1. Основные понятия IT-проекта включают в себя определение целей, описание требований, распределение ролей и ответственностей, планирование ресурсов и времени, разработку и реализацию проекта, а также контроль и оценку его результатов.  
  
2. Компьютерный червь – это вредоносная программа, способная самостоятельно распространяться по сети. Она может внедряться в другие файлы и программы, копировать себя на другие компьютеры, а также выполнять различные деструктивные действия.  
  
3. Компьютерный троян – это вредоносная программа, которая маскируется под полезное приложение или файл, чтобы обмануть пользователей и получить доступ к их системам. Трояны часто служат для сбора и передачи конфиденциальной информации, установки дополнительных вредоносных программ или управления компьютером удаленно.  
  
4. Компьютерный блокировщик экрана (winlocker) – это тип вредоносной программы, которая блокирует доступ к компьютеру или определенным функциям системы, например, экрану входа или файлам. Злоумышленники используют блокировщики экрана для вымогательства денег или выполнения других вредоносных действий.  
  
5. Необходимые условия применения измерительных мониторов включают наличие объекта измерения, доступность измеряемого параметра, возможность установки монитора и его настройку, а также наличие технической инфраструктуры для передачи данных.  
  
6. Измерительные методы используются для получения количественных характеристик объектов и процессов. Они могут включать в себя наблюдение, эксперимент, анкетирование, анализ данных и другие методы.  
  
7. Компоненты схемы измерений включают в себя измерительные приборы, сенсоры, устройства для сбора данных, средства обработки и анализа информации, а также системы хранения и отображения результатов.  
  
8. Аппаратные измерительные мониторы включают в себя физические устройства, такие как датчики, приборы для измерения и регистрации данных, а также устройства для их передачи и обработки. Они используются для контроля и измерения физических параметров в различных областях, таких как производство, наука, медицина и т. д.  
  
9. Измерительные мониторы включают в себя программные и аппаратные средства для сбора, обработки и представления данных. Они могут использоваться для мониторинга и контроля различных процессов и параметров, таких как температура, давление, скорость и другие.  
  
10. Статическое тестирование - это метод тестирования программного обеспечения, при котором исследуется и анализируется его исходный код или другие артефакты без его запуска. Этот метод позволяет выявить ошибки и потенциальные проблемы программы до ее выполнения.  
  
11. Динамическое тестирование - это метод тестирования программного обеспечения, при котором программа запускается и выполняется с целью проверки ее работоспособности и выявления ошибок. Этот метод позволяет провести тщательное тестирование программы в реальных условиях ее использования.  
  
12. Этапы формирования эталонов для тестирования включают в себя определение требований к эталону, сбор и анализ информации, разработку и реализацию эталона, а также его проверку и подтверждение его соответствия требованиям.  
  
13. Классификация анализа вредоносных программ может включать в себя различные аспекты, такие как тип вредоносной программы, ее цель, метод распространения, воздействие на систему и другие характеристики.  
  
14. Этапы изучения вредоносного программного обеспечения включают в себя его анализ и классификацию, изучение его функциональности, исследование его алгоритмов и методов, анализ уязвимостей, а также исследование средств защиты и способов обнаружения.  
  
15. Основные инструменты для исследования кода программ могут включать в себя дизассемблеры, отладчики, статические анализаторы, декомпиляторы и другие программы, которые позволяют изучать и анализировать код программы.  
  
16. Файловый монитор (filemonitor) - это программное обеспечение, которое отслеживает и контролирует доступ и изменение файлов в системе. Он может использоваться для обнаружения и предотвращения несанкционированного доступа к файлам или изменения их содержимого.  
  
17. Методы, применяемые для обнаружения модифицированного кода, могут включать в себя сравнение хэш-сумм файлов, анализ сигнатур и шаблонов вредоносных программ, а также статический и динамический анализ кода. Каждый метод имеет свои преимущества и недостатки, и его применение зависит от конкретной ситуации.  
  
18. Отладчик - это программное средство, которое позволяет программисту исследовать и отслеживать выполнение программы, анализировать ее состояние и исправлять ошибки. Он может включать в себя различные функции, такие как установка точек останова, пошаговое выполнение кода, просмотр значений переменных и другие.  
  
19. Отладочный механизм с трассировкой кода программы позволяет программисту отслеживать выполнение программы по шагам и анализировать ее состояние в различные моменты времени. Принцип работы заключается в записи информации о выполнении кода и его состоянии во время выполнения программы.  
  
20. Отладочный механизм с контрольными точками останова позволяет программисту установить точки останова в программе, где выполнение будет приостановлено, чтобы он мог анализировать состояние программы и выполнять отладку. Принцип работы заключается в установке специальных инструкций или условий, при выполнении которых выполнение программы приостанавливается.  
  
21. Тестирование программы - это процесс проверки программного обеспечения на соответствие требованиям и выявление ошибок или несоответствий. Оно может включать в себя различные методы, такие как функциональное тестирование, интеграционное тестирование, системное тестирование и другие.

22. Понятие анализа программного кода - это процесс изучения и оценки кода с целью выявления его качества, эффективности, безопасности и соответствия стандартам.

23. Понятие и основные отличия тестирования моделей - тестирование моделей включает в себя проверку на соответствие требованиям, проверку на корректность и проверку на полноту. Основное отличие заключается в том, что при тестировании моделей проверяется их работоспособность, а не только код программы.

24. Характеристика и описание процесса тестирования методом «черный ящик» - это метод тестирования, при котором проверяется функциональность программы без знания внутренней реализации. Тестировщики работают с интерфейсом программы, проверяя ее поведение на входных и выходных данных.

25. Характеристика и описание процесса тестирования методом «стеклянный ящик» - это метод тестирования, при котором тестировщики имеют частичное знание о внутренней реализации программы. Они могут использовать это знание для создания более эффективных тестов.

26. Определение и основные особенности для термина: «эмулирующий отладчик» - это инструмент, который позволяет разработчикам эмулировать работу программы для выявления ошибок и отладки.

27. Методы проникновения вредоносного программного обеспечения на персональный компьютер - вредоносное программное обеспечение может проникать на компьютер через вредоносные ссылки, вложения в электронной почте, уязвимости в операционной системе или приложениях, а также через сетевые атаки.

28. Понятие вредоносного программного обеспечения - это программное обеспечение, созданное с целью нанесения вреда компьютеру или пользователям, такое как вирусы, черви, троянские программы и шпионское ПО.

29. Понятие метрики измерений - это числовые показатели, используемые для измерения качества программного обеспечения, такие как производительность, надежность, доступность и эффективность.

30. Хранилище (Repository): основные понятия - это место хранения файлов, кода и другой информации, используемое для управления версиями и совместной работы.

31. Рабочая копия (Working Copy): основные понятия - это локальная копия файлов из репозитория, с которой разработчики могут работать над кодом и вносить изменения.

32. Тестирование ПО

Тестирование программного обеспечения (ПО) - это процесс проверки программы или приложения на соответствие спецификациям и оценка ее функциональности. Цель тестирования ПО состоит в обнаружении ошибок и уверенности в том, что программное обеспечение работает корректно. Виды тестирования могут варьироваться от функционального и производительностного до безопасности и совместимости.

33. Управление контроля версиями (СКВ)

Управление контроля версиями (СКВ) представляет собой систему, предназначенную для отслеживания изменений в исходном коде и других текстовых файлов. СКВ позволяет разработчикам работать параллельно, контролировать версии кода, возвращаться к предыдущим версиям и объединять изменения.

34. Классификация СКВ

СКВ классифицируются на централизованные (CVS, Subversion) и децентрализованные (Git, Mercurial). Централизованные СКВ имеют единую центральную базу данных, в то время как децентрализованные СКВ позволяют каждому пользователю иметь полную копию репозитория.

35. Контроль версий типа SVN

Система контроля версий (СКВ) SVN (Subversion) представляет собой централизованную систему контроля версий, обеспечивающую отслеживание изменений в файлах и директориях. Она также предоставляет возможность восстановления предыдущих версий файлов и отслеживания изменений в коде.

36. Уровни тестирования и цикл тестирования

Уровни тестирования включают модульное тестирование, интеграционное тестирование и системное тестирование. Цикл тестирования представляет собой последовательность этапов, включающую планирование, подготовку, выполнение и анализ результатов тестирования.

37. Импорт и экспорт настроек

Импорт и экспорт настроек позволяют передавать конфигурационные параметры или настройки между различными средами, программами или устройствами.

38. Обратное проектирование

Обратное проектирование (reverse engineering) представляет собой процесс анализа готового продукта с целью понимания его конструкции, строения и принципов работы на основе уже существующего кода или бинарных файлов.

39. Применение и функции обратного проектирования

Обратное проектирование может использоваться для повышения понимания сложных систем, обнаружения ошибок, повышения совместимости и принятия решений о дальнейшем развитии проекта.

40. Описание анализа обратного проектирования

Анализ обратного проектирования позволяет лучше понять структуру и принципы работы уже существующего программного обеспечения, что облегчает его дальнейшую поддержку и развитие.

41. Модульное тестирование

Модульное тестирование - это проверка отдельных модулей программы на корректность их работы. Результатом модульного тестирования является определение соответствия модуля его ожидаемому поведению.

42. Интеграционное тестирование

Интеграционное тестирование включает проверку взаимодействия между различными модулями программы или компонентами системы. Результатом интеграционного тестирования является проверка работоспособности системы в целом.

43. Системное тестирование

Системное тестирование представляет собой проверку программы как единого целого для подтверждения ее соответствия требованиям. Результатом системного тестирования является проверка работоспособности программы или системы в различных условиях.

44. Локальный репозиторий

Локальный репозиторий - это копия удаленного репозитория (например, на GitHub), обычно хранящаяся на компьютере разработчика. Он служит для хранения версий кода и управления изменениями.

45. Тестирование методом "серый ящик"

Тестирование методом "серый ящик" - это метод тестирования программного обеспечения, при котором тестировщики имеют доступ только к интерфейсам программы (как пользователь), без доступа к внутренним деталям ее реализации. Они тестируют поведение системы, не зная ее внутренней структуры.

46. Доступ к GitHub

Для получения доступа к GitHub требуется создать учетную запись на их сайте. Основные функции GitHub включают управление версиями кода, контроль доступа к репозиториям, использование проблем и запросов на объединение, создание страниц вики и др.

47. Организация разработки в сервисе Github

Разработка на GitHub основана на работе с репозиториями. Команда разработчиков может создавать ветви для новых функций или исправлений, предлагать изменения через запросы на объединение, обсуждать изменения и использовать инструменты управления проблемами.

48. Уровни доступа GitHub

GitHub имеет несколько уровней доступа, включая чтение, запись и администрирование репозиториев. Различные роли (например, владелец, участник, коллаборатор) имеют разное количество прав.

49. Способы слияния копий репозитория

Существует несколько способов слияния копий репозитория, включая слияние (merge), rebase, squash и другие. Эти методы используются для объединения изменений из одной ветки в другую.

50. Модели создания кода в репозитории

Модели создания кода в репозитории включают централизованную модель (Centralized Workflow), ветвление по функциональности (Feature Branch Workflow) и др.

51. Архитектура MVC

Архитектура MVC (Model-View-Controller) - это паттерн проектирования, разделяющий приложение на три основных компонента: модель (model), представление (view) и контроллер (controller).

52. Процесс реализации проверки на стороне клиента

Процесс реализации проверки на стороне клиента включает разработку функциональности, выполняемой на стороне клиента, и тестирование этой функциональности для обеспечения корректной работы в различных браузерах и устройствах.

53. JavaScript: инструменты языка

JavaScript имеет множество инструментов, включая среды разработки (IDE), библиотеки (например, React, Angular), фреймворки (например, Node.js) и многое другое.

54. Процесс реализации проверки на стороне клиента. Преимущества проверки на сервере

Процесс реализации проверки на стороне клиента включает создание и тестирование кода, который выполняется на стороне клиента, а не на сервере. Проверка на стороне сервера обеспечивает большую безопасность и контроль доступа к данным.

55. Библиотека классов

Библиотека классов - это набор классов и методов программной библиотеки, предназначенный для решения определенных задач или обеспечения определенной функциональности.

56. Язык Ассемблера

Возможности: Язык ассемблера предоставляет возможность написания программ, близких к машинному коду, что делает его более эффективным по сравнению с высокоуровневыми языками. Он также обеспечивает прямой доступ к ресурсам компьютера.

Преимущества: Эффективное использование ресурсов, простота перевода в машинный код, непосредственное управление аппаратурой.

Недостатки: Сложность написания и отладки, зависимость от конкретной архитектуры процессора, низкий уровень абстракции, что делает его менее понятным для разработчиков, привыкших к высокоуровневым языкам.

57. Сегментная адресация

Определение: Способ организации памяти, при котором она разделяется на сегменты, каждый из которых может иметь свой размер и признаки доступа.

Примеры применения: Сегментная адресация используется в различных архитектурах процессоров, таких как x86, для управления памятью и обеспечения изоляции процессов друг от друга.

58. Дизассемблер

Возможности: Преобразование машинного кода или исполняемых файлов в исходный код на языке ассемблера или другом читаемом виде для облегчения понимания работы программы или обнаружения ошибок.

Преимущества: Позволяет анализировать работу программы на более низком уровне и восстанавливать исходный код.

Недостатки: Полученный исходный код может быть сложным для понимания из-за отсутствия комментариев и человекочитаемых имен переменных и функций.

59. Мониторинг работоспособности системы контроля версий

Назначение: Мониторинг работоспособности системы контроля версий предназначен для отслеживания доступности и производительности системы контроля версий, включая анализ использования ресурсов.

Инструменты мониторинга: Для мониторинга систем контроля версий часто используются инструменты для мониторинга сети, производительности серверов и баз данных.

60. Коллективная разработка

Характеристика: Коллективная разработка предполагает работу над IT-проектом группой разработчиков, обычно используя системы контроля версий, согласованное планирование и коммуникацию для совместного создания программного обеспечения.

61. Проверка сервера

Характеристика: Проверка сервера включает анализ работоспособности и производительности сервера, обеспечение безопасности, контроль доступности и надежности серверных процессов.

62. Проверка клиента

Характеристика: Проверка клиента включает контроль работоспособности клиентского программного обеспечения, его интерфейса, удобства использования и адаптивности к различным условиям.

63. Отличительные особенности IDE C++/C#

IDE для C++ может обеспечивать подсветку синтаксиса, автодополнение кода, управление проектами и отладку на низком уровне. IDE для C# может предоставлять интегрированное окружение для разработки, различные инструменты для проектирования интерфейсов и т. д.

64. Отличительные особенности IDE Java

IDE для Java часто предоставляет возможности разработки, отладки, профилирования, создания пользовательского интерфейса и интеграции с популярными фреймворками и библиотеками Java.

65. Отключение JavaScript на стороне клиента

Особенности: При отключении JavaScript на стороне клиента могут быть ограничены функции и возможности взаимодействия с веб-сайтами, использующими большое количество JavaScript.

66. Команды GitHub: git init, git branch, git push/pull

Характеристика: git init создает новый репозиторий, git branch создает новую ветку, git push отправляет изменения в удаленный репозиторий, а git pull загружает изменения из удаленного репозитория.

67. Приемочное тестирование

Основные понятия: Приемочное тестирование предназначено для проверки соответствия разработанного продукта изначально выдвинутым требованиям заказчика.

68. Выходное тестирование

Основные понятия: Выходное тестирование (иногда называемое тестированием приемки) - это проверка готового продукта перед его внедрени

ем для уверенности в его работоспособности и соответствии требованиям.

69. Основные элементы для тестирования ПО

Результат: Основные элементы тестирования ПО включают в себя создание тестовых случаев (test cases), выполнение тестов и анализ результатов, обеспечивая корректность функционирования программного обеспечения.

70. Команды git rebase, git clone, git log

Характеристика: git rebase используется для переноса или комбинирования изменений; git clone создает копию репозитория на локальной машине; git log показывает историю коммитов.