***Теоретические вопросы:***

1. IT-проект: основные понятия.
2. Компьютерный червь: свойства, особенности.
3. Компьютерный троян: свойства, особенности.
4. Компьютерный блокировщик экрана (WinLocker): свойства, особенности.
5. Необходимые условия применения измерительных мониторов.
6. Измерительные методы: назначение.
7. Описание компонентов схемы измерений.
8. Аппаратные измерительные мониторы: виды, основные назначения.
9. Измерительные мониторы: виды, основное назначение.
10. Статическое тестирование: общие сведения, значение.
11. Динамическое тестирование: общие сведения, значение.
12. Этапы формирования эталонов для тестирования и их описание.
13. Классификация анализа вредоносных программ.
14. Этапы изучения вредоносного программного обеспечения и их описание.
15. Основные инструменты для исследования кода программ: виды, свойства, значение.
16. Файловый монитор (FileMonitor): возможности, предназначение.
17. Методы, применяемые для обнаружения модифицированного кода. Преимущества и недостатки их применения.
18. Понятие отладчика. Назначение, виды.
19. Отладочный механизм: трассировка кода программы. Назначение, принцип работы.
20. Отладочный механизм: контрольные точки останова. Назначение, принцип работы.
21. Понятие тестирования программ. Назначение, методы.
22. Понятие анализа программного кода.
23. Понятие и основные отличия тестирования моделей.
24. Характеристика и описание процесса тестирования методом «черный ящик».
25. Характеристика и описание процесса тестирования методом «стеклянный ящик».
26. Определение и основные особенности для термина: «эмулирующий отладчик».
27. Методы проникновения вредоносного программного обеспечения на персональный компьютер.
28. Понятие вредоносного программного обеспечения.
29. Понятие метрики измерений.
30. Хранилище (Repository): основные понятия.
31. Рабочая копия (Working Copy): основные понятия.
32. Тестирование ПО общая характеристика, понятия, виды.
33. Понятия управления контроля версиями, применение и функции системы контроля версий.
34. Классификация СКВ.
35. Смысл контроля версий типа SVN. Современные подходы к анализу ПО.
36. Уровни тестирования: описание. Цикл тестирования: иерархическое древо: графическое отображение.
37. Импорт и экспорт настроек: общие сведения, применение.
38. Обратное проектирование: основные понятия, виды применяемых инструментов и описание процесса.
39. Применение и функции обратного проектирования.
40. Описание анализа обратного проектирования.
41. Модульное тестирование: основные понятия, элементы, результат.
42. Интеграционное тестирование: основные понятия, элементы, результат.
43. Системное тестирование: основные понятия, элементы, результат.
44. Локальный репозиторий: определение и функции.
45. Понятие тестирования методом «серый ящик» и описание процесса.
46. Получение доступа к GitHub. Основные функции и принцип GitHub.
47. Организация разработки в сервисе Github.
48. Описание уровней доступа GitHub.
49. Характеристика способов слияния копий репозитория.
50. Модели создания кода в репозитории: описание и графическое представление.
51. Архитектура MVC: основные понятия, структура.
52. Описание процесса реализации проверки на стороне клиента.
53. JavaScript: инструменты языка. Состав среды разработки (IDE).
54. Описание процесса реализации проверки на стороне клиента Преимущества проверки на сервере.
55. Библиотека классов: понятие, доступ к членам класса.
56. Язык Ассемблера: возможности языка, преимущества и недостатки.
57. Сегментная адресация: определение, примеры применения.
58. Дизассемблер: возможности программы, преимущества и недостатки.
59. Мониторинг работоспособности системы контроля версий: назначение и инструменты мониторинга.
60. Коллективная разработка: характеристика, структура.

Ответы:

1. IT-проект: основные понятия - IT-проект это процесс разработки и внедрения информационных технологий для решения определенной задачи или достижения целей.

2. Компьютерный червь: свойства, особенности - компьютерный червь это вредоносная программа, способная самостоятельно распространяться по компьютерным сетям, копируя себя на другие устройства без ведома пользователя и нанося ущерб системе.

3. Компьютерный троян: свойства, особенности - компьютерный троян это вредоносная программа, которая маскируется под полезное приложение или файл, чтобы получить несанкционированный доступ к компьютеру и осуществлять различные вредоносные действия без ведома пользователя.

4. Компьютерный блокировщик экрана (WinLocker): свойства, особенности - компьютерный блокировщик экрана это вредоносная программа, которая блокирует доступ пользователя к компьютеру, заставляя его вводить определенные данные или платить определенную сумму, чтобы вернуть контроль над устройством.

5. Необходимые условия применения измерительных мониторов - наличие объекта, который требуется измерять, наличие датчиков и измерительного оборудования, способного точно и надежно измерять параметры объекта.

6. Измерительные методы: назначение - измерительные методы используются для получения количественной информации об объекте, оценки его характеристик, контроля и анализа процессов и результатов.

7. Описание компонентов схемы измерений - компоненты схемы измерений включают объект измерения, датчики для снятия данных, измерительные приборы, способы передачи и обработки данных.

8. Аппаратные измерительные мониторы: виды, основные назначения - аппаратные измерительные мониторы могут быть различных типов, таких как осциллографы, спектроанализаторы, мультиметры и т.д., и используются для измерения и контроля различных параметров в электронике, электротехнике и других областях.

9. Измерительные мониторы: виды, основное назначение - измерительные мониторы могут быть как аппаратными, так и программными, и основное их назначение заключается в измерении и контроле различных параметров, таких как температура, давление, скорость, потребление энергии и другие.

10. Статическое тестирование: общие сведения, значение - статическое тестирование это метод тестирования программного обеспечения, при котором анализируется исходный код без его запуска, что позволяет выявить ошибки, недочеты и потенциальные проблемы.

11. Динамическое тестирование: общие сведения, значение - динамическое тестирование это метод тестирования программного обеспечения, при котором тестируемая программа запускается и исполняется, чтобы проверить ее работоспособность, корректность и соответствие требованиям.

12. Этапы формирования эталонов для тестирования и их описание - этапы формирования эталонов включают выбор тестовых данных и сценариев, создание эталона, проверку и валидацию эталона. Эталон представляет собой набор ожидаемых результатов от программного продукта и используется для сравнения с фактическими результатами при тестировании.

13. Классификация анализа вредоносных программ - анализ вредоносных программ может быть классифицирован как статический (анализ исходного кода или исполняемых файлов без их запуска) и динамический (анализ работы программы в режиме выполнения).

14. Этапы изучения вредоносного программного обеспечения и их описание - этапы изучения вредоносного программного обеспечения включают анализ кода, выявление функциональности и намерений программы, выявление возможных уязвимостей и способов защиты от вредоносных эффектов.

15. Основные инструменты для исследования кода программ: виды, свойства, значение - к основным инструментам исследования кода программ относятся отладчики, дизассемблеры, профилировщики, статические и динамические анализаторы и другие. Эти инструменты помогают анализировать, отлаживать и оптимизировать код программы, а также выявлять потенциальные уязвимости и проблемы в нем.

16. Файловый монитор (FileMonitor) - это программное обеспечение, которое отслеживает и записывает все изменения, происходящие с файлами на компьютере. Он предназначен для обеспечения безопасности и контроля файловой системы.

17. Методы обнаружения модифицированного кода включают статический и динамический анализ, сигнатурное сопоставление, эвристический анализ и др. Преимущества и недостатки их применения зависят от конкретных условий, но в общем статический анализ может быть более надежным, но требует знания кода, в то время как динамический анализ может быть более гибким.

18. Отладчик - это инструмент для анализа и исправления ошибок в программном коде. Он позволяет программистам проверить состояние переменных, изменять значения, выполнять код пошагово и т.д. Виды отладчиков включают символьные отладчики, эмуляторы и исследовательские отладчики.

19. Трассировка кода программы - это отладочный механизм, который записывает последовательность выполнения кода программы. Его назначение состоит в том, чтобы помочь программисту выявить ошибки и найти место их возникновения. Принцип работы трассировки кода заключается в записи информации о выполненных операциях и последовательности их выполнения.

20. Контрольные точки останова - это отладочный механизм, который позволяет программисту установить места в программе, при достижении которых выполнение программы останавливается. Назначение контрольных точек состоит в том, чтобы анализировать промежуточные результаты и состояние программы в определенных точках. Принцип работы контрольных точек заключается в приостановке выполнения программы при достижении определенного адреса или условия.

21. Тестирование программ - это процесс проверки программного кода на наличие ошибок, дефектов и соответствие требованиям. Назначение тестирования программ заключается в обеспечении качества программного обеспечения, выявлении и исправлении ошибок. Методы тестирования включают модульное, интеграционное, системное тестирование и др.

22. Анализ программного кода - это процесс изучения и оценки кода программы с целью выявления ошибок, дефектов, потенциальных уязвимостей и улучшения его качества. Он включает в себя просмотр кода, статический анализ, метрики кода и др.

23. Тестирование моделей - это процесс проверки соответствия созданных моделей требованиям и выявления ошибок и дефектов в них. Отличия тестирования моделей от тестирования программ заключаются в том, что модели могут иметь более абстрактный уровень и не включать в себя полный код программы.

24. Метод "черный ящик" - это метод тестирования программы, при котором тестировщик не имеет доступа к внутренним структурам и деталям программы. Он основывается на входных и выходных данных программы, и тестировщики проверяют только результаты работы программы, не обращая внимание на то, как она достигла этих результатов.

25. Метод "стеклянный ящик" - это метод тестирования программы, при котором тестировщик имеет доступ к внутренним структурам и деталям программы. Он основывается на знании внутренней структуры программы и позволяет проводить более глубокий анализ кода и обнаруживать проблемы, которые не могут быть выявлены при использовании метода "черный ящик".

26. "Эмулирующий отладчик" - это отладчик, который используется для эмуляции или имитации определенных условий или окружения, которые могут возникнуть при выполнении программы. Он позволяет программисту проверить работу программы в различных ситуациях и обнаружить возможные ошибки или проблемы.

27. Методы проникновения вредоносного программного обеспечения на персональный компьютер включают использование уязвимостей операционной системы, социальная инженерия, фишинг, вредоносные вложения в электронных письмах и др.

28. Вредоносное программное обеспечение - это програмное обеспечение, созданное с целью нанесения вреда или выполнения незаконных действий на компьютере пользователя. Оно может включать в себя вирусы, троянские программы, черви, шпионы и другие вредоносные программы.

29. Метрика измерений - это инструмент для количественной оценки и анализа характеристик или процессов

30. Хранилище (Repository): основные понятия.

Хранилище - это централизованное место, где хранятся все версии файлов и папок проекта. Оно позволяет управлять изменениями, отслеживать историю изменений, а также обеспечивает возможность работы в команде над одним проектом.

31. Рабочая копия (Working Copy): основные понятия.

Рабочая копия - это локальная копия файлов и папок из репозитория, которая хранится на компьютере пользователя. Она позволяет вносить изменения в файлы, добавлять новые файлы и синхронизироваться с репозиторием для получения последних обновлений.

32. Тестирование ПО: общая характеристика, понятия, виды.

Тестирование ПО - это процесс проверки программного обеспечения на соответствие требованиям, выявление ошибок, анализ качества и оценка работоспособности. Оно может включать такие виды тестирования, как функциональное тестирование, нагрузочное тестирование, автоматизированное тестирование и другие, в зависимости от целей и требований проекта.

33. Понятия управления контроля версиями, применение и функции системы контроля версий.

Система контроля версий (СКВ) - это инструмент, который позволяет отслеживать изменения в файлах и папках проекта, управлять версиями, сравнивать и объединять изменения, а также восстанавливать предыдущие версии файлов. Применение СКВ особенно полезно при работе в команде, когда несколько людей одновременно вносят изменения в проект.

34. Классификация СКВ.

Системы контроля версий можно классифицировать на централизованные (Centralized VCS) и распределенные (Distributed VCS). В централизованных СКВ все файлы хранятся на одном сервере, а пользователи работают с локальными копиями. В распределенных СКВ каждый пользователь имеет свою копию репозитория, что позволяет работать без подключения к серверу и совместно разрабатывать проекты.

35. Смысл контроля версий типа SVN. Современные подходы к анализу ПО.

SVN (Subversion) - это система контроля версий с централизованной моделью. Она позволяет отслеживать изменения в файлах, работать в команде, синхронизироваться с репозиторием и управлять версиями. Современные подходы к анализу ПО включают использование автоматизированных инструментов для тестирования и отладки, анализа кода и выявления потенциальных ошибок для повышения качества и надежности программного обеспечения.

36. Уровни тестирования: описание. Цикл тестирования: иерархическое древо: графическое отображение.

Уровни тестирования включают модульное тестирование (тестирование отдельных модулей программы), интеграционное тестирование (проверка взаимодействия между модулями), системное тестирование (проверка работы всей системы), приемочное тестирование (тестирование на соответствие требованиям заказчика) и другие. Цикл тестирования представляет собой последовательность этапов тестирования, включающую планирование, подготовку, выполнение тестов, анализ результатов и отчетность. Иерархическое древо в графическом отображении показывает взаимосвязь и зависимость между различными видами тестирования.

37. Импорт и экспорт настроек: общие сведения, применение.

Импорт и экспорт настроек - это процесс сохранения и загрузки настроек программного обеспечения. Это позволяет переносить настройки между различными устройствами или использовать их на других компьютерах. Например, можно экспортировать настройки программы на одном компьютере, а затем импортировать их на другом компьютере, чтобы быстро восстановить используемые настройки.

38. Обратное проектирование: основные понятия, виды применяемых инструментов и описание процесса.

Обратное проектирование - это процесс анализа уже существующего программного обеспечения с целью понимания его структуры, функциональности и взаимодействия компонентов. Виды применяемых инструментов включают декомпиляторы, статический анализатор кода, визуализаторы и другие инструменты для анализа и восстановления структуры программы. Процесс обратного проектирования включает анализ кода, выделение компонентов, построение диаграмм и документации.

39. Применение и функции обратного проектирования.

Обратное проектирование позволяет разработчикам понять уже суще

Обратное проектирование - это процесс анализа и изучения готового продукта или системы для создания его модели или документации. Основная цель обратного проектирования состоит в том, чтобы понять, как работает продукт или система, и использовать полученные знания для улучшения или модификации существующего продукта.

Функции обратного проектирования включают:

1. Понимание функциональности: обратное проектирование помогает разобраться в структуре и функциональности готового продукта или системы.

2. Модификация и улучшение: обратное проектирование позволяет внести изменения в существующий продукт или систему для улучшения его характеристик или функциональности.

3. Разработка аналогов: на основе анализа готового продукта или системы можно создать аналогичный или более продвинутый продукт или систему.

4. Разработка документации: обратное проектирование помогает создавать документацию, необходимую для понимания, сопровождения и ремонта готового продукта или системы.

5. Диагностика и устранение неисправностей: анализ готового продукта или системы может помочь выявить проблемы и неисправности и предложить решения для их устранения.

В целом, обратное проектирование позволяет разобраться в работе готового продукта или системы и использовать полученные знания для оптимизации и улучшения.

40. Анализ обратного проектирования - это процесс анализа существующего кода или системы с целью понять их функциональность, структуру, зависимости и другие характеристики. Это позволяет разработчикам лучше понять и улучшить код, а также создать документацию и поддерживать легаси-код.

41. Модульное тестирование - это процесс проверки отдельных модулей программного обеспечения на соответствие заданным требованиям. Он включает в себя создание тестовых сценариев и выполнение тестовых случаев для каждого модуля. Результатом модульного тестирования является определение наличия дефектов в модуле и его работоспособности.

42. Интеграционное тестирование - это процесс проверки взаимодействия между различными модулями или компонентами программного обеспечения. Оно включает в себя проверку передачи данных между модулями, обработку ошибок и согласованность поведения системы. Результатом интеграционного тестирования является определение наличия дефектов взаимодействия между модулями и их исправление.

43. Системное тестирование - это процесс проверки всей системы в целом на соответствие требованиям и ожиданиям. Оно выполняется после интеграционного тестирования и включает в себя проверку всех функциональных и нефункциональных аспектов системы. Результатом системного тестирования является определение наличия дефектов в системе и ее готовности к применению.

44. Локальный репозиторий - это копия удаленного репозитория, которая хранится на локальном компьютере разработчика. Он используется для сохранения версий кода и управления изменениями. Локальный репозиторий позволяет разработчикам работать над проектом без подключения к сети и обмениваться изменениями с удаленным репозиторием при необходимости.

45. Тестирование методом "серый ящик" - это подход к тестированию программного обеспечения, при котором тестировщик имеет доступ к внутреннему коду или структуре системы. Процесс включает в себя анализ внешних входных данных и ожидаемых результатов, а также проверку работы системы в соответствии с описанными требованиями и спецификациями.

46. Получение доступа к GitHub - это процесс создания учетной записи на платформе GitHub и получение прав доступа к репозиториям и другим функциям. Основные функции GitHub включают в себя управление версиями кода, совместную работу разработчиков, создание запросов на слияние, отслеживание задач и управление проектами.

47. Организация разработки в сервисе GitHub включает создание репозиториев для хранения кода и управления его версиями, создание задач, создание веток для разработки новых функций и исправления дефектов, а также совместную работу и общение разработчиков через инструменты GitHub, такие как запросы на слияние и комментарии к коду.

48. Уровни доступа GitHub определяют различные права доступа, которыми могут обладать участники репозитория. Это включает в себя владельца репозитория, который имеет полные права управления репозиторием, администраторов, которые могут управлять настройками репозитория, коллабораторов, которые могут вносить изменения в репозиторий, и наблюдателей, которые могут просматривать код и участвовать в обсуждениях, но не могут его изменять.

49. Способы слияния копий репозитория включают в себя ветвление и слияние (Branch and Merge), перебазирование (Rebase) и объединение (Squash). Ветвление и слияние позволяют создавать отдельные ветки для разработки новых функций и сливать их с главной веткой. Перебазирование позволяет переместить изменения из одной ветки на другую. Объединение позволяет объединить несколько коммитов в один для улучшения читаемости и поддерживаемости истории изменений.

50. Модели создания кода в репозитории - это способы организации кода в репозитории, которые могут быть использованы разработчиками. Некоторые из популярных моделей включают централизованную модель, где все разработчики работают с одним центральным репозиторием, и модель ветвления и слияния, где каждый разработчик создает свою собственную ветку для работы и сливает ее с главной веткой при завершении.

51. Архитектура MVC - Model-View-Controller (MVC) - это структурный шаблон проектирования программного обеспечения, который разделяет код на три основных компонента: модель (отвечающая за обработку данных и бизнес-логику), представление (отображение данных пользователю) и контроллер (управление взаимодействием между моделью и представлением). Эта архитектура помогает разделить логику приложения и улучшить его модульность и поддерживаемость.

52. Процесс реализации проверки на стороне клиента включает создание скриптов на стороне клиента (например, веб-страницы или приложения), которые проверяют вводимые пользователем данные перед отправкой на сервер. Это позволяет предотвратить отправку неверных или вредоносных данных на сервер и улучшить качество и безопасность приложения.

53. JavaScript - это язык программирования, который используется для создания интерактивных веб-страниц и веб-приложений. Он включает в себя инструменты для работы с HTML и CSS, а также для обработки событий, манипулирования DOM и взаимодействия с сервером. Некоторые популярные инструменты для разработки на JavaScript включают Visual Studio Code, Sublime Text и WebStorm.

54. Преимущества проверки на стороне клиента включают более быструю обратную связь для пользователя, уменьшение нагрузки на сервер и экономию сетевого трафика. Однако, проверка на стороне клиента может быть обходной пользователем и не гарантировать безопасности данных. Поэтому, важно также выполнять проверки на серверной стороне для обеспечения надежности и безопасности.

55. Библиотека классов - это коллекция классов, которые предоставляют функциональность и возможности для разработчика. Она включает в себя различные классы с методами и свойствами, которые можно использовать в приложениях для решения разных задач. Доступ к членам класса осуществляется через инкапсуляцию, с помощью публичных и приватных методов и свойств класса.

56. Язык Ассемблера - низкоуровневый язык программирования, который использует мнемоники и команды, понятные процессору. Он позволяет разработчикам писать программы, взаимодействуя с аппаратной частью компьютера без промежуточных уровней абстракции. Преимуществами языка Ассемблера являются более эффективное использование ресурсов компьютера и возможность точного управления аппаратурой, недостатками - сложность и трудность в разработке и поддержке программ.

57. Сегментная адресация - это метод организации памяти компьютера, при котором адресация данных осуществляется с использованием сегментов. Каждый сегмент имеет свой базовый адрес, с которого начинается адресация данных внутри сегмента. Примеры применения сегментной адресации включают разделение памяти на кодовый сегмент, системные сегменты и сегменты данных.

58. Дизассемблер - это программа, которая преобразует машинный код или исполняемый файл в читаемый для человека вид. Дизассемблер позволяет разработчикам анализировать и понимать работу программы на машинном уровне, исследовать алгоритмы и структуру кода. Помимо преимущества в изучении и дизассемблировании программ, дизассемблер

59. Мониторинг работоспособности системы контроля версий (СКВ) необходим для проверки и отслеживания состояния и производительности СКВ. Инструменты мониторинга могут включать:

- мониторинг доступности сервера, на котором работает СКВ;

- проверка целостности и доступности репозитория;

- контроль использования ресурсов сервера, таких как CPU и память;

- отслеживание активности пользователей и работы с репозиторием;

- информирование о возможных проблемах или неполадках в СКВ.

60. Коллективная разработка - это процесс, в котором группа разработчиков работает над одним проектом, используя систему контроля версий (СКВ). Характеристики коллективной разработки:

- совместный доступ к коду и другим ресурсам проекта;

- возможность одновременной работы нескольких разработчиков;

- совместное управление изменениями и конфликтами внутри проекта;

- обмен знаниями и опытом между участниками проекта;

- повышение производительности и качества проекта благодаря совместной работе и контролю версий;

- распределенная структура, где каждый разработчик может иметь свою собственную копию репозитория.