопроходного/многопроходного интерпретатора? (Лебедев Н.А.)
- Смысл транслятора генерировать самого себя? (Лебедев Н.А.)
- Хочу написать программу, которая сама себя дописывает в процессе работы. Существуют ли такие трансляторы под это дело? (Лебедев Н.А.)
- Можно ли обмануть транслятор? (Лебедев Н.А.)
- Возможно ли превзойти статическую компиляцию? (Лебедев Н.А.)
- Если ли золотая середина среди трансляторов? (Лебедев Н.А.)
- Можно ли JVM считать за транслятор? (Михальцова А.Ю.)
- Примеры видов трансляторов. (Михальцова А.Ю.)
- Что эффективнее: компилятору переводить программу на Ассемблер или в байт-код? И почему? (Михальцова А.Ю.)
- Существует ли какое-нибудь понятие/характеристика о скорости работы компилятора? (Михальцова А.Ю.)
- Приведите таблицу данных о том, какие виды компиляции/интерпретации используются в популярных языках программирования (Java, C/C++, Python, Ruby, PHP, JavaScript...) (Ровдо Д.И.)
- В чем существенная разница между интерпретатором компилирующего типа и компилятором с интерпретацией? (Ровдо Д.И.)
- В чем преимущество смены компиляции и интерпретации, как в Forth? (Ровдо Д.И.)
- На каких языках обычно пишутся трансляторы современных языков программирования? (Щавровский С.А.)
- В чем выигрывает компилирование кода, по сравнению с интерпретированием, за исключением скорости? (Щавровский С.А.)
- Каковы подробности про взаимопроникновение процессов трансляции и интерпретации? (Ярошевич Я.О.)
- Что такое дизассемблер? (Ярошевич Я.О.)
- Какие языки являются интерпретируемыми? Какие существуют соответствующие интерпретаторы, как они работают? (Ярошевич Я.О.)

Базы данных

Реляционные базы данных. SQL

- Какие есть отличительные особенности реляционных баз данных перед другими? (Борисевич П.И.)
- Что такое суррогатный ключ, и чем он лучше или хуже естественного ключа? (Борисевич П.И.)
- Можно ли полностью избежать дублирования информации в реляционных базах данных? (Борисевич П.И.)
- Почему реляционные БД имеют такой успех? (Гетьман С.И.)
- Почему с них не уходят в пользу нереляционных БД (Гетьман С.И.)
- Реляционные базы данных должны ли всегда соблюдать три правила? (Гетьман С.И.)

- Какие существуют нормальные формы (кроме трёх названных)? (Григорьев А.В.)
- Существуют ли ещё языки для работы с реляционными БД помимо SQL? (Григорьев А.В.)
- Следуют ли всем нормальным формам в серьёзных проектах? (Григорьев А.В.)
- Всегда ли в проектах используют БД, или существуют другие способы эффективного хранения данных? (Григорьев А.В.)
- В чем преимущество реляционных БД перед нереляционными и наоборот? (Грушевский А.А.)
- Какой процент полезных данных в разных реляционных БД? (Грушевский А.А.)
- Что значит программа на SQL? Ведь это просто язык запросов. (Грушевский А.А.)
- Востребован ли на данный момент SQL? (Ипатов А.Е.)
- В чем преимущества реляционных баз данных, а в чем недостатки? (Ипатов А.Е.)
- Какие наиболее популярны реляционные или нет? (Ипатов А.Е.)
- Есть ли какая-нибудь альтернатива SQL? (Ипатов А.Е.)
- Известны случаи, когда выполнение 3 нормальных форм в сочетании с ОРМ ухудшает процесс разработки. Как быть? (Лебедев Н.А.)
- Какие существуют пользовательские функции в MySQL? Зачем оно надо и как пользоваться? (Лебедев Н.А.)
- В каких случаях целесообразно пользоваться view? (Лебедев Н.А.)
- Зачем разделение на нормальные формы отношения в реляционной модели данных? (Как это помогает в работе с данными?) (Михальцова А.Ю.)
- Почему считается, что данные лучше хранить в базах данных, а не на сервере? (Михальцова А.Ю.)
- Есть ли наиболее популярный аналог MySQL? (Михальцова А.Ю.)
- (Михальцова А.Ю.)
- Отличия 3 нормальной формы от других? (Ровдо Д.И.)
- Что такое кортеж? (Ровдо Д.И.)
- В чем разница между MySQL, MS SQL Server и тд? Какие ещё существуют реляционные БД? (Ровдо Д.И.)
- Какие диалекты SQL существуют и чем они отличаются? (Ровдо Д.И.)
- Есть ли такие программы, как MySQL Workbench, у других реляционных БД? (Ровдо Д.И.)
- Какие ORM существуют для работы с БД? (Трубач Г.Г.)
- Как организуются транзакции в SQL? (Трубач Г.Г.)
- Что такое триггеры и зачем они нужны? (Трубач Г.Г.)
- Каковы различия между БД Mysql и Oracle? (Трубач Г.Г.)
- Зачем нужно такое разнообразие реляционных БД: MySQL, Postgres, Oracle и т.д.? (Щавровский С.А.)
- Что такое триггер? (Щавровский С.А.)
- Посредством какого языка, кроме SQL происходит управление БД? (Щавровский С.А.)
- Какие есть нормальные формы, которые не были указаны в докладе? (Ярошевич Я.О.)
- Расскажите про 12 правил Кодда. (Ярошевич Я.О.)

- Выстраивали ли вы когда-нибудь архитектуру для реляционной базы данных? (Ярошевич Я.О.)

Нереляционные базы данных

- Какие есть аналоги Hibernate для MongoDB? (Белый А.А.)
- Когда какие базы (реляционные или нет) выгоднее использовать? (Белый А.А.)
- Какие есть самые странные базы данных? (Белый А.А.)
- Нереляционные базы данных могут полностью заменить реляционные? (Борисевич П.И.)
- Как правильно подобрать тип хранилища нереляционной БД? (Борисевич П.И.)
- Можно ли объективно заявить, что нереляционные БД лучше, чем реляционные, или наоборот? Почему? (Григорьев А.В.)
- В каких случаях лучше использовать нереляционные БД? (Григорьев А.В.)
- Какие БД тебе больше нравятся: реляционные или нереляционные? (Грушевский А.А.)
- Подробнее про UnQL? (Грушевский А.А.)
- Каким образом использование нереляционных БД сокращает время разработки? (Грушевский А.А.)
- Как происходит создание баз данных без создания конкретной схемы? (Ипатов А.Е.)
- Расскажи подробнее о Base. (Ипатов А.Е.)
- Где на сегодняшний день используются? Примеры? (Ипатов А.Е.)
- В чем преимущества и какие недостатки? (Ипатов А.Е.)
- Будут ли у UnQL схожие с NoSQL типы хранилищ данных? (Михальцова А.Ю.)
- А какие еще типы хранилищ можно было бы реализовать? (Михальцова А.Ю.)
- В каких случаях удобнее использовать нереляционные БД, чем реляционные, отражающие логическую сущность в таблицах? (Михальцова А.Ю.)
- В чем различия, плюсы и минусы реляционных и нереляционных БД? (Трубач Г.Г.)
- Подробнее про Amazon DynamoDB. (Трубач Г.Г.)
- Как организуются запросы в noSQL? (Трубач Г.Г.)
- Каковы перспективы развития? (Трубач Г.Г.)
- В реляционных БД для управления используется SQL. А в нереляционных? (Щавровский С.А.)
- Правильно ли я понял: на малых мощностях реляционные БД быстрее, а с ростом мощности растёт преимущество noSQL? (Щавровский С.А.)
- Какие существуют IDE для MongoDB? (Ярошевич Я.О.)
- Расскажите, пожалуйста, про mongoose. (Ярошевич Я.О.)
- В каких случаях лучше использовать нереляционные базы данных? А конкретно MongoDB? (Ярошевич Я.О.)
- Какие существуют подходы хранения картинок в MongoDB? (Ярошевич Я.О.)

Разработка программного обеспечения

Разница между разработкой и производством программного обеспечения

- Понятия производства ПО и внедрения ПО эквивалентны между собой? (Борисевич П.И.)
- Можно ли считать разработку и производство разными стадиями жиненного цикла ПО? (Борисевич П.И.)
- Компьютерные игры разрабатываются или производятся? Почему? (Григорьев А.В.)
- Станет ли когда-нибудь разработка/производство софта конвеерным? (Григорьев А.В.)

- Как решаются проблемы разработки такие как: недостаточность трассировки, недостаточность контроля? (Грушевский А.А.)
- Что такое бережливая разработка ПО? (Грушевский А.А.)
- Если эти определения настолько похожи, то может их таки стоит объединить? (Ипатов А.Е.)
- Какие главные отличия между процессами? (Ипатов А.Е.)
- Различают ли их как-то на реальных проектах? (Ипатов А.Е.)
- Кто чаще всего занимается реализацией того или иного процесса? (Ипатов А.Е.)
  "Разработка - это каркас и на него что-то вешается". Что понимается под каркасом? Просто, на мой взгляд, разработка - это, в основном, процесс анализа и проектирования, те каркаса как такового быть не может. (Лебедев Н.А.)
- Производству присущ недостаток контроля, но как такое может быть? Разве производство, особенно массовое, не предполагает этого? (Лебедев Н.А.)
- Почему разработка и производства ПО аналогична разработке техники? Да, эти вещи близки, но если говорить о разработке, то тут скорее второе является частью первого. (Лебедев Н.А.)
- Можно ли реализацию, написание кода отнести к разработке? (Михальцова А.Ю.)
- Как лучше организовать часть жизненного цикла: разработка до производства или наоборот? (Михальцова А.Ю.)
- Что подразумевается под сырьем в определении производства? (Михальцова А.Ю.)
- Долго ли просуществует компания, которая занимается только разработкой? А только производством? (Ровдо Д.И.)
- Какие виды производства различают и их особенности? (Трубач Г.Г.)
- Разработка входит в производство или нет? (Трубач Г.Г.)
- Пример неправильного выбора методологии. Как поступать в такой ситуации? (Ярошевич Я.О.)
- Бережливая разработка ПО (Ярошевич Я.О.)
- Можно ли избежать всех проблем разработки ПО? (Ярошевич Я.О.)

Парадигмы программирования

- Что представляет собой стек-ориентированное программирование? (Белый А.А.)
- Какие парадигмы являются основными для определения языка программирования? (Белый А.А.)
- Какие основные отличия нестрого программирования от строгого? (Борисевич П.И.)
- Где используется или использовалось табличное программирование? (Борисевич П.И.)
- Какие есть примеры языков, поддерживающих матричное программирование? Сейчас такие языки программирования где-нибудь используются? (Борисевич П.И.)
- Почему нет языка, чётко воплощающего одну и только одну парадигму программирования? (Гетьман С.И.)
- Табличное программирование способно выступить хорошим ORM? (Гетьман С.И.)
- А возможно написать язык без использования парадигмы на уровне значений? (Гетьман С.И.)

- Были ли попытки обобщить все эти парадигмы в единое целое для создания самого универсального языка программирования? Чем они закончились и почему? (Гетьман С.И.)
- Возможно ли существование языка программирования, который поддерживает лишь одну парадигму? (Григорьев А.В.)
- Есть ли парадигма, которая очевидно превосходит другие парадигмы? Или у всех есть как плюсы, так и минусы? (Григорьев А.В.)
- Почему ООП стал так популярен? (Григорьев А.В.)
- Почему в нашем университете нас знакомили только с ООП? (Григорьев А.В.)
- Есть ли язык, который поддерживает все парадигмы, и делает ли это его лучшим языком? (Григорьев А.В.)
- В чем преимущества одних парадигм перед другими, когда это критически? (Грушевский А.А.)
- Что такое матричное программирование? (Грушевский А.А.)
- В каких случаях использование одной или другой парадигмы значительно упрощает код или увеличивает скорость написания программы? (Грушевский А.А.)
- Почему эзотерическая парадигма не воспринимается всерьёз? (Ипатов А.Е.)
- Где применяется табличная парадигма? (Ипатов А.Е.)
- Расскажите про скалярную парадигму. (Ипатов А.Е.)
- Что представляют собой "нестрогие функции"? (Михальцова А.Ю.)
- Как возникают парадигмы программирования? (Михальцова А.Ю.)
- К какой парадигме можно отнести язык R? (Михальцова А.Ю.)
- Чем отличаются табличная и матричная парадигма? (Михальцова А.Ю.)
- Что представляет собой парадигма "обмен сообщениями"? (Ровдо Д.И.)
- В чем заключаются особенности языка «Cat»? (Ровдо Д.И.)
- Что обозначает термин табличное программирование? Табличное программирование — это как программа, в которой все данные находятся в базе данных? (Ровдо Д.И.)
- В чем минусы ООП? (Трубач Г.Г.)
- Что такое АОП? (Трубач Г.Г.)
- Что такое строгие и нестрогие функции? (Трубач Г.Г.)
- Что такое атом в парадигме на уровне функций? (Трубач Г.Г.)
- Где используется процедурная парадигма? (Ярошевич Я.О.)
- Когда и при каких обстоятельствах появился термин "парадигма программирования»? (Ярошевич Я.О.)
- С какими языками связана обобщенная парадигма? (Ярошевич Я.О.)
- Что означает парадигма "обмен сообщениями»? (Ярошевич Я.О.)

Стили программирования

- С какой целью создана функция eval? (Белый А.А.)
- Как осуществляется замена условного оператора полиморфизмом? (Белый А.А.)
- Для ООП есть подход с использованием uml-диаграмм. А какой подобный подход есть для функциональных языков? (Белый А.А.)

- Какие стили программирования используются в разработке мобильных приложений? (Борисевич П.И.)
- Какие методы рефакторинга на ваш взгляд самые интересные? (Борисевич П.И.)
- Чем стили программирования отличаются от парадигм программирования? (Борисевич П.И.)
- При других стилях программирования кроме ООП наследование запрещено/не используется? (Гетьман С.И.)
- Были ли до Роберта Мартина попытки обозначить и унифицировать понятие "clean code"? (Гетьман С.И.)
- Как программисты пришли к необходимости следовать clean code? (Гетьман С.И.)
- Должен ли clean code в различных языках быть реализован одинаково? (Гетьман С.И.)
- Есть такое пожелание к clean code: если метод private, то имя его должно быть длинным. А какая максимальная длина допустима? (Гетьман С.И.)
- Всегда задаю этот вопрос: какое количество аргументов в функции приемлемо? (Гетьман С.И.)
- Необходимо ли всегда, вне зависимости от сложности проекта, производить рефакторинг? (Гетьман С.И.)
- По Р. Мартину 10 строк в методе - уже перебор. А есть ли алгоритмы, которые не уложить в эти 10 строк? Как их "подгоняют" под требования clean code? (Гетьман С.И.)
- Clean code приемлем для языков низкого уровня, таких как ASM? Или же это исключительно порождение ООП? (Гетьман С.И.)
- Можно ли усовершенствовать clean code или этот набор требований не нуждается в дополнении? (Гетьман С.И.)
- Существуют ли языки, которые рассчитаны на использование конкретного стиля? (Григорьев А.В.)
- Если стили программирования можно сравнить со стилями одежды, то существует ли мода в программировании? То есть, бывает ли, что в какой-то момент времени модно писать в каком-то стиле? (Григорьев А.В.)
- Почему clean code для разных языков различается? С чем это связано? (Григорьев А.В.)
- Каков clean code для языка Assembler? (Григорьев А.В.)
- На что делать упор, а чем пренебрегать: понятностью или скоростью? (Григорьев А.В.)
- Какой из стилей программирования является классическим стилем? (Ипатов А.Е.)
- Что представляет из себя "выделение метода"? (Ипатов А.Е.)
- Является ли знание каких-то основ "клин кода" критическим при устройстве на работу, работе в каком-то реальном проекте? (Ипатов А.Е.)
- Как осуществляется процесс рефакторинга? (Ипатов А.Е.)
- Какие существуют самые известные методы рефакторинга. (Лебедев Н.А.)
- Иногда клин-код приводит к значительному увеличению метода/класса, с другой стороны простыню нельзя допускать. Как быть? (Лебедев Н.А.)
- Если функциональное программирование вынесено как отдельный стиль, почему тогда нет визуального? (Лебедев Н.А.)

- Если стили программирования аналогичны соответствующим парадигмам, почему их так мало? (Лебедев Н.А.)
- Как стиль программирования зависит от выбранной методологии? (Михальцова А.Ю.)
- Если нет специалистов в функциональном стиле программирования, значит ли это, что знание это не востребовано или область изучения достаточно сложная? (Михальцова А.Ю.)
- Рекомендуют писать код используя clean code. Но в одной статье было написано, что чтобы закрепить за собой рабочее место, не стоит писать так, чтобы код был понятен другим программистам и вас легко можно было бы заменить. Так как быть? (Михальцова А.Ю.)
- Что, на твой взгляд, лучше: "чистый код" или "быстрый код"? (имеется в виду хороший алгоритм, который эффективно работает, но плохо читабелен) (Ровдо Д.И.)
- На каких этапах разработки следует делать рефакторинг? (под разработкой понимается само написание кода) (Ровдо Д.И.)
- Стиль программирования не является стилем написания кода программиста, а чем-то схож с парадигмами? (Ровдо Д.И.)
- Рефакторинг нужен исключительно для более быстрого понимания чужого кода? (Ровдо Д.И.)
- Какой стиль программирования сейчас наиболее популярен? (Трубач Г.Г.)
- Особенности clean code для C++. (Трубач Г.Г.)
- Какие трудности могут возникнуть, если не производить рефакторинг? (Трубач Г.Г.)
- Какие существуют возможности авторефакторинга в различных IDE? (Трубач Г.Г.)
- Какие существуют книги по стилям программирования, по рефакторингу? (Щавровский С.А.)
- Существует практика, в которой рефакторинг не используется, и даже не приветствуется, ввиду его "необоснованной трудоемкости". Какие есть мнения на этот счет? (Щавровский С.А.)

Паттерны проектирования

- Какие хорошие практические советы по реализации паттерна "адаптер"? (Белый А.А.)
- Расскажите, что из себя представляют антипаттерны, какие используются активно в серьезной разработке? (Белый А.А.)
- В чем различие антипаттернов <название_еды>-код? (Белый А.А.)
- Существуют ли паттерны, которые невозможно реализовать на каком-нибудь распространённом языке программирования? (Белый А.А.)
- Какие структурные паттерны проектирования сейчас наиболее популярны? (Борисевич П.И.)
- Какие известные анти-паттерны, на ваш взгляд, самые интересные? (Борисевич П.И.)
- Какие трудности могут возникнуть с использованием шаблонов проектирования? (Борисевич П.И.)
- Вносились ли поправки в книгу Design Patterns после 1991 года? (Гетьман С.И.)
- Почему MVC является паттерном? (Гетьман С.И.)
- Всегда ли паттерны программирования предполагают использование парадигмы ООП? (Гетьман С.И.)
- Зачем нужны антипаттерны? (Гетьман С.И.)
- Что такое GRASP? (Гетьман С.И.)

- Почему до 1991 года никто не выпускал серьёзных работ по шаблонированию разработки? (Григорьев А.В.)
- Особенности паттерна singleton? (Григорьев А.В.)
- В следствии чего появляются новые паттерны? (Григорьев А.В.)
- Нужно ли использовать паттерны или нет? (Григорьев А.В.)
- Какие паттерны, на твой взгляд, самые полезные? (Грушевский А.А.)
- Какие антипаттерны наиболее популярны? (Грушевский А.А.)
- Что такое блоб? (Грушевский А.А.)
- Как бороться с адом зависимостей? (Грушевский А.А.)
- Имеет ли шаблон concurrency какое-то отношение к многопоточности? (Михальцова А.Ю.)
- Что представляет из себя такой шаблон, как абстрактная фабрика? (Михальцова А.Ю.)
- Что такое антипаттерны, и для чего они применяются? (Михальцова А.Ю.)
- Какие шаблоны проектирования самые популярные? Какие спрашивают на собеседованиях? (Ровдо Д.И.)
- Что из себя представляет антипаттерн "слепая вера"? (Ровдо Д.И.)
- Каковы плюсы и минусы паттерна неизменяемого объекта (immutable)? (Ровдо Д.И.)
- Говорят, что паттернами увлечены, в основном, программисты среднего уровня, опытные относятся к ним куда прохладней. Почему так? (Ровдо Д.И.)
- Почему примитивы синхронизации указывались как паттерны в ответвлении concurrency? (Трубач Г.Г.)
- Что из себя представляет паттерн строитель? (Трубач Г.Г.)
- Какие существуют паттерны для взаимодействия с БД? (Трубач Г.Г.)
- Как нужно выбирать паттерн? (Трубач Г.Г.)
- Какие книги на тему паттернов проектирования обязательного необходимо прочитать? (Щавровский С.А.)
- Считаете ли вы, что знание и следование паттернам проектирования является необходимым для современного специалиста? (Щавровский С.А.)
- Паттернов много. А какой базис среди них можно выделить? (Щавровский С.А.)
- Особенности структурного паттерна "Декоратор" ("Обертка"). (Ярошевич Я.О.)
- Когда лучше использовать MVC, а когда MVVM? (Ярошевич Я.О.)
- В чем различия антипаттернов "Спагетти-код", "Лазнья-код", "Равиоли-код"? (Ярошевич Я.О.)
- Особенности антипаттерна "Золушкина туфелька " (Ярошевич Я.О.)

Разработка мобильных приложений

- За какие нарушения приложение могут не выложить в App Store, Google Play? (Борисевич П.И.)
- Какие есть популярные движки для разработки мобильных приложений? (Борисевич П.И.)
- Как можно заработать на бесплатном приложении? (Борисевич П.И.)
- А выгодно ли разрабатывать приложения на мобильные устройства без желания заработать? (Гетьман С.И.)
- Как не застрять на этапе написания User's Story для мобильного приложения? (Гетьман С.И.)

- Когда ждать кризис идей из-за коммерциализации на рынке мобильных приложений? (Гетьман С.И.)
- Зачем нужны комментарии в AppStore, если там высказывается необъективная критика, которой не могут воспользоваться разработчики? (Гетьман С.И.)
- Является ли Xamarin хорошим подспорьем для разработки приложений как wMMD или BuildAnApp? (Гетьман С.И.)
- Стоит ли создавать платное приложение? Или всё же бесплатное с монетизацией? (Григорьев А.В.)
- Какие есть виды монетизации? И какие требования тот или иной вид имеет к самому приложению? (Григорьев А.В.)
- С какими ограничениями сталкиваются разработчики мобильных приложений? (Григорьев А.В.)
- Если в файле .html в теге <link> в атрибуте media укажем "handheld", то где мы сможем протестировать страницу? (Михальцова А.Ю.)
- Как разместить свои приложения в различных Store? (Михальцова А.Ю.)
- Куда обращаться, если найден баг в мобильном приложении? (Михальцова А.Ю.)
- (Ровдо Д.И.)
- Что нужно сделать для того, чтобы можно было выкладывать приложения в Google market/Appstore? (Трубач Г.Г.)
- Зачем нужны manifest файлы в android приложениях. (Трубач Г.Г.)
- Какие есть советы, чтобы ваше приложение попало в топ магазина? (Трубач Г.Г.)
- Каковы принципы размещения рекламы в приложениях? (Трубач Г.Г.)
- В плане оптимизации и производительности, нативная разработка приложений будет лучше, чем кроссплатформенная. А все же почему кроссплатформенная разработка пользуется популярностью? (Щавровский С.А.)
- Как происходит зарабатывание денег в самых известных магазинах мобильных приложений (какой процент идет разработчику, какие условия магазинов?) (Щавровский С.А.)
- Какой магазин аккумулирует больше денег разработчикам? (Щавровский С.А.)
- Сейчас существует множество одинаковых мобильных приложений от различных кафе. "Фабрика лояльности" указана в каждом из них. Не могли бы Вы рассказать подробнее про такие приложения? (Ярошевич Я.О.)
- Зачем тестировщику нужен шкаф с телефонами, если сейчас есть достаточно возможностей эмулировать любое мобильное устройство на компьютере? (Ярошевич Я.О.)
- Можно ли заплатить Маркету, чтобы Ваше приложение рекомендовалось (поднималось в топы). Например, такая функция точно есть у Google: если хотите, чтобы Ваш сайт был в топе по таким-то ключевым словам, то нужно заплатить им какую-то сумму. (Ярошевич Я.О.)

Тестирование

- Как типы тестирования используют в разработке мобильных приложений? (Борисевич П.И.)
- Что должен хорошо уметь делать тестировщик? (Борисевич П.И.)
- Какие есть особенности тестов, написанных перед разработкой приложения? (Борисевич П.И.)

- Зачем в тест-дизайне документация? Разве это не отягощает проект и работу над ним? (Гетьман С.И.)
- Зачем нужны обычные тестировщики, если они могут всё испортить? Откуда эти люди берутся? (Гетьман С.И.)
- Можно ли осуществить Test Driven Development в C++? (Гетьман С.И.)
- Кто важнее: разработчик или тестировщик? (Гетьман С.И.)
- Насколько чёткое разделение между разработчиками и QA? (Гетьман С.И.)
- Можно ли зарабатывать, будучи исключительно тестировщиком? (Григорьев А.В.)
- Если близко дедлайн и необходимо чем-то пожертвовать, то можно ли пожертвовать тестированием? Или стоит чем-то другим? (Григорьев А.В.)
- Для любого ли проекта подойдёт открытое тестирование? (Григорьев А.В.)
- Почему к тестировщикам относятся менее уважительно? (Грушевский А.А.)
- Что такое регрессивное тестирование? (Грушевский А.А.)
- Какие случаются трудности с usability-тестированием? (Грушевский А.А.)
- Какие из видов тестирования используются в реальных проектах? (Ипатов А.Е.)
- Какие виды тестирования используются чаще/реже всего? (Ипатов А.Е.)
- Как осуществляется выбор применяемого вида тестирования? Кто выбирает? (Ипатов А.Е.)
- Какие отличие между компонентным и модульным тестированием? (Михальцова А.Ю.)
- В чем суть приемочного уровня тестирования? (Михальцова А.Ю.)
- Почему выделяют TDD, если в жизненном цикле обычно определены фазы "разработка", а потом "тестирование"? (Михальцова А.Ю.)
- Связаны ли альфа- и бета-тестирования? И может ли одно осуществляться без другого? (Михальцова А.Ю.)
- Для чего используется каждый из видов тестирования? (Примеры ситуаций, когда используют нагрузочное, стресс-тестирование и т.д.) (Михальцова А.Ю.)
- Тесты, связанные с изменениями, включают в себя функциональные и нефункциональные? То есть, на самом деле не являются типом наравне с вышеперечисленными? (Ровдо Д.И.)
- Больше ли работы у тестировщиков при параллельном тестировании по сравнению с "тестами до кода"? (Ровдо Д.И.)
- Почему статус тестировщика считается ниже, чем у разработчика? (Ровдо Д.И.)
- Unit тесты. Что это и для чего? (Трубач Г.Г.)
- Как работает багтрекинговая система? (Трубач Г.Г.)
- Как можно автоматизировать тестирование? (Трубач Г.Г.)
- Какая наиболее используемая система тестирования? (Щавровский С.А.)
- Какие программные продукты для тестирования существуют? (Щавровский С.А.)
- Понятно, что пренебрегать тестированием нельзя, но каковы причины? (Щавровский С.А.)
- Какие есть особенности тестирования программ, написанных на определённых языках программирования? Есть ли такая связь тестирования с конкретным языком? (Ярошевич Я.О.)
- Какие есть роли тестировщиков? Какое для каждой роли требуется образование и какие нужны знания? (Ярошевич Я.О.)

- Автоматизация тестирования сейчас набирает обороты, можете рассказать подробнее про это? (Ярошевич Я.О.)
- Как Вы относитесь к практике написания тестов до написания самого кода? Используется ли сейчас такой подход? (Ярошевич Я.О.)

Стандартизация

- Каковы нововведения в Java 9? (Белый А.А.)
- Для чего, собственно, нужна стандартизация (вернее, как дошли до того, что она нужна) (Белый А.А.)
- Что должно быть указано в стандарте для ЯП, например? (Белый А.А.)
- Когда начали появляться первые стандартизированные языки? (Борисевич П.И.)
- Кто первым предложил использовать стандарты для языков программирования? (Борисевич П.И.)
- Без стандартизации обойтись на производстве и во время разработки можно? (Гетьман С.И.)
- Насколько заметны внесения изменений в стандартизацию и как это влияет на нас с вами программистов? (Гетьман С.И.)
- Так что такое стандарт языка? Просто документация? (Григорьев А.В.)
- Кто создаёт стандарты для различных продуктов? Почему им достаётся такая честь? (Григорьев А.В.)
- Что такое The Open Group? (Грушевский А.А.)
- В чем смысл крупного спонсирования организаций, занимающихся стандартизацией? Что это дает спонсорам? (Грушевский А.А.)
- Какие стандарты есть для QR кодов? (Грушевский А.А.)
- Что представляет из себя тот или иной стандарт? (Ипатов А.Е.)
- Стоит ли вообще изучать какие-то стандарты языков? (Ипатов А.Е.)
- Как происходит внедрение стандарта в среду? (Ипатов А.Е.)
- В чем отличия стандартов разных языков программирования? (Ипатов А.Е.)
- Какова роль стандартов в развитии языков программирования и их популяризации? (Лебедев Н.А.)
- Какое принципиальное отличие частного стандарта от международного? (Лебедев Н.А.)
- Понятно, что при расширении возможностей языка выходит новый стандарт. А были ли случаи кардинального изменения? (Лебедев Н.А.)
- Почему так принято, что данные хранятся именно в двоичном коде? (Михальцова А.Ю.)
- Можно ли синтаксис языка программирования отнести к стандартизации? (Михальцова А.Ю.)
- Обязан ли программист знать всё о стандартах того языка программирования, на котором он пишет код? (Михальцова А.Ю.)
- А как пришли к осознанию необходимости в стандартизации? (Ровдо Д.И.)
- Каковы нововведения в Java 9? Насколько сильно отличается от Java 8? (Ровдо Д.И.)
- Есть ли предел версий Java, C++? То есть, может ли получится так, что, например, что Java 32 будет такой, что дополнять уже просто нечего? (Ровдо Д.И.)

- В чем различия 8-го стандарта Java и 9-го? (Трубач Г.Г.)
- Какие самые значащие различия между C++ 11 и C++ 14? (Трубач Г.Г.)
- Каковы нововведения Java 9? (Ярошевич Я.О.)
- Почему официальный документ стандарта C++ стоит денег? (Ярошевич Я.О.)
  Какие стандарты ECMAScript готовятся к релизу? Какие нововведения? (Ярошевич Я.О.)

Активно развивающиеся технологии

Искусственный интеллект

- В чем фундаментальная разница между ИИ, который создали в компьютерных играх, и тем, что пытаются сделать в реальной жизни? (Белый А.А.)
- Думают ли ученые, которые создают ИИ, что он окажется в разы умнее человека? Как это может отразиться на нашей дальнейшей жизни (Белый А.А.)
- Какой основной принцип ИИ (наличие чего позволяет отнести нечто к ИИ)? (Белый А.А.)
- Искусственный интеллект сможет поработить мир? (Гетьман С.И.)
- Что можно сказать о фразе Айзека Азимова относительно робототехники (три принципа: робот не может причинить вред человеку или своим бездействием допустить, чтобы человеку был причинён вред, робот должен повиноваться всем приказам, которые даёт человек, кроме тех случаев, когда эти приказы противоречат Первому Закону, робот должен заботиться о своей безопасности в той мере, в которой это не противоречит Первому и Второму Законам.)? (Гетьман С.И.)
- Что может помочь воплотить эти принципы в жизнь? (Гетьман С.И.)
- Использование ИИ в игровой области. (Григорьев А.В.)
- Может ли ИИ быть опасным? Рассматривают ли современные исследователи его опасность? (Григорьев А.В.)
- С чем именно связано то, что ИИ до сих пор не достиг уровня человека? (Григорьев А.В.)
- Существовали ли какие-либо попытки изучения и создания ИИ в более ранней истории (до 20 века)? (Григорьев А.В.)
- Можно ли считать искусственным интеллектом такие программы, как программу управления светофорами, в зависимости от загруженности дороги, а также программу управления движением нескольких лифтов, работающих от одной кнопки? (Грушевский А.А.)
- Где находится граница между «умной» программой и ИИ? (Грушевский А.А.)
- Можно ли считать искусственным интеллектом сознание человека, записанное на компьютер (например, как в фильме «Превосходство»)? (Грушевский А.А.)
- Где на сегодняшний дань применяется искусственный интеллект? В каких сферах ожидается его внедрение? (Ипатов А.Е.)
- Искусственный интеллект стал популярен относительно недавно. С чем это связано? (Ипатов А.Е.)
- Какое средство является наиболее удобным и популярным для разработки системы ИИ? (Ипатов А.Е.)
- Развивается ли данная система в нашей стране? (Ипатов А.Е.)
- Что вообще относят к системам ИИ? (Ипатов А.Е.)
- Существуют ли принципиально другие виды хранения знаний в памяти? (Лебедев Н.А.)

- Почему в докладе делается упор на хранение знаний в БД, ведь реакция и анализ внешних данных тоже важны? (Лебедев Н.А.)
- По сути, пока не решена проблема объединения знаний в памяти, ИИ - обычный вывод информации из БД. Так ли это? (Лебедев Н.А.)
- Возможно ли создать искусственный разум, который был бы полностью аналогичен человеческому? (Михальцова А.Ю.)
- Перспектива развития ИИ. (Михальцова А.Ю.)
- Примеры использования ИИ. (Михальцова А.Ю.)
- Есть ли отрицательное влияние у ИИ? (Михальцова А.Ю.)
- Можно ли Wolfram считать за ИИ? (Михальцова А.Ю.)
- Какая на сегодняшний день существует самая умная система и что она умеет? (Ровдо Д.И.)
- Сможет ли когда-нибудь машина думать как человек? (Ровдо Д.И.)
- Не считаете ли вы, что перечисленные вами языки программирования довольно устаревшие? Действительно ли их сейчас используют для создания интеллектуальных систем? (Ровдо Д.И.)
- Как происходит обучение ИИ? (Трубач Г.Г.)
- Может ли компьютер/робот развить свой ИИ так, что выйдет из-под контроля? (Трубач Г.Г.)
- Какие существуют самые развитые машины/системы с ИИ? (Трубач Г.Г.)
- Удастся ли создать искусственный интеллект в полном смысле этого слова или нет? (Щавровский С.А.)
- А что насчет эмоций, чувств? (Щавровский С.А.)
- Не повредит ли создание искусственного интеллекта человечеству? (Щавровский С.А.)
- Можно ли Siri считать ИИ? (Ярошевич Я.О.) Расскажите, пожалуйста, про игровой искусственный интеллект подробнее. Какие есть подходы? (Ярошевич Я.О.)
- Что такое искусственный геном? (Ярошевич Я.О.)
- Как в контексте науки философия относятся к ИИ? А какое отношение со стороны религии и этики? (Ярошевич Я.О.)

Виртуальная реальность

- Реальна ли такая ситуация, что мы сами живём в виртуальной реальности (как в фильме "Матрица")? (Белый А.А.)
- К чему стремятся разработчики виртуальной реальности в будущем? Возможен ли такой печальный исход, что люди будут сидеть у себя дома и контактировать с этим миром исключительно при помощи, так называемых, аватаров? (Белый А.А.)
- Какие производители очков виртуальной реальности сейчас наиболее популярны? (Борисевич П.И.)
- Как виртуальная реальность влияет на человека? (Борисевич П.И.)
- Какие компании лидируют в развитии технологий виртуальной реальности? (Борисевич П.И.)
- Виртуальная реальность - это система, влияющая на органы чувств человека. То есть, имитация звуков и запахов это тоже виртуальная реальность? (Григорьев А.В.)
- Сейчас в игровой индустрии активно продвигается технология виртуальной реальности. А почему дополненная реальность не получила развития? (Григорьев А.В.)

- Возможен ли переход человечества в виртуальную реальность? (Григорьев А.В.)
- Как думаешь, получит ли развитие технология прямого подключения к мозгу/нервной системе? (Например, в играх) (Григорьев А.В.)
- Можем ли мы быть уверены, что не находимся в очень развитой виртуальной реальности? (Григорьев А.В.)
- Сможет ли сейчас виртуальная реальность заменить человеку реальный мир? (Лебедев Н.А.)
- К чему это стремится сфера виртуальной реальности? Какие цели она несет в себе? (Лебедев Н.А.)
- Какие основные математические модели/парадигмы используются в этой сфере? (Лебедев Н.А.)
- Что полезного в виртуальной реальности? Понятно, что реальность, которая имитирует условия реального мира, позволяет подготовить человека к определенным ситуациям. Есть ли еще примеры полезного использования виртуальной реальности? (Лебедев Н.А.)
- С большего виртуальная реальность — это обман. Тем не менее, многие согласны с тем, что ее нужно донести до масс. А правильно ли обманывать людей? (Лебедев Н.А.)
- Можно ли использовать виртуальную реальность как средство для управления людьми, поставив их в условия этого виртуального мира? (Лебедев Н.А.)
- А почему в определении виртуальной реальности присутствует слово мир? (Лебедев Н.А.)
- Примеры использования виртуальной реальности. (Михальцова А.Ю.)
- В комнате, когда шарик подвешивается по углам комнаты, а человек находится внутри него, как "считывается" реакция человека? (Михальцова А.Ю.)
- Каковы перспективы развития виртуальной реальности? (Михальцова А.Ю.)
- Какие существуют применения Omni? (Ровдо Д.И.)
- Как вообще можно сымитировать тактильные ощущения? (Ровдо Д.И.)
- Не случится ли так, как во всяких фильмах, что человек такими темпами перестанет различать виртуальную реальность от настоящего? (Ровдо Д.И.)
- Какова средняя цена таких игрушек? (Ровдо Д.И.)
- Перспективы развития виртуальной реальности. (Трубач Г.Г.)
- Oculus Rift. Каковы принципы его работы, и каково его устройство? (Трубач Г.Г.)
- Возможно ли восстание машин (например, роботов), как, например, Skynet? (Трубач Г.Г.)
- Что можно рассказать о системе управления махами руки в играх (например, в телевизорах Samsung, виртуальные тиры, консоли и т.д.)? (Трубач Г.Г.)
- С какими проблемами столкнулись разработчики виртуальной реальности? (Щавровский С.А.)
- Был ли у вас опыт пользования виртуальной реальности? (Щавровский С.А.)
- Расскажите про технические средства, на данный момент существующие (компании производители, лидеры рынка)? (Щавровский С.А.)
- Какие фирмы являются ведущими в направлении виртуальной реальности? (Топ 5) Какие у них последние разработки? (Ярошевич Я.О.)
- Какие есть самые необычные и весёлые разработки в этом направлении? (Ярошевич Я.О.)
- Какие разработки в этом направлении существуют в космической сфере? (Ярошевич Я.О.)

- Существует ли виртуальная еда? (Ярошевич Я.О.)

Нейронные сети

- Какой существует механизм обработки информации? (Белый А.А.)
- Как в бытовом плане могут использоваться нейронные сети (какие могут быть устройства, например)? (Белый А.А.)
- Какие есть известные модели НС? Чем они отличаются? Какие модели лучше подходят для прогнозирования результата? (Борисевич П.И.)
- Какие алгоритмы используются для обучения НС? (Борисевич П.И.)
- Как выбрать нужное число нейронов для сети? Можно ли считать, что большее число нейронов в сети будет лучше справляться с поставленной задачей? (Борисевич П.И.)
- Какие ещё существуют интересные и экзотические пороговые функции? (Гетьман С.И.)
- Вторая их предложенных пороговых функций (1/(1+exp{...})) позволяет "предотвратить переполнение"... поясни пожалуйста, переполнение чего? (Гетьман С.И.)
- Сумматор может дать сбой? А пороговая функция? В чём выражаются эти сбои? (Гетьман С.И.)
- Что есть понятие "слой" в контексте искусственных нейронных сетей? (Гетьман С.И.)
- Роботов обучают или они обучаются сами? Приведи примеры. (Гетьман С.И.)
- Как написать сумматор? (Гетьман С.И.)
- Возможно ли объединить архитектуру фон Неймана и архитектуру НИС? (Гетьман С.И.)
- Какие производственные задачи кроме тех, что ты перечислил, может и сможет в некотором будущем решать ИНС? (Гетьман С.И.)
- Что будет если вживить в человека ИНС каким-то образом, к примеру, через наномашины? (Гетьман С.И.)
- Может ли ИНС заместить биологическую нейронную сеть? (Гетьман С.И.)
- Можно поподробнее о процессе обучения с учителем, в частности о стимул-реакционной системе. (Гетьман С.И.)
- Могут ли меняться в ней пути передачи импульсов, создаваться новые нейроны, меняться веса сумматора в нейронах? (Грушевский А.А.)
- Есть ли аналоги синапсов для ИНС? Если есть, то какие их функции? (Грушевский А.А.)
- Как ИНС может обучаться? (Грушевский А.А.)
- Как адекватно определить, что ИНС обучилась правильно? (Грушевский А.А.)
- Какие существует аналоги AIBO? В каких сферах применяются? (Ипатов А.Е.)
- Каким образом происходит разработка и тестирование? (Ипатов А.Е.)
- Как происходит процесс обучения? (Ипатов А.Е.)
- Какие существуют способы обучения (возможно какие-то алгоритмы)? Какие сроки обучения? (Ипатов А.Е.)
- Как происходит распознавание текста/изображения? (Михальцова А.Ю.)
- Какое существует применение ИНС (подробнее)? (Михальцова А.Ю.)
- В чем заключается программное воплощение математической модели? (Михальцова А.Ю.)

- Как нейронные сети используются в робототехнике? (Михальцова А.Ю.)
- Сможет ли когда-нибудь искусственная нейронная сеть сравняться с биологической? (Ровдо Д.И.)
- Процесс обучения нейронной сети ограничен только временем и памятью? Или есть еще что-то? (Ровдо Д.И.)
- В каких случаях и почему лучше использовать полносвязную, многослойную и слабосвязную нейронную сеть? (Ровдо Д.И.)
- Как используются НС в поисковых системах? (Трубач Г.Г.)
- Как нейронные сети распознают картинки/звуки? (Трубач Г.Г.)
- На каком языке программирования в основном разрабатываются НС? (Трубач Г.Г.)
- Каковы перспективы развития НС? (Трубач Г.Г.)
- Какая наиболее используемая технология (зык программирования, например, или может какой-нибудь фреймворк) при создании нейронных сетей? (Щавровский С.А.)
- Пример компаний (команд) занимающихся нейронными сетями. Какие известные продукты есть у этих компаний (команд)? (Щавровский С.А.)
- Как происходит рисование картин нейронными сетями. (Ярошевич Я.О.)
- Какие есть известные типы сетей? Особенности сети Джордана. (Ярошевич Я.О.)
- Какие есть варианты будущего для нейронных сетей? (Ярошевич Я.О.)
- Какие есть продвижения в исследовании вопроса о возможности развития психологической интуиции у нейросетевых экспертных сетей? (Ярошевич Я.О.)

Облачные вычисления

Компьютерная и информационная безопасность

Компьютерная и информационная безопасность

- Как работает вирус троян (каким образом он получает информацию о введённых ключах авторизации)? (Белый А.А.)
- Назовите самое примитивное устройство, с которого можно взломать компьютерную сеть. (Белый А.А.)
- Какое самое масштабное преступление, связанное с хакерством? (Белый А.А.)
- Как злоумышленник может получить исходный код мобильного приложения? (Борисевич П.И.)
- На что в первую очередь следует обратить внимание разработчику, чтобы защитить своё приложение? (Борисевич П.И.)
- Почему операционные системы, основанные на ядре Linux, считают наиболее защищёнными от вирусов? (Борисевич П.И.)
- Что можно рассказать о безопасности и анонимности систем луковичных серверов? (Tor, к примеру) (Гетьман С.И.)
- Какие существуют стандарты шифрования? (Гетьман С.И.)
- Что такое DDoS-атаки? (Гетьман С.И.)
- Какие есть различия между безопасностью персонального компьютера и безопасностью сервера / mainframe машины? (Гетьман С.И.)

- Как аппаратно обеспечивается экран? (Гетьман С.И.)
- Если о шифре / безопасности системы знают уже двое, то её можно взломать. Как системами безопасности обходится такой барьер? (Гетьман С.И.)
- На любом ли языке программирования можно написать вирус? Почему? (Григорьев А.В.)
- Какие наказания грозят за хакерскую деятельность? (Григорьев А.В.)
- Можно ли законно зарабатывать, будучи хакером? (Григорьев А.В.)
- Как правильно удалять аккаунты, какую-либо информацию? (Ипатов А.Е.)
- Как работает брандмауэр? (Ипатов А.Е.)
- Как используется цифровая подпись? (Ипатов А.Е.)
- По каким критериям выделяют те или иные типы вирусов? (Ипатов А.Е.)
- Топ 10 хакеров и их атак. (Ипатов А.Е.)
- Как хоть как-нибудь защитить себя от тех или иных атак (хотя бы несколько способов)? (Ипатов А.Е.)
- А почему бы не ввести на всех сайтах виртуальную клавиатуру? И тогда кейлогер не страшен. (Ровдо Д.И.)
- Какой вирус самый забавный? (Ровдо Д.И.)
- А есть ли на Linux антивирусы? (Ровдо Д.И.)
- Какими способами можно положить сайт? (Трубач Г.Г.)
- Какие существуют способы защиты от DDoS атак? (Трубач Г.Г.)
- Как защищаются операции с денежными средствами? (Трубач Г.Г.)
- Как можно получить доступ к удаленному компьютеру? (Трубач Г.Г.)
- Как хакеры подбирают пароли от различных аккаунтов? (Трубач Г.Г.)
- Популярен ли хакинг сейчас? (Щавровский С.А.)
- Каково отношение к хакерам? (Щавровский С.А.)
- Что такое DDoS-атака? (Щавровский С.А.)
- Какие самые необычные компьютерные вирусы имели место? (Топ 5). В чем заключалась уловка, как впоследствии происходило заражение? (Ярошевич Я.О.)
- Какой уровень наказания хакеров в нашей республике? А в какой стране самое жестокое наказание для хакеров? (Ярошевич Я.О.)
- Какие были самые известные истории, связанные с хакерством? (Ярошевич Я.О.)

Антивирусы

- Антивирусы на мобильных устройствах, какие у них есть особенности? (Борисевич П.И.)
- Почему антивирус Касперского сильно тормозит всю систему, что он делает такого сложного? (Борисевич П.И.)
- Какие файлы обновляются в антивирусе ESET, что входит в несколько килобайт его обновления и достаточно ли их ему для надёжной защиты компьютера? (Борисевич П.И.)
- Когда стали появляться первые антивирусы? (Борисевич П.И.)
- Почему Avast такой медленный? (Гетьман С.И.)
- В чем преимущества антивируса Касперского? (Гетьман С.И.)

- Чем Касперский наделал в своё время столько шуму, что стал русским антивирусом номер один? (Гетьман С.И.)
- Встречаются ли антивирусы со встроенным фаерволлом? (Гетьман С.И.)
- Стоит ли пользоваться антивирусом на Linux? (Гетьман С.И.)
- Как ведёт себя антивирус в случае заражения вирусом его исходных и рабочих файлов? Баз сигнатур? (Гетьман С.И.)
- Есть ли качественные оффлайн-антивирусы? (Гетьман С.И.)
- У Windows есть собственные средства борьбы с угрозами. Так что, востребованы ли сейчас антивирусы? (Григорьев А.В.)
- Возможны ли ошибки в базах сигнатур? (Например, занесение невредоносного кода кода в базу.) (Григорьев А.В.)
- Есть ли среди бесплатных антивирусов объективный лидер? (Григорьев А.В.)
- (Грушевский А.А.)
- Какие из антивирусов являются наиболее эффективными, а какие наоборот лучше вообще не использовать? (Ипатов А.Е.)
- Существуют ли какие-либо алгоритмы для выявления той или иной атаки? (Ипатов А.Е.)
- Как вообще происходит выявление той или иной атаки? (Ипатов А.Е.)
- В чем суть функции anti-work? Что она делает? (Ипатов А.Е.)
- Как и кем осуществляется поддержка словаря сигнатур? (Лебедев Н.А.)
- Какой из методов защиты наиболее успешен/популярен? (Лебедев Н.А.)
- Какие математические методы участвуют в анализе кода? Можно ли говорить о высокой эффективности этих методов? (Лебедев Н.А.)
- Как распознается новый вирус? Обратная связь? (Лебедев Н.А.)
- Правда, что качество антивируса прямо пропорционально количеству потребляемых им ресурсов? (Лебедев Н.А.)
- Какие есть антивирусы под Linux? (Михальцова А.Ю.)
- Можно ли утверждать, что вирусы пишутся специально, чтобы потом продавать для них антивирусы? (Михальцова А.Ю.)
- Может ли компьютер нормально функционировать без антивируса, если нет подключения к интернету? (Михальцова А.Ю.)
- Эффективны ли антивирусы для мобильных устройств? (Трубач Г.Г.)
- Рейтинг использования антивирусов? (Трубач Г.Г.)
- Как происходит карантин, лечение зараженного файла? (Трубач Г.Г.)
- Существуют ли open source антивирусы? (Трубач Г.Г.)
- Особенности Norton anti-virus? Почему он сейчас один из наиболее популярных? (Трубач Г.Г.)
- Каким образом разрабатывается вредоносное программное обеспечение? (вопрос слишком абстрактный, поэтому конкретизирую) Как разработать своего червя? (Щавровский С.А.)
- Пробовали ли вы создавать вредоносное ПО? (Щавровский С.А.)

- Как выстроена защита от вредоносного ПО на огромных серверах компаний-гигантов (Google, AWS)? (Щавровский С.А.)
- Как антивирусы распознают так называемые "zip-бомбы”? (Ярошевич Я.О.)
- Нужно ли ставить антивирус на телефон? (Для разных ОС) (Ярошевич Я.О.)
- Какие есть самые лучшие антивирусы для ОС Windows? (Ярошевич Я.О.)
- Платили ли вы когда-нибудь за антивирусы? (Ярошевич Я.О.)