

УтверждАЮ

Зам. директора по ОУП

\_\_\_\_\_\_\_\_\_\_\_И.Г. Бозрова

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_2022 г.

**ВОПРОСЫ К КОМПЛЕКСНОМУ ЭКЗАМЕНУ**

по междисциплинарным курсам

МДК.3.2 Управление проектами

и МДК.3.1 Моделирование и анализ программного обеспечения

профессионального модуля:

ПМ. 03 Ревьюирование программных модулей

специальность 09.02.07 Информационные системы и программирование (квалификация: специалист по информационным ресурсам)

курс 2

Преподаватель первой категории \_\_\_\_\_\_\_\_\_\_\_/Д.А. Голиков

Преподаватель \_\_\_\_\_\_\_\_\_\_\_/Б.И. Боклач

***Теоретические вопросы:***

1. **IT-проект: основные понятия.**

В системе управления проектом реализуются 2 группы процессов:

* проектно-ориентированные процессы - связаны с объектом УП, т.е. с самим проектом, выполняются исполнителями работ проекта и направлены на достижение результатов проекта — создание нового продукта или услуги. Они относятся преимущественно к производственно-технологическим, определяются жизненным циклом проекта и зависят от сферы приложения, типа и вида проекта.
* процессы управления проектом - связаны с субъектом УП, т.е. командой управления проектом и ее деятельностью по описанию, планированию, организации и координации работ в проекте для обеспечения его успешного завершения. Они относятся преимущественно к организационно-экономическим и направлены на решение задач, связанных с реализацией различных функций управления проектом.

Информационные технологии (ИТ) – это процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов. Как отрасль человеческой деятельности информационные технологии занимаются созданием, внедрением, эксплуатацией и развитием информационных систем.

Информационная система (ИС) – система обработки информации и соответствующие организационные ресурсы (человеческие, технические, финансовые и т. д.), которые обеспечивают и распространяют информацию (ISO/IEC 2382-1:1993[11]).

Можно выделить следующие направления в ИТ:

* разработка и развитие программного обеспечения;
* внедрение информационных систем;
* развитие и модернизация вычислительных мощностей и инфраструктуры передачи данных.

ИТ-проект *–* это краткосрочное усилие по созданию уникального продукта, сервиса или среды, например, замещение старых сервисов новыми, разработка коммерческого сайта, создание новых видов настольных компьютеров или слияние баз данных. Процессы управления ИТ-проектами в компании часто имеют сложный и многоступенчатый характер.

Процессы управления проектом объединены в пять групп:

* инициация,
* планирование,
* исполнение,
* мониторинг и контроль,
* закрытие

1. **Вредоносное программное обеспечения типа «червь»: свойства, особенности.**

**Сетевой червь** – программа, способная к самораспространению путем многократного самокопирования и передаче в компьютерных сетях. Черви используют для своего распространения компьютерные и мобильные сети так же, как вирусы используют файлы, и могут рассматриваться как разновидность компьютерных вирусов.

Одной из отличительных особенностей сетевых червей является возможность их чрезвычайно быстрого распространения, что может приводить к массовым компьютерным вирусным эпидемиям. Основной путь распространения сетевых червей – электронная почта (почтовые черви – mail worms), но также черви могут использовать файлообменные (P2P) и IRC-сети, LAN, сети обмена данными между мобильными устройствами (смартфонами, карманными компьютерами) и т. д. Большинство червей распространяется в виде файлов как вложение в электронное письмо, ссылка на зараженный файл на Web- или FTP-ресурсе, в ICQ- и IRC-сообщениях, файл в каталоге обмена P2P и т. д. Некоторые, так называемые «бесфайловые» или «пакетные» черви, распространяются в виде сетевых пакетов и, проникая непосредственно в память компьютера, активизируют свой код

Компьютерный вирус и компьютерный червь — это вредоносные программы, которые способны воспроизводить себя на компьютерах или через компьютерные сети. в виде файла, отправленного во вложении в электронном письме; в виде ссылки на интернет - или FTP-ресурс; в виде ссылки, переданной через сообщение ICQ или IR; через пиринговые сети обмена данными P2P (peer-to-peer); некоторые черви распространяются как сетевые пакеты. Они проникают прямо в компьютерную память, затем активируется код червя.

1. Вредоносное программное обеспечения типа «Trojan»: свойства, особенности.

**Троянская программа** (программа «Троянский конь», троян, троянец) **-**компьютерная программа, скрытно осуществляющая несанкционированные пользователем действия. «Классический» троян запускает скрытую внутри какой-либо легальной программы несанкционированную функцию, обеспечивающую выполнение действий, не предусмотренных автором легальной программы. Как правило, трояны используются как скрытые программы, предназначенные для несанкционированного получения обрабатываемой или хранимой в компьютере информации и передачи ее злоумышленнику, что может приводить к потере конфиденциальности информации. Например, троянcкие программы-клавиатурные мониторы способны перехватывать все нажатия клавиш клавиатуры, записывать эту информацию и передавать ее по сети. Некоторые разновидности троянов могут вызывать уничтожение или модификацию информации, нарушение работоспособности компьютера, использование ресурсов компьютера в несанкционированных целях. Существуют разновидности троянов, не приносящие вреда зараженному компьютеру, но наносящие ущерб удаленным компьютерным системам, как, например, троянские программы, разработанные для массированных DoS-атак (Denial of Service – отказ в обслуживании) на удалённые ресурсы сети. Как и черви, троянские программы иногда рассматриваются как отдельная разновидность компьютерных вирусов.

Это вредоносный агент, основное отличие которого от классического вируса состоит в методе распространения: обычно он проникает в систему под видом обычной, легитимной программы, чем и обусловлена традиция называть его «троянским конем». После проникновения он способен на многое: может собирать информацию об устройстве и его владельце, воровать хранящиеся на компьютере данные, блокировать доступ к пользовательской информации, выводить операционную систему из строя и т.п.

1. **Вредоносное программное обеспечения типа «блокировщик экрана (WinLocker)»: свойства, особенности.**

Это вредоносный агент, основное отличие которого от классического вируса состоит в методе распространения: обычно он проникает в систему под видом обычной, легитимной программы, чем и обусловлена традиция называть его «троянским конем». После проникновения он способен на многое: может собирать информацию об устройстве и его владельце, воровать хранящиеся на компьютере данные, блокировать доступ к пользовательской информации, выводить операционную систему из строя и т.п. Распространяется он обычно посредством различных электронных сообщений, которые в той или иной мере интересуют пользователя.

1. **Необходимые условия применения измерительных мониторов.**

Необходимые условия применения измерительных методов: наличие готовой программы подлежащей измерительному исследованию, наличие реальной вычислительной системы для прогона программы, наличие аппаратных или программных средств проведения измерений, создание условий снижения искажений вносимых в функционирование системы в процессе проведения измерений.

1. **Измерительные методы: назначение.**

**В целом измерительные методы имеют следующее назначение:**

1) Измерение параметров потребления программами ресурсов вычислительной системы с целью устранения дефектов производительности.

2) Предварительное измерение параметров системы для имитационных или аналитических моделей программ перед их последующим использованием.

**Необходимые условия применения измерительных методов:**

1) Наличие готовой программы, подлежащей измерительному исследованию;

2) Наличие реальной вычислительной системы (а не её модели) для прогона программы;

3) Наличие аппаратных или программных средств проведения измерений;

4) Создание условий снижения искажений, вносимых в функционирование системы в процессе проведения измерений, до приемлемого уровня.

Измерение параметров потребления программами ресурсов вычислительной системы с целью дефектов производительности, предварительное измерение параметров системы для имитационных или аналитических моделей программ перед их следующим использованием, проверка адекватности имитационных или аналитических моделей и методов расчёта характеристик выполнения программ по результатам моделирования

1. **Описание компонентов схемы измерений.**

**Схема измерений включает следующие компоненты:**

* Исследуемую вычислительную систему с установленными программами
* Средства регистрации параметров потребляемых ресурсов при выполнении данной рабочей нагрузки.
* Архив для хранения результатов многочисленных измерений
* Результаты измерений обрабатываются некоторой вычислительной системой ВС (отдельная ВС или та же, на которой снимались измерения, но после выполнения сеанса измерений)
* Рабочая нагрузка – одна или несколько программ или наборов данных для получения статистики проводимых измерений

1. **Аппаратные измерительные мониторы: виды, основные назначения.**

**АИМ подразделяется на встроенные и автономные:**  
Встроенные АИМ – включаются в аппаратуру системы заводом изготовителем, как правило, для выполнения тестовых измерений в фиксированном наборе внутренних точек устройств системы. Основное назначение встроенного АИМ – для проверки, контроля и настройки ВС. Но так как в его составе могут находиться триггеры состояний схемы, счётчики и некоторые другие устройства, то они могут использоваться и для измерения параметров программ. Автономные АИМ подключаются к измерительным точкам извне системы через специальные разъёмы и могут задавать любые точки доступные для измерений. Автономные АИМ наиболее общий случай.

1. **Измерительные мониторы: виды, основное назначение.**

АИМ (Аппаратные измерительные мониторы) - подключаются к точкам зондирования и измеряют электрические сигналы, соответствующие параметрам ресурсов. Встроенные заводом-изготовителем, выполняющие фиксирование параметров в фиксированном числе точек с целью настройки и тестирования системы. Большей частью следят за аппаратными частями, но могут измерять параметры программного обеспечения, расположенные в памяти. Более широко используются аппаратные мониторы, подключаемые извне системы.  
ГИБРИДНЫЕ - обычно программным способом выполняется регистрация событий, связанных с выполнением программ, а аппаратным – параметры при наступлении событий. Бывают встроенные и автономные. Встроенные отличаются микрокодом. Микропрограммы служат для регистрации событий и параметров. Работают быстрее программных. Могут иметь доступ к любым точкам зондирования. Недостаток – малая гибкость из-за малых возможностей изменения микрокода. У автономных наличие измерительного компьютера наряду с измеряемым. Можно изменять программу измерений и по линиям связи с измеряемой системы получать доступ к измерению любых ресурсов.  
ПИМ (Программные измерительные мониторы) – это совокупность команд или программ, выполняемых на системе исключительно с целью измерения параметров ресурсов. Встроенные ПИМ включаются в состав ОС на этапе генерации. Используются для регистрации фиксированного набора параметров при фиксированном объеме событий. Обычно это журналы регистрации событий. Их задача – определить потребление ресурсов той или иной программой.

1. **Статическое тестирование: общие сведения, значение.**

Статическое тестирование – ручное тестирование программ, начиная со стадии формирования требований к программе. На стадии кодирования при статическом тестировании некоторую часть маршрутов исполнения тестируют вручную.

1. **Динамическое тестирование: общие сведения, значение.**

Динамическое тестирование – подразумевает достаточно полную структурную и функциональную проверку выполнения программы.

1. **Этапы формирования эталонов для тестирования и их описание.**

Существуют несколько способов формирования эталонов:

* **Использование аналитических выражений.** Этот способ особенно подходит при детерминированном тестировании, так как имеется возможность сравнить результаты тестирования с ожидаемыми результатами. Имеются ограничения в использовании этого метода, если неизвестны или отсутствуют аналитические выражения связывающие входные данные и результаты, иногда требуется использовать много допущений.
* **Использование моделирования на ЭВМ.** Способ является универсальным. При этом ряд данных моделируется другими способом и по другим алгоритмом, нежели испытываемая программа и на других ЭВМ. Причем набор входных данных создаются по случайным законам, что обеспечивает высокую гибкость этого способа.
* **Использование результатов испытаний предшествующих вариантов программ.** При этом используются ранее накопленный опыт испытателя или других исследователей, выраженный в экстренных оценках ожидаемых результатов. Степень достоверности проверки корректности программ при использовании этих методов убывает по номерам способов формирования эталонов. В 1-ом случае обеспечивается 100% гарантия корректности программ, в 3 случае такой уверенности нет, но мы можем убедится в том, что программа работает так же или иначе, чем аналогичный вариант. Менее достоверные тесты приходится использовать из-за недостаточности сил и средств.

1. **Классификация анализа вредоносных программ.**

**Можно выделить следующие способы сбора данных для выявления вредоносных программ:**

* Работа с файлами как с массивом байтов
* Эмуляция кода программы
* Запуск программы в «песочницу» (sandbox 2) (а также использование близких по смыслу технологий виртуализации)
* Мониторинг систем событий
* Поиск системных аналогий

**Считывание файлов** – в процессе поиска вредоносных программ данных, передаваемые в комплект принятия решений, извлекаются из файлов и предоставляют собой определенным образом упорядоченные массивы бит. Характерная особенность этого способа в том, что работа производится только с исходным байтовым кодом программы, не затрагивая ее поведение.

**Эмуляция** – это моделирование работы какого-либо программного устройства на другом компьютере, возможно отличающиеся по архитектуре, с применением исходного кода выполняемой программы.

**«Песочница»** – это система для выявления вредоносных программ, при использовании которой подозрительный объект запускается в виртуальной машине с полнофункционально ОС, а для обнаружения зловредности объекта применяется анализ его поведения. Если объект выполняет вредоносные действия, песочница признает его вредоносной программой.

**Мониторинг систем событий** – это процесс сбора, анализа м передачи сигналов о событиях абонентам, таким как процессы ОС, активные правила БД, а также операторы – люди

Поиск системных аномалий – это инструмент, помогающий выявлять нарушения (умышленные и неумышленные) информационной безопасности, неправильную работу ПО, ошибки в конфигурировании ОС.

**Считывание файлов** – код формируется в упорядоченные массивы байт и анализируется.

**Эмуляция** – моделирование работы программного устройства на другом компьютере без исходного программного кода программы.

**«Песочница»** - запуск объекта в виртуальной среде и анализ его поведения.

**Мониторинг системных событий** – сбор, анализ, передача сигналов о событиях абонентам (процессам операционной системы, операторы-люди, активные правила БД).

**Поиск системных аномалий** – инструмент, помогающий выявлять нарушения, ошибки в конфигурации системы, неправильную работу ПО.

1. **Этапы изучения вредоносного программного обеспечения и их описание.**

**Этапы изучения вредоносного ПО:**

* Обратное проектирование;
* Интерактивный анализ поведения;
* Анализ статических свойств;
* Полностью автоматизированный анализ.

**Обратное проектирование** (Reverse Engineering) – исследования устройства и документации с целью понять принцип его работы.

**Интерактивный анализ поведения** – взаимодействие с программами с целью их изучения.

**Анализ статических свойств** – анализ данных, которые можно собрать без необходимости запуска программы (детали заголовка, хеши, встроенные ресурсы, подписи упаковщика, метаданные).

**Полностью автоматизированный анализ** – основные функции и шаги теста (запуск, инициализация, выполнение, анализ, выдача результата) проводятся полностью автоматически.

**Обратное проектирование** – исследование некоторого готового устройства или программы, а также документации на него с целью понять принцип его работы, например, чтобы обнаружить недокументированные возможности (в том числе … закладки), сделать изменение или воспроизвести устройство, программу или иной объект с аналогичными функциями, но без прямого…

**Интерактивный анализ поведения** – инструменты поведения участвуют в процедуре анализа вредоносных программ когда аналитики использовали автоматизированные инструменты и изучили статические свойства, чтобы лучше понять образец, тогда аналитик решает взаимодействовать с вредоносными программами вместо того, чтобы пассивно наблюдать за ними.

**Анализ статических свойств** – инструменты статических свойств предоставляют метрики статических свойств для аналитиков, которые  
помогают им более точно анализировать подозрительные файлы нем необходимости запускать вредоносные программы для получения статических данных, поэтому получить эти данные несложно. Детали заголовка, хеши встроенные ресурсы, подписи упаковщика, метаданные являются экземплярами статических свойств.

**Полностью автоматизированный** функции и шаги теста **анализ** – процесс тестирования ПО, при котором основные, такие как запуск, инициализация, выполнение, анализ и выдача результата, производятся автоматически с помощью инструментов для автоматизированного тестирования.

1. **Основные инструменты для исследования кода программ: виды, свойства, значение.**

**Дизассемблирование** – получение исполняемого кода программы на языке ассемблера.

**Дизассемблер** – программа, осуществляющая дизассемблирование.

**Интерактивный дизассемблер** – программа, тесно взаимодействующая с пользователем в процессе дизассемблирования.

**Отладчик** – программа, предназначенная для анализа поведения другой программы, обеспечивающая остановку в указанных точках и позволяющая просматривать (редактировать) содержимое ячеек памяти, регистров процессора и команды программы.

**Эмулирующий отладчик** – самостоятельно интерпретирует и выполняет команды программы (без использования реального процессора).

**Программы-утилиты** (предназначены для вспомогательных операций по изучению логики работы механизма защиты):

1. **Файловый монитор (FileMonitor): возможности, предназначение.**

FileMonitor может отследить работу защищённой программы с файлами и обнаружить ключ (пароль), хранящийся в некотором файле.

1. **Методы, применяемые для обнаружения модифицированного кода. Преимущества и недостатки их применения.**
2. **Понятие отладчика. Назначение, виды.**

**Отладчик** – программа, предназначенная для анализа поведения другой программы, обеспечивающая остановку в указанных точках и позволяющая просматривать (редактировать) содержимое ячеек памяти, регистров процессора и команды программы.

1. **Отладочный механизм: трассировка кода программы. Назначение, принцип работы.**

**Трассировка** – выполнение программы на ЭВМ с использованием специальных средств, позволяющих выполнять программу в пошаговом режиме, получать доступ к регистрам, областям памяти, производить остановку программы по определённым адресам и т.д.

Для защиты программ от изучения необходимо иметь средства противодействия как дизассемблированию, так и трассировке.

1. **Отладочный механизм: контрольные точки останова. Назначение, принцип работы.**

**Контрольная точка** – любое место в программе, на котором обычное выполнение программы приостанавливается и осуществляется переход в особый режим. Например, в режим покомандного выполнения.

1. **Понятие тестирования программ. Назначение, методы.**

**Тестирование программного обеспечения (Software Testing)** — проверка соответствия реальных и ожидаемых результатов поведения программы, проводимая на конечном наборе тестов, выбранном определённым образом.

**Методы**:

* Метод черного ящика
* Метод белого (стеклянного ящика)
* Метод серого ящика

1. **Понятие** **инспекции программного кода. Отличительные особенности, цели данного метода тестирования.**

Просмотр кода (англ. code review) или **инспекция кода** (англ. code inspection) – систематическая проверка исходного кода программы с целью обнаружения и исправления ошибок, которые остались незамеченными в начальной фазе разработки. **Целью** просмотра является улучшение качества программного продукта и совершенствование навыков разработчика.

1. **Понятие и основные отличия тестирования моделей.**

**Тестирование моделей** – анализируется и тестируется модель системы, а не она сама. На модели можно создать ситуации, которые невозможно создать при лабораторных тестах. На модели тестируются оптимальность алгоритмов, устойчивость системы.

Но т.к. разработать формальное описание для теста моделей сложно, такое тестирование используется редко (например, в системе связи).

1. **Характеристика и описание процесса тестирования методом «черный ящик».**

**Чёрный ящик (Black Box)** – тестировщик не знает устройство программы. Он может только воздействовать на входы программы, и получать на выходе некоторый результат.

Чёрный ящик может представлять собой набор классов или модулей с известными внешними интерфейсами, но недоступным исходным кодом.

**Выявляемые проблемы:**

* Несоответствие поведения системы требованиям (Обычно вызывает изменение кодов);
* Неадекватное поведение системы в ситуациях, не предусмотренных требованиям (явно вызывают изменения требований).

1. **Характеристика и описание процесса тестирования методом «стеклянный ящик».**

**Стеклянный (белый) ящик** – тестировщик видит код. Может определить соответствие требованиям участков программного кода, видеть участки кода, для которых не существуют требования (непокрытый требованиями код). Более углубленный анализ.

1. **Определение и основные особенности для термина: «эмулирующий отладчик».**

**Эмулирующий отладчик** – самостоятельно интерпретирует и выполняет команды программы (без использования реального процессора).

1. **Методы проникновения вредоносного программного обеспечения на персональный компьютер.**

**Флеш-накопители (флешки)**

В настоящее время USB-флешки заменяют дискеты и повторяют их судьбу — большое количество вирусов распространяется через съёмные накопители, включая цифровые фотоаппараты, цифровые видеокамеры, цифровые плееры (MP3-плееры), сотовые телефоны. Использование этого канала ранее было преимущественно обусловлено возможностью создания на накопителе специального файла autorun.inf, в котором можно указать программу, запускаемую Проводником Windows при открытии такого накопителя. В последней версии MS Windows под торговым названием Windows 7 возможность автозапуска файлов с переносных носителей была устранена. Флешки — основной источник заражения для компьютеров, не подключённых к Интернету.

**Электронная почта**

Сейчас один из основных каналов распространения вирусов. Обычно вирусы в письмах электронной почты маскируются под безобидные вложения: картинки, документы, музыку, ссылки на сайты. В некоторых письмах могут содержаться действительно только ссылки, то есть в самих письмах может и не быть вредоносного кода, но если открыть такую ссылку, то можно попасть на специально созданный веб-сайт, содержащий вирусный код. Многие почтовые вирусы, попав на компьютер пользователя, затем используют адресную книгу из установленных почтовых клиентов типа Outlook для рассылки самого себя дальше.

**Системы обмена мгновенными сообщениями**

Также распространена рассылка ссылок на якобы фото, музыку либо программы, в действительности являющиеся вирусами, по ICQ и через другие программы мгновенного обмена сообщениями.

**Веб-страницы**

Возможно также заражение через страницы Интернета ввиду наличия на страницах всемирной паутины различного «активного» содержимого: скриптов, ActiveX-компонент. В этом случае используются уязвимости программного обеспечения, установленного на компьютере пользователя, либо уязвимости в ПО владельца сайта (что опаснее, так как заражению подвергаются добропорядочные сайты с большим потоком посетителей), а ничего не подозревающие пользователи, зайдя на такой сайт, рискуют заразить свой компьютер.

**Интернет и локальные сети (черви)**

**Черви** — вид вирусов, которые проникают на компьютер-жертву без участия пользователя. Черви используют так называемые «дыры» (уязвимости) в программном обеспечении операционных систем, чтобы проникнуть в компьютер. Уязвимости — это ошибки и недоработки в программном обеспечении, которые позволяют удаленно загрузить и выполнить машинный код, в результате чего вирус-червь попадает в операционную систему и, как правило, начинает действия по заражению других компьютеров через локальную сеть или Интернет.

Вирусы действуют только программным путем. Они, как правило, присоединяются к файлу или проникают в тело файла. В этом случае говорят, что файл заражен вирусом. Вирус попадает в компьютер только вместе с зараженным файлом. Для активизации вируса нужно загрузить зараженный файл, и только после этого, вирус начинает действовать самостоятельно.

1. **Понятие вредоносного программного обеспечения. Отличительные свойства, виды вредоносного программного обеспечения, Функциональные возможности каждого вида.**

**Вредоносная** **программа** (другие **термины**: зловредная **программа**, вредонос, зловред; англ. malware — словослияние слов malicious и software (рус. малварь)) — любое **программное** **обеспечение**, предназначенное для получения несанкционированного доступа к вычислительным ресурсам самой ЭВМ или к информации, хранимой на ЭВМ, с целью несанкционированного использования ресурсов ЭВМ или причинения вреда (нанесения ущерба) владельцу информации, и/или владельцу ЭВМ.

Выделяют несколько типов вредоносного ПО: **шпионские, рекламные, фишинговые, троянские, вымогательские программы, вирусы, черви, руткиты и программы, нацеленные на захват контроля над браузером**.

**Рекламные закладки(Adware)**

Такая программа загружает рекламу автоматически. Adware, без вашего ведома отправляют вашу личную информацию рекламодателям или же встраивают рекламные блоки в другие программы, например, в браузеры. Обычно они входят в комплект бесплатного программного обеспечения, что позволяет производителю получить определенный доход на рекламе. Adware замедляют загрузку веб-страниц, снижают производительность системы, модифицируют программы посредством вставки специальных скриптов.

**Бот**

Боты - это программы, созданные для автоматического выполнения определенных операций.  Хотя некоторые боты создаются для относительно безвредных целей (видеоигры, интернет-аукционы, онлайн-конкурсы и т. д.), все чаще встречаются боты, которые используются злонамеренно.

Боты могут использоваться в бот-сетях (собраниях компьютеров, контролируемых третьими лицами) для DdoS-атак, в качестве спам-ботов, которые размещают рекламу на веб-сайтах, в качестве веб-пауков, которые очищают данные сервера, и для распространения вредоносных программ, замаскированных под популярные поисковые элементы на сайтах загрузки.

**Баги**

Баг – это ошибка в программе, которая приводит к ее неправильной работе. Баги часто являются результатом человеческой ошибки и обычно существуют в исходном коде или компиляторах программы. Серьезные баги могут привести к сбою или зависанию. Баги системы безопасности являются наиболее серьезным типом ошибок и могут позволить злоумышленникам обойти аутентификацию пользователя, переопределить права доступа или украсть данные.

**Вирус-вымогатель (Ransomware)**

Ransomware - это форма вредоносного ПО, которая, по сути, удерживает компьютерную систему в плену, требуя выкуп. Вредоносное ПО ограничивает доступ пользователей к компьютеру путем шифрования файлов на жестком диске или блокировки системы. Также программа отображает сообщения, которые должны заставить пользователя заплатить создателю вредоносного ПО за снятие ограничений и восстановление доступа к своему компьютеру. Ransomware обычно распространяется, как обычный компьютерный червь и попадает на компьютер через загруженный файл или же через какую-то другую уязвимость в сетевой службе.

**Руткит**

Руткит - это разновидность вредоносного программного обеспечения, которое получает  удаленный доступ к компьютеру или управляет им без возможности обнаружения пользователями или программами безопасности. После установки руткита злоумышленник, находящийся за руткитом, может удаленно управлять файлами, красть информацию, изменять конфигурацию системы, изменять программное обеспечение, устанавливать скрытое вредоносное ПО или контролировать компьютер как часть ботнета. Поскольку руткит постоянно скрывает свое присутствие, типичные продукты безопасности не эффективны для обнаружения и удаления руткитов. Руткит можно обнаружить с помощью мониторинга поведения компьютера, сканирования подписей и анализа дампы памяти.

**Шпионские программы (spyware)**

Шпионское ПО - это тип вредоносного программного обеспечения, которое следит за действиями пользователя без их ведома. Эти программы могут выполнять мониторинг активности, сбор нажатий клавиш, сбор данных (учетная запись, логины, финансовые данные) и многое другое. Шпионские программы обычно распространяются в купе с легальным программным обеспечением или троянскими программами.

**Троянская программа**

Троянская программа, также известная как «троянский конь» -  это тип вредоносного ПО, которое маскируется под обычный файл или программу, чтобы обманным путем заставить пользователей загружать и устанавливать вредоносное ПО. Троян может предоставить злоумышленнику удаленный доступ к зараженному компьютеру. После того, как злоумышленник получит доступ к зараженному компьютеру, он сможет украсть данные (логины, финансовые данные, даже электронные деньги), установить дополнительные  вредоносные программы, изменить файлы, отслеживать действия пользователя (просмотр экрана, ведение журнала и т. д.).

**Вирус**

Вирус - это форма вредоносного ПО, которое способно копировать и распространять себя на другие компьютеры. Вирусы часто распространяются на другие компьютеры, подключаясь к различным зараженным программам, в то время как пользователь запускает их. Вирусы могут быть использованы для кражи информации, нанесения вреда компьютерам и сетям, создания бот-сетей, кражи денег, рекламы и многого другого.

**Сетевой червь**

Компьютерные черви являются одними из самых распространенных типов вредоносных программ.  
Они распространяются по локальной сети и Интернету, создавая свои копии. Черви обычно попадают в систему через флеш-накопитель или интернет, создают в большом количестве свои копии, находят в ПК все электронные адреса и рассылают по ним дубли. Внедрившись в устройство, пакетные черви стремятся проникнуть в его оперативную память с целью сбора персональных данных, ценной информации. В отличие от файловых вирусов сетевые черви способны использовать для распространения сетевые протоколы и сетевые устройства.

1. **Понятие метрики измерений. Виды метрик, функциональные особенности.**

**Метрики используются для измерения характеристик и критериев качества.**

**Метрика качества программ** – это система измерений качества ПО. Эти измерения проводятся на уровне критериев качества программ или на уровне отдельных характеристик качества.

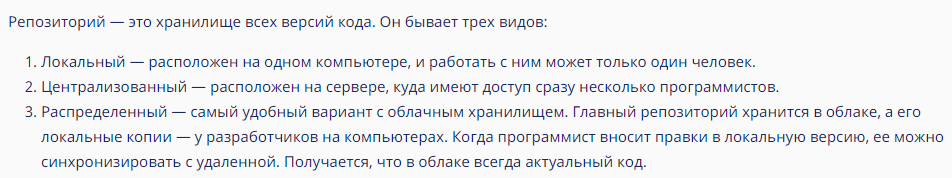
* В первом случае, система измерений позволяет сравнивать программы по качеству, при этом сами измерения не могут быть проведены без субъективных оценок свойств программы.
* Во втором случае, измерения характеристик можно выполнить объективно и достоверно, но оценка будет связана с субъективной интерпретацией оценок.

**Основные направления применения метрик:**

* Оценки топологической и информационной сложности программ;
* Оценки надёжности Программных Систем для прогнозирования отказовых ситуаций;
* Оценки производительности ПО и повышение эффективности путём выявления ошибок проектирования;
* Оценки уровня языковых средств;
* Оценки трудности восприятия и понимания программных текстов;
* Оценки производительности труда программистов.

**Виды:**

1. **Хранилище (Repository): основные понятия.**

****

1. **Рабочая копия (Working Copy): основные понятия.**

Работа с рабочей копией файла:

При работе с рабочей копией часто необходимо понять, какие файлы разработчик заменил, добавил, удалил, переименовал, а также изменения файлов другими разработчиками. Теперь после извлечение рабочей копии из хранилища в Subversion можно заметить, что значки в проводнике Windows немного изменились, и это одна из причин популярности TortoiseSVN. Она добавляет «пометку» на значке для каждого файла, которая накладывается на исходный значок файла.

1. **Тестирование ПО общая характеристика, понятия, виды.**

**Тестирование ПО** – процесс исследования, испытания программного продукта, имеющий своей целью проверку соответствия между реальным поведением и её ожидаемым поведением на конкретном наборе тестов, выбранным определённым образом (ISO/IECTR 19757 2005).

Тестирование представляет собой деятельность по проверке программного кода и документации. Она должна заранее планироваться и систематически проводиться специально назначенным независимым тестировщиком. Работа тестировщика начинается до утверждения спецификаций требований. Он проверяет требования к ПП на полноту и возможность тестирования, определяет методы тестирования.

**Виды:**

* **Модульное тестирование** – процесс проверки отдельных программных процедур и подпрограмм, входящих в состав программ или программных систем.
* **Интеграционное тестирование** – проводится для проверки совместной работы отдельных модулей и предшествует тестированию всей системы как единого целого.
* **Системное тестирование** – предназначен для проверки программной системы в целом её организации и функционирования на соответствие спецификациям требований заказчика. Его проводит независимый тестировщик после успешного завершения интеграционного тестирования.
* **Выходное тестирование** –завершающий этап тестирования, на котором проверяется готовность ПП к поставке заказчику. Данный вид тестирования проводит независимый тестировщик.
* **Приёмочное тестирование** – проводится организацией, отвечающей за инсталляцию сопровождение программной системы и обучение конечного пользователя.

1. **Понятия управления контроля версиями, применение и функции системы контроля версий.**

Под **системой контроля версий** понимается механизм сохранения промежуточных состояний кода, разрабатываемого программного обеспечения. То есть с помощью этой системы программист может управлять своими файлами во времени: смотреть историю изменения файлов и каталогов, возвращаться к более ранним версиям кода, объединять несколько версий файлов.

Основной **областью применения контроля версий** является коллективная разработка (чаще всего программы, область применения программирования которых не ограничивается). Однако, и для разработчика одиночки контроль версий может быть полезен.

**СКВ/СУВ** – это ПО, предназначенное для работы с постоянно изменяющейся информацией.

**СУВ предоставляет следующие возможности:**

* Поддержка хранения файлов в репозитории;
* Поддержка истории версий файлов в репозитории;
* Нахождение конфликтов при изменении исходного кода и обеспечение синхронизации при работе в многопользовательской среде разработки
* Отслеживание авторов изменений

1. **Классификация систем контроля версий.**

* Централизованные (распределённые) – в централизованных СКВ вся работа производится с центральным репозиторием; в распределённых – у каждого разработчика есть локальная копия репозитория.
* Блокирующие (неблокирующие) – блокирующие СКВ позволяют наложить запрет на изменения файла, пока один из разработчиков работает над ним; в неблокирующих один файл может одновременно изменяться несколькими разработчиками.
* Для текстовых данных (бинарных). Для СКВ с текстовыми данными важна поддержка слияния изменений. Для СКВ с бинарными данными важна возможность блокировки.

1. **Функциональное значение контроля версий типа SVN. Современные подходы к анализу ПО.**

SVN – одна из наиболее известных систем версионного управления исходным кодом. Применяется во многих open source проектах. Подобные системы позволяют узнать кто, когда и как именно вносил изменения в файлы проектов. Всё это значительно облегчает групповую разработку ПО.

1. **Уровни тестирования: описание. Цикл тестирования: иерархическое древо: графическое отображение.**
2. **Импорт и экспорт настроек: общие сведения, применение.**

**ЛР№2**

**Экспорт:**

В разделе “средства” вы должны сделать свой выбор

вручную, если действительно хотите, чтобы эта информация была

сохранена в резервном файле. Выбрав настройки, которые хотите

экспортировать, переходите к следующему этапу работы мастера,

который может занять несколько минут, в зависимости от того,

сколько настроек вы экспортируете

**Импорт:**

Для этого используется тот же самый мастер, но теперь на первом этапе вам следует выбрать команду Import. Вместо простой перезаписи текущей конфигурации мастер позволяет вам сначала сохранить резервную копию текущих настроек.

Затем можете выбрать существующий файл конфигурации из

списка.

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

ПР№6

**Обратным проектированием (Reverse engineering) называется** процесс преобразования в модель кода, записанного на каком-либо языке программирования.

Процесс обратного проектирования делится на два этапа: анализ и генерацию модели. На первом этапе производятся все подготовительные операции по анализу текста программы на отсутствие синтаксических ошибок. Второй этап – преобразование кода в модель.

**Для чего выполняется:** для превращения бинарного кода в читаемый текст программы. Для того, чтобы понять принцип работы; например, чтобы обнаружить недокументированные возможности (в том числе программные закладки), сделать изменения или воспроизвести программу с аналогичными функциями.

**Результатом обратного проектирования** базы данных или сценария на языке определения данных (Data Definition Language - DDL) обычно является набор элементов модели (таблиц, представлений, хранимых процедур и т.д.). В зависимости от сложности базы данных может потребоваться разделить полученные элементы модели на пакеты, содержащие логически связанные наборы таблиц.

**Инструменты:** существуют, например, IDA и GHIDRA.

1. **Применение и функции обратного проектирования.**

**Для чего выполняется:** для превращения бинарного кода в читаемый текст программы. Для того, чтобы понять принцип работы; например, чтобы обнаружить недокументированные возможности (в том числе программные закладки), сделать изменения или воспроизвести программу с аналогичными функциями.

1. **Описание анализа обратного проектирования.**

**В результате** этого процесса вы получаете огромный объем информации, часть которой находится на более низком уровне детализации, чем необходимо для построения полезных моделей. В то же время обратное проектирование никогда не бывает полным. Как уже упоминалось, прямое проектирование ведет к потере информации, так что полностью восстановить модель на основе кода не удастся, если только инструментальные средства не включали в комментариях к исходному тексту информацию, выходящую за пределы семантики языка реализации.

**Процесс обратного проектирования делится на два этапа**: анализ и генерацию модели. **На первом этапе** производятся все подготовительные операции по анализу текста программы на отсутствие синтаксических ошибок. **Второй этап** – преобразование кода в модель.

1. **Модульное тестирование: основные понятия, элементы, результат.**

**Модульное тестирование** – процесс проверки отдельных программных процедур и подпрограмм, входящих в состав программ или программных систем.

**Элементы модульного тестирования:**

* Синтаксическая проверка – проверка с использованием некоторого инструментального средства для выявления синтаксических ошибок в программном коде;
* Проверка соответствия стандартам кодирования – проверка кода на соответствие стандартам кодирования компании;
* Технический обзор программного кода.

После успешного завершения модульного тестирования все изменённые модули и наборы тестов сохраняются в БД проекта.

1. **Интеграционное тестирование: основные понятия, элементы, результат.**

**Интеграционное тестирование** – проводится для проверки совместной работы отдельных модулей и предшествует тестированию всей системы как единого целого.

**Элементы интегрированного тестирования:**

* Проверка функциональности – проверка соответствия отдельных функций, выполняемых совокупностями модулей, функциям, заданных в спецификациях требований;
* Проверка промежуточных результатов – проверка всех промежуточных результатов и файлов на наличие и корректность;
* Проверка интеграции – проверка того, что модули передают друг другу информацию корректно.

Ошибки, выявленные в ходе интеграционного тестирования, заносится в БД ошибок.

Результаты интеграционного тестирования включаются в отчёт о ходе тестирования при завершении цикла тестирования.

1. **Системное тестирование: основные понятия, элементы, результат.**

**Системное тестирование** – предназначен для проверки программной системы в целом её организации и функционирования на соответствие спецификациям требований заказчика. Его проводит независимый тестировщик после успешного завершения интеграционного тестирования.

**Элементы системного тестирования:**

* Граничное тестирование – тестирование в граничных условиях;
* Прогоночное тестирование – тестирование всех функциональных характеристик реальной работы системы;
* Целевое тестирование – тестирование на целевой платформе (по возможности);
* Проверка документации – проверка пользовательской документации на корректность;
* Другие тесты, определяемые тестировщиком.

Ошибки, выявленные при системной тестировании, заносятся в БД проекта.

Результаты системного тестирования включаются в отчёт о ходе тестирования.

1. **Локальный репозиторий: определение и функции.**

Локальный – расположен на одном компьютере, и работать с ним может только один человек.

Локальный репозиторий может храниться в локальной сети или на одном устройстве

1. **Понятие тестирования методом «серый ящик» и описание процесса.**

**Тестирование серого ящика** - это комбинация тестирования белого ящика и тестирования черного ящика. **Целью**этого тестирования является поиск дефектов, если таковые имеются, из-за неправильной структуры или неправильного использования приложений.

Это комбинация тестирования белого ящика и тестирования черного ящика. Специальный метод тестирования программного обеспечения с неполным знанием его внутреннего устройства. Чтобы выполнить подобный вид тестов, не нужно иметь доступ к исходному коду ПО. Все тесты создаются на базе простого знания алгоритмов, архитектуры и иных высокоуровневых характеристик поведения продукта.

1. **Получение доступа к GitHub. Основные функции и принцип GitHub.**

**ЛР№7**

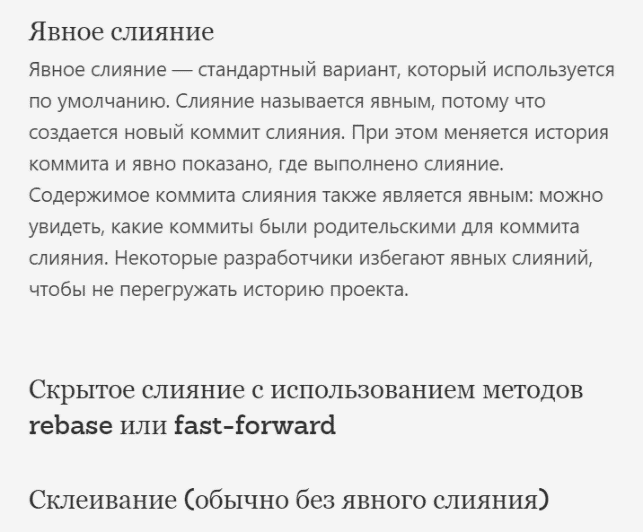
* Можно создавать приватные репозитории, которые будут видны только вам и выбранным вами людям. Раньше такая возможность была платной.
* Есть возможность прямого добавления новых файлов в свой репозиторий через веб-интерфейс сервиса.
* Код проектов можно не только скопировать через Git, но и скачать в виде обычных архивов с сайта.
* Кроме Git, сервис поддерживает получение и редактирование кода через SVNи Mercurial
* На сайте есть pastebin-сервис gist.github.com для быстрой публикации фрагментов кода.
* Файлы из репозитория могут автоматически публиковаться в виде статического сайта с помощью GitHub Pages

**Получение доступа:** нужно зарегистрироваться на сайте и скачать приложение GitHub

1. **Организация разработки в сервисе Github.**
2. **Описание уровней доступа GitHub.**

* Pull Only: выборка и слияние с другим репозиторием или локальной копией. Доступ только для чтения.
* Push and Pull: – доступно обновлением удаленного репозитория. Читайте + Запись.
* Push, Pull & Administrative добавляются права созданием команд, а также удаление аккаунтов организации. Чтение + запись + доступ администратора

1. **Характеристика способов слияния копий репозитория.**

****

1. **Модели создания кода в репозитории: описание и графическое представление.**
2. **Архитектура MVC: основные понятия, структура.**

**MVC** (Model-View-Controller: модель-вид-контроллер) — шаблон архитектуры ПО, который подразумевает разделение программы на 3 слабосвязанных компонента, каждый из которых отвечает за свою сферу деятельности.

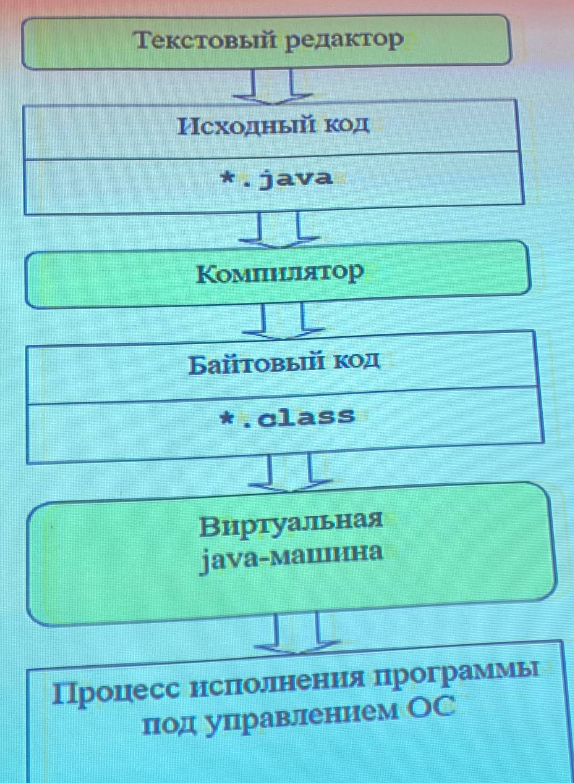
1. **Описание процесса реализации проверки на стороне клиента.**

С другой стороны, проверка пользовательского ввода, выполняемая на стороне клиента, называется проверкой на стороне клиента. Языки сценариев, такие как JavaScript и VBScript, используются для **проверки на стороне клиента** . В этом виде проверки вся проверка ввода пользователя выполняется только в браузере пользователя. Это не так безопасно, как проверка на стороне сервера, потому что хакер может легко обойти наш язык сценариев на стороне клиента и отправить опасный ввод на сервер.

1. **JavaScript: инструменты языка. Состав среды разработки (IDE).**

**Состав IDE:**

* специализированный **текстовый редактор**, который облегчает форматирование текста программы, подсвечивает синтаксис и предоставляет прочие удобства;
* **отладчик** - программный комплекс для поиска и исправления ошибок в программе;
* **фоновый компилятор**, который указывает на синтаксические ошибки еще в процессе набора текста программы;
* **справочная система**.



* Для исполнения программы на компьютере должна быть установлена JRE, java-машина интерпретирует байтовый код, учитывая особенности данной ОС (исполняемый файл НЕ создается).
* НО: Исполняемый файл (нативный бинарный код для конкретной операционной системы) создать можно при помощи сторонних программ-утилит.

1. **Описание процесса реализации проверки на стороне клиента. Преимущества проверки на сервере.**

Проверка на стороне клиента очень удобна для пользователей. Она экономит время, которое в противном случае тратилось бы на круговой путь к серверу, в результате которого выдавались бы ошибки проверки. С точки зрения бизнеса даже доли секунды, умножаемые в сотни раз каждый день, позволяют значительно сократить расходуемое время, деньги и усилия. Простая и немедленная проверка позволяет пользователям работать эффективнее и повышает точность входных и выходных данных. Даже при проведении проверки на стороне клиента следует всегда проверять команды или входящие объекты переноса данных в серверном коде, ведь API сервера являются возможным вектором атаки.

1. **Библиотека классов: понятие, доступ к членам класса.**
2. **Язык Ассемблера: возможности языка, преимущества и недостатки.**
3. **Сегментная адресация: определение, примеры применения.**
4. **Дизассемблер: возможности программы, преимущества и недостатки.**
5. **Мониторинг работоспособности системы контроля версий: назначение и инструменты мониторинга.**

**Bazaar** (ранее Bazaar-NG) – утилита командной строки bzr, распределённая система управления версиями, разработка которой спонсируется фирмой Canonical Ltd, в последнюю версию по сравнения с предыдущей было внесено более 50 изменений. Данная система разработана в целях облегчения создания и развития проектов для пользователей.

Bazaar – это удобная система контроля версий с приятным интерфейсом, она хорошо подходит для пользователей, которых не привлекает перспектива работы с командной строкой. Имеется множество дополнительных опций и расширений, что позволяет настроить программу под свои нужды.

**Mercurial** (англ. «Подвижный») – распределённая система управления версиями, способная функционировать на многих операционных системах и различных аппаратных платформах, разработанная для эффективной работы с очень большими кодами.

Говоря о Mercurial, следует отметить, что простой и отточенный интерфейс, набор команд, возможность импорта репозиториев с других систем контроля версий сделают переход на данную программу безболезненным и быстрым, а её надёжность и скорость работы позволяют пользоваться им для контроля версий огромных проектов. Всё это позволяет Mercurial стать достойным конкурентом git’а.

**Git** – распределённая система управления версиями файлов. Код программы был написан на языке “C”, проект создан Линусом Торвальдсом в 2005 году для управления разработкой ядра Linux, является общедоступным программным обеспечением. Данная система была введена многими ведущими разработчиками, используется в известных Linux-сообществу проектах.

В свою очередь Git – это гибкая, удобная и мощная система контроля версий, способная удовлетворять абсолютное большинство пользователей. Git – один из лидеров систем контроля версий.

**Concurrent Versions System** (или CVS, англ. “Система Одновременных Версий”) – представляет собой программный продукт, который относится к разряду систем управления версиями. Программа хранит историю изменений исходного кода программного обеспечения, тем самым облегчая совместную работу программистов над одним проектом. CVS популярна в мире открытого программного обеспечения.

Несмотря на то, что CSV достаточно устарела и обладает весомыми недостатками, она всё ещё является одной из самых популярных систем контроля версий и отлично подходит для управления малыми проектами, не требующих создания нескольких параллельных версий, которые надо периодически соединять.

1. **Коллективная разработка: характеристика, структура.**

**Коллективная разработка –** одним из основных вопросов кр является разделение труда, от равноправных соисполнителей до организации в виде жесткой иерархии (например: бригады главного программиста)

Бригада равноправных соисполнителей обычно состоит из специалистов, занимающиеся примерно подобными задачами в рамках одного проекта. Специализации в одной бригаде может быть несколько:

* Инженеры разработчики
* Технические писатели
* Инженеры тестирования
* Инженеры качества
* Специалисты по сопровождению продукта
* Специалисты по продажам продукта

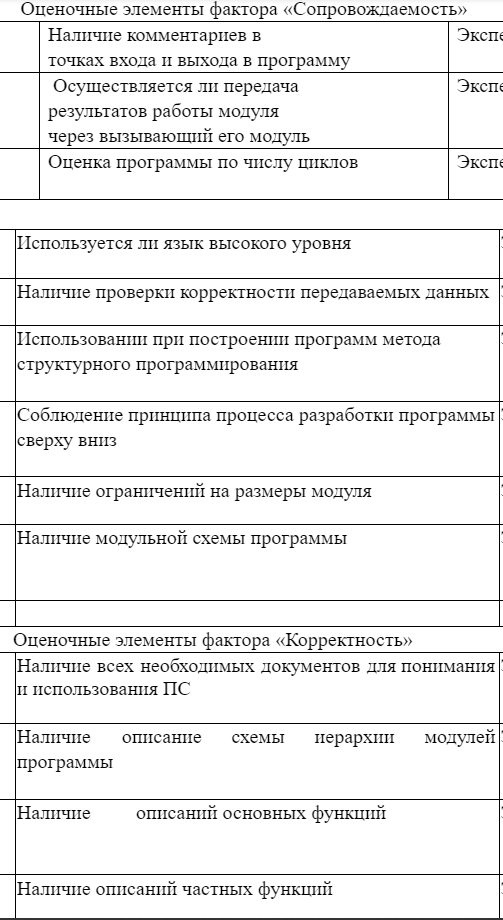
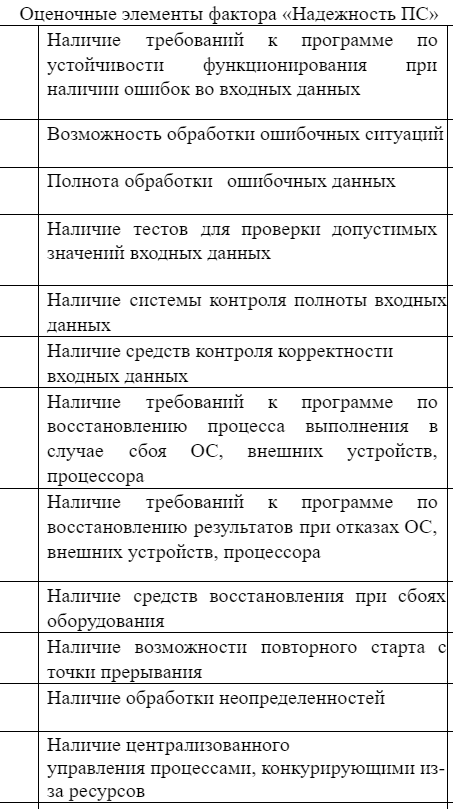
**Основные этапы разработки ПО:**

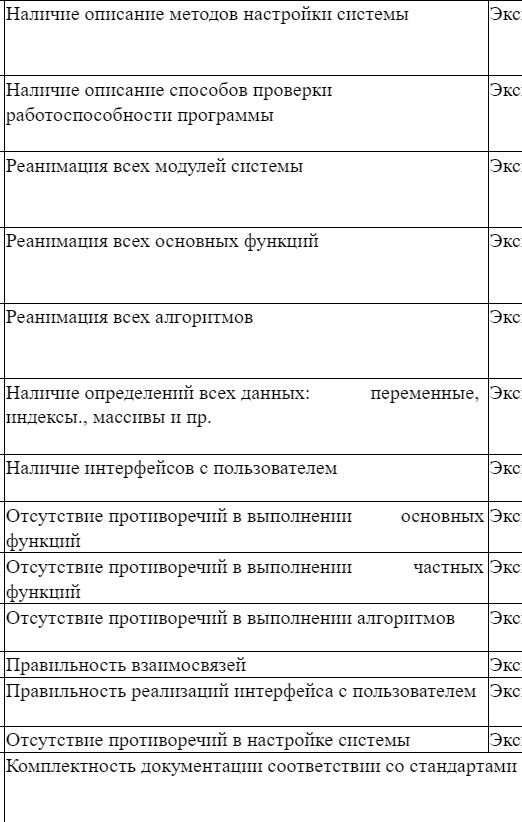
1. Анализ (определение процесса разработки ПО)
2. Проектирование (управление проектом разработки)
3. Конструирование (описание целевого ПО программного продукта)
4. Программирование (проектирование продукта)
5. Разработка продукта
6. Тестирование (тестирование частей п.п.)
7. Отладка (интегрирование частей п.п. и тестирования продукта в целом)
8. Развертывание п.п. (распространение)
9. Выпуск продукта

**Минимальные функции системы коллективной разработки:**

* Регистрация изменений вносимых в проект
* Хранение файлов проекта

***Практические задания:***





1. **Сравнить функциональные особенности калькулятора, предуставленного в ОС Windows и предустановленного в папке «экзамен» на рабочем столе под названием «1.exe» и описать отличительные черты каждого из них. Результат оформить в документе Word в виде таблицы.**

Описать по скринам сверху возможности первой и второй программы, сравнивая их по 1 - если подходит, 0 - если не подходит.

1. **На основании основных метрик описать функциональные особенности представленной в папке «экзамен» на рабочем столе программы под названием «2.exe». Определить основную цель приложения. Результат оформить в документе Word в виде таблицы.**

Описать по скринам сверху особенности программы ставля 1 - если подходит, 0 - не подходит. Целью подразумевается, для чего программа нужна и что она делает.

1. **На основании основных метрик описать функциональные особенности представленной в папке «экзамен» на рабочем столе программы под названием «3.exe». Определить основную цель приложения. Результат оформить в документе Word в виде таблицы.**

то же самое, что и в задании 2.

1. **На основании основных метрик описать функциональные особенности представленной в папке «экзамен» на рабочем столе программы под названием «4.exe». Определить основную цель приложения. Результат оформить в документе Word в виде таблицы.**

то же самое, что и в задании 2.

1. **На основании основных метрик описать функциональные особенности представленной в папке «экзамен» на рабочем столе программы под названием «5.exe». Определить основную цель приложения. Результат оформить в документе Word в виде таблицы.**

то же самое, что и в задании 2.

1. **Сравнить функциональные особенности представленных в папке «экзамен» на рабочем столе программ под названием «2.exe» и «5.exe». Определить основную цель приложений, описать отличительные черты каждого из них. Результат оформить в документе Word в виде таблицы.**

то же самое, что и в задании 1.

1. **Сравнить функциональные особенности представленных в папке «экзамен» на рабочем столе программ под названием «4.exe» и «5.exe». Определить основную цель приложений, описать отличительные черты каждого из них. Результат оформить в документе Word в виде таблицы.**

то же самое, что и в задании 1.

1. **Сравнить функциональные особенности представленных в папке «экзамен» на рабочем столе программ под названием «2.exe» и «3.exe». Определить основную цель приложений, описать отличительные черты каждого из них. Результат оформить в документе Word в виде таблицы.**

то же самое, что и в задании 1.

1. **Сравнить функциональные особенности представленных в папке «экзамен» на рабочем столе программ под названием «4.exe» и «3.exe». Определить основную цель приложений, описать отличительные черты каждого из них. Результат оформить в документе Word в виде таблицы.**

то же самое, что и в задании 1.

1. **Ввести в действие пять ветвей разработок в двух репозиториях, переместить в подкаталог.**

Создать 2 папки, далее прописать git branch “название новой ветки”. git checkout fish - для выбора ветки. Далее cp “первый” “второй”. git commit -m “a”. git push

Git clone ссылка на репозиторий (в репозитории уже иметь файл(dog))

Cd cat(название репозитория)

Git branch fish создаем ветку

Git branch просмотр веток

Git checkout fish мы перешли в ветку

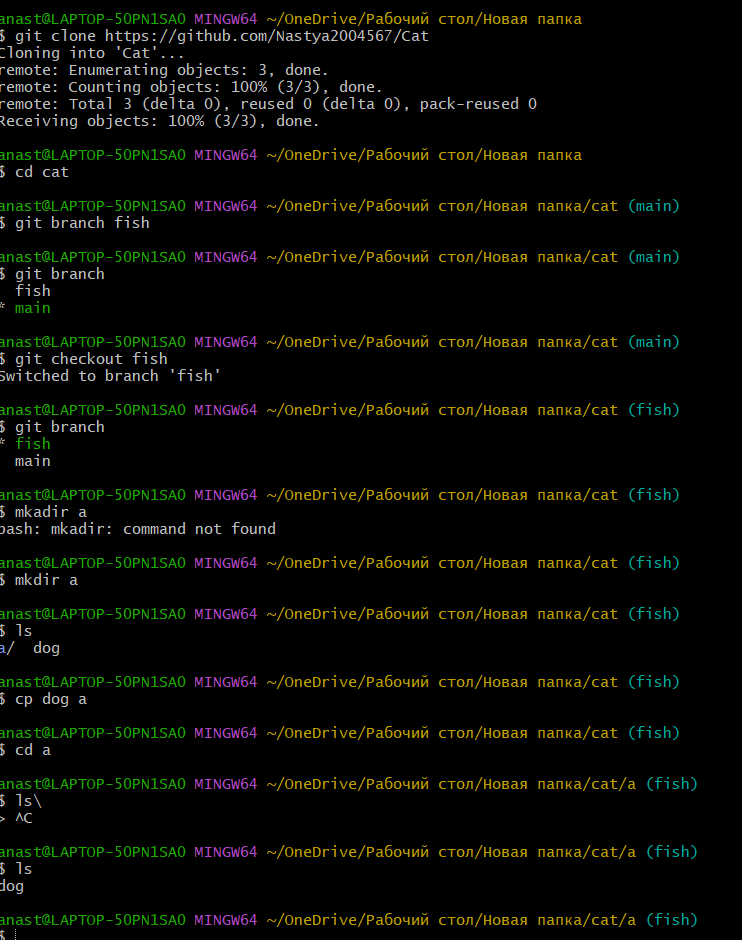
Mkdir a(название каталога) создаем подкаталог

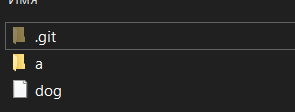
Ls смотрим что находится в cat

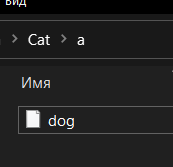
Cp dog(что) (куда)a

Cd a переходим

Ls смотрим







1. **Переместить Branch1 из исходного репозитория в соседний, добавить изменения и применить команду «Push», и просмотреть по системе контроля версий, дату обновления файла Branch1.**

Создать 2 папки, далее прописать git branch “название новой ветки”. git checkout fish - для выбора ветки. Далее cp “первый” “второй”. git commit -m “a”. git push.Далее в самом GitHub нажать на commits.

Создаем новую папку (Нова папка 2, там все находится)

Git clone ссылка на репозиторий (в репозитории уже иметь файл(dog) прошлый)

Там создаем b.txt прописываем что либо

Cd cat заходим в репозиторий

Git push конектим git

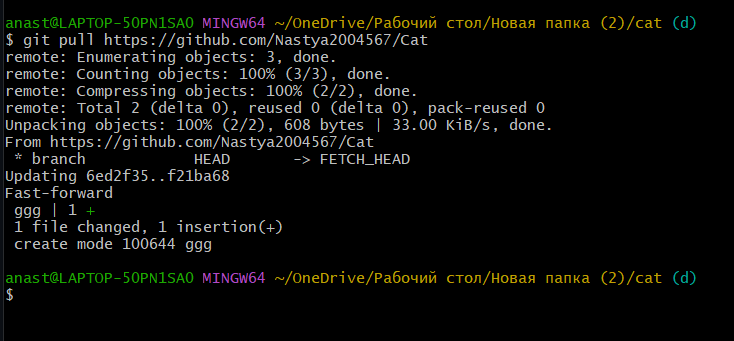
Создать ветку git branch d

Переходим git checkout d

Cкрины в дискорде

1. **Извлечь изменения из удалённого репозитория и применить их к текущей ветви с помощью команды «Git Pull».**

Создаем на git hube файл



Создаем второй репозиторий и в первом пишем git pull “ссылка на второй”

1. **Синхронизировать финальную версию предлагаемого проекта с репозиторием на GitHub.**

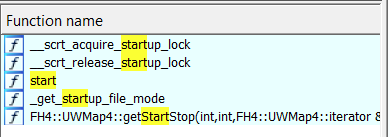
git pull ссылка на репозиторий. создаем какой-то файл. git add файл

git commit -m “a”

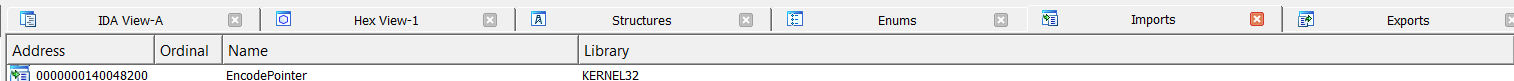
git push

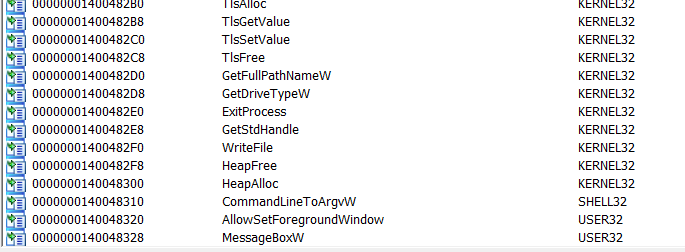
1. **Объединить ветви «final» и «master» из двух разных репозиториев, представленных удалённо.**
2. **Разработать в репозитории с помощью функции «menu» две ветви проекта.**
3. **В программе IDA pro 8.0 дизассемблировать файл «6.exe», находящийся в папке «экзамен» на рабочем столе. Отразить в виде таблицы в MS Word библиотеки, к которым он обращается, и указать, где начинается работа программы.**

Слева в окне будут функции, нужно найти функцию с названием start или main.



Для поиска библиотек нужно открыть вкладку Imports и в ней будут показаны библиотеки





Почти во всех проектах используется KERNEL32, можно ориентироваться по ней.

1. **Создать хранилища (репозитория) SVN и разместить в нем каталог. Выполнить импорт данных (Tortoise SVN).**
2. **Установить параметры работы среды Visual Studio, соответствующие Web-разработке.**

Создать программу в Visual Studio “Веб-приложение” ASP.NET (.NET Framework). Показать параметры.

1. **В Visual Studio импортировать настройки из файла в папке «Экзамен» на рабочем столе, в который выгружены настройки.**

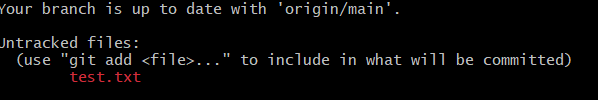
Для начала надо создать любой проект в Visual Studio. Далее заходим в Средства -> Импорт и экспорт параметров -> Импортировать выбранные параметры окружения. Далее нажимаем на обзор и выбираем нужный файл (он должен быть с расширением .vssettings.

1. **В программе IDA pro 8.0 дизассемблировать файл \*\*\*.exe. Отобразить главную функцию работы программы. Для функции Start выделить функции sub,call и push.**

Главная функция может называть main или start. Чтобы выделить нужно выделить саму строку с функцией потом нажать Edit -> Others -> Color instruction

1. **Создать более двух ветвей в локальном репозитории, и соединить с глобальным репозиторием GitHub и настройкой доступа к каждой ветке.**
2. **Создать новую ветку СКВ с помощью команды Git Branch, перейти в репозиторий и создать новый текстовый файл.**
3. **Создать локальный репозиторий, связанный с заданным глобальным репозиторием. Настроить параметры доступа к локальному репозиторию GitHub.**
4. **Выполнить загрузку изменений выполненных в локальном репозитории в глобальный репозитарий GitHub.**

Создать репозиторий в GitHub, далее создать папку на рабочем столе и загрузить репозиторий с GitHub в папку с помощью “git clone ссылка”. Далее создать любой текстовый файл в папке и проверить через git status

если будет так же, то все работает. Далее пишем “git add название\_файла.txt” . Потом git commit -m “тут пишем комментарий любой”. И в конце git push - теперь файл находится в репозитории GitHub.

1. **Создать репозиторий на сервисе GitHub. Создать несколько типов организаций с различными параметрами доступа.**
2. **Создать несколько учетных записей для разработчиков. Задать параметры, необходимые для отправки и слияния копии репозитория.**
3. **Создать приложение MVC в среде Visual Studio, обеспечивающее ввод данных о некотором объекте.**

как и в 30

1. **Проверьте правильность ввода данных с помощью отрицательного и положительного теста на стороне клиента.**

как и в 30

1. **Создать сервер (TCP/IP) в среде Visual Studio на языке C#.**

как и в 30

1. **Реализовать на сервере проверку данных (TCP/IP) в среде Visual Studio на языке C#.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Threading.Tasks;

using System.Net.Sockets;

namespace Server

{

class Program

    {

        TcpListener Listener;

        public Program(int Port)

        {

            Listener = new TcpListener(IPAddress.Any, Port);

            Listener.Start();

            while (true)

            {

                new Client(Listener.AcceptTcpClient());

            }

        }

        ~Program()

        {

            if (Listener != null) Listener.Stop();

        }

        static void Main(string[] args)

        {

            new Program(80);

        }

    }

    class Client

    {

        public Client(TcpClient Client)

        {

            string Html = "<html><body><h1> Its not working</h1></body></html>";

            string Str = "HTTP/1.1 200 OK\nContent-type: text/html\nContent-Length:" +Html.Length.ToString() + "\n\n" + Html;

          byte[] Buffer = Encoding.ASCII.GetBytes(Str);

            Client.GetStream().Write(Buffer, 0, Buffer.Length);

            Client.Close();

        }

    }

Рассмотрено на заседании ПЦК ОПД и ПМ специальности 09.02.07. ИСП (специалист по ИР)

Протокол от «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. № \_\_\_\_\_

Председатель ПЦК \_\_\_\_\_\_\_\_\_\_\_/Ю.В. Климова

Методист ОП№5 \_\_\_\_\_\_\_\_\_\_\_/А.А. Юмаева