***Теоретические вопросы:***

1. **IT-проект: основные понятия.**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.
2. **Вредоносное программное обеспечения типа «червь»: свойства, особенности.**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.
3. **Вредоносное программное обеспечения типа «Trojan»: 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.
4. **Вредоносное программное обеспечения типа «блокировщик экрана (WinLocker)»: свойства, особенности.**Вредоносные программы, блокирующие или затрудняющие работу с ОС, требуют перевод денег за разблокировку. После запуска прописывается в autorun, блокирует сочетания клавиш, explorer подменяется вирусным файлом в реестре.
5. **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.

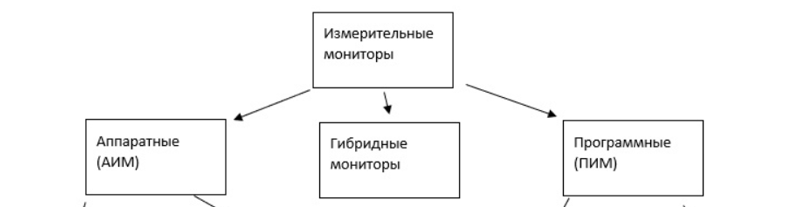
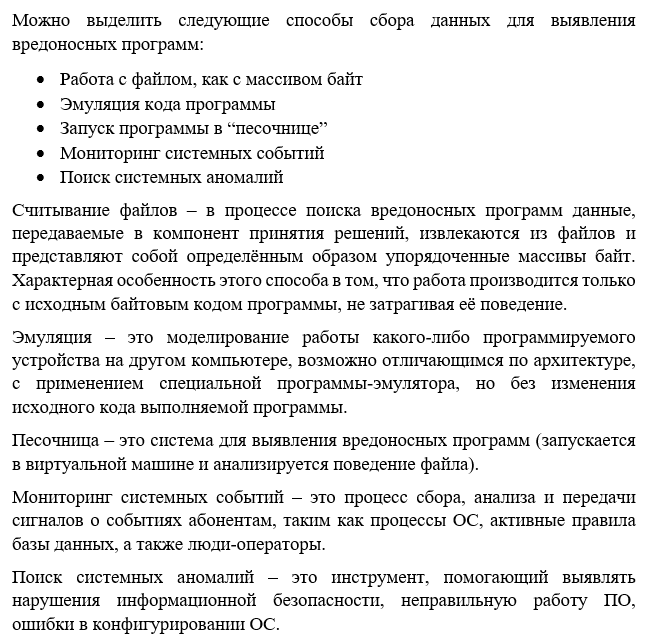
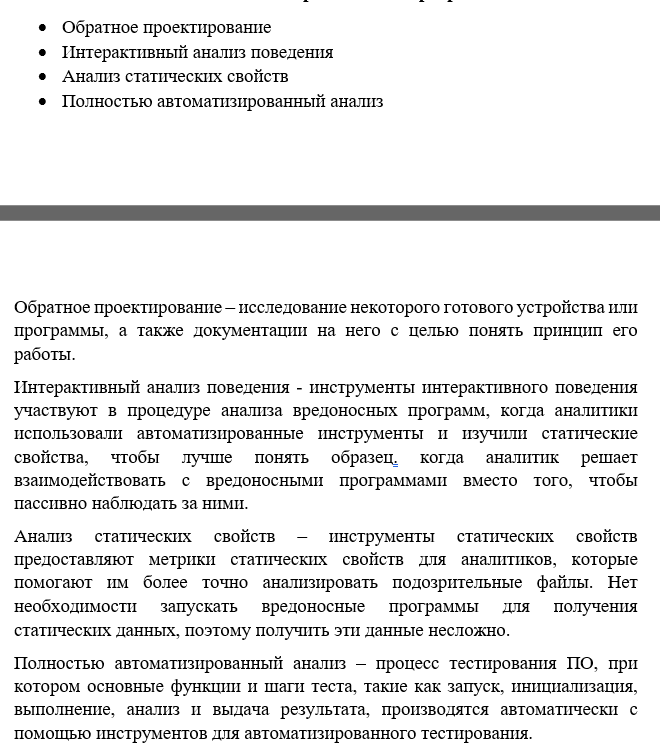
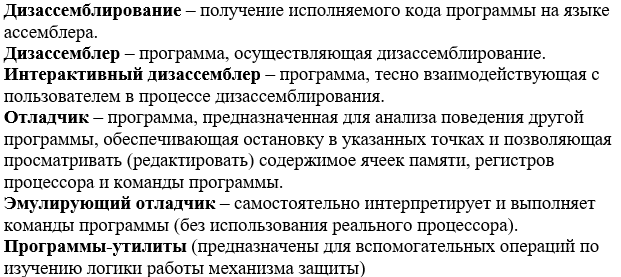
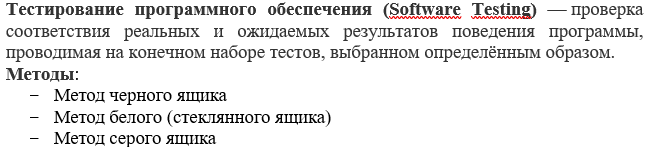
1. **Измерительные методы: назначение.**Измерение параметров потребления программами ресурсов вычислительной системы с целью дефектов производительности,

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.**

* Исследуемую ВС с установленными программами;
* Средства регистрации параметров потребляемых ресурсов при выполнении данной рабочей нагрузки;
* Архив для хранения результатов многочисленных измерений;
* Результаты измерений обрабатываются некоторой 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.
2. **Измерительные мониторы: виды, основное назначение.  
   ***АИМ* - 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* - 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.
3. **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.
4. **Динамическое тестирование: общие сведения, значение.**Динамическое тестирование – подразумевает достаточно полную структурную и функциональную проверку выполнения программы.
5. **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.
6. **Классификация анализа вредоносных программ.  
   **
7. **Этапы изучения вредоносного программного обеспечения и их описание.  
   **
8. **Основные инструменты для исследования кода программ: виды, свойства, значение.  
   **
9. **Файловый монитор (FileMonitor): возможности, предназначение.**File Monitor может отследить работу защищенной программы с файлами и обнаружить ключ (пароль), хранящийся в некотором файле.
10. **Методы, применяемые для обнаружения модифицированного кода. Преимущества и недостатки их применения.**Для обнаружения модифицированного кода традиционно применялись следующие методы: − подсчет контрольных сумм критических участков; − использование контрольной суммы всего кода для расшифровки некоторого фрагмента; − многопроходная расшифровка кода с ключом, вычисляемым на основе контрольной суммы всего кода либо критического участка; − использование корректирующих кодов, позволяющих определить местоположение контрольного байта.
11. **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.
12. **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.
13. **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.
14. **Понятие тестирования программ. Назначение, методы.  
    **
15. **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.
16. **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).
17. **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.  
    Черный ящик может представлять собой набор классов или модулей с известными внешними интерфейсами, но недоступным исходным кодом.
18. **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.
19. **Определение и основные особенности для термина: «эмулирующий отладчик».**Эмулирующий отладчик – самостоятельно интерпретирует и выполняет команды программы (без использования реального процессора).
20. **Методы проникновения вредоносного программного обеспечения на персональный компьютер.**Флеш-накопители (флешки), электронная почта, системы обмена мгновенными сообщениями, веб-страницы, интернет и локальные сети, черви.
21. **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.

Такая программа загружает рекламу автоматически. Рекламные закладки 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.

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.

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 - 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 - 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.

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.

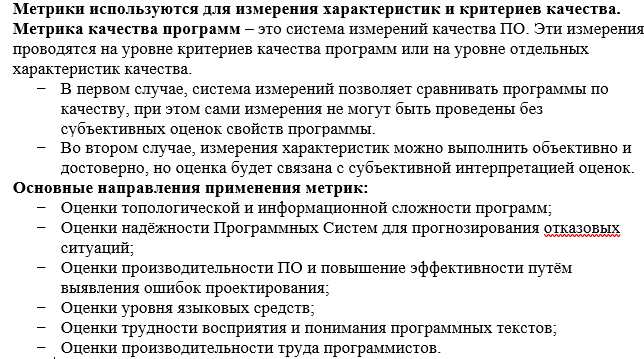
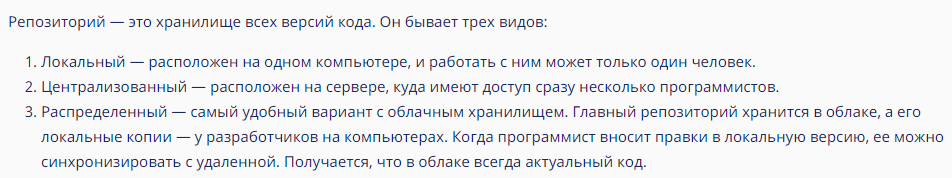
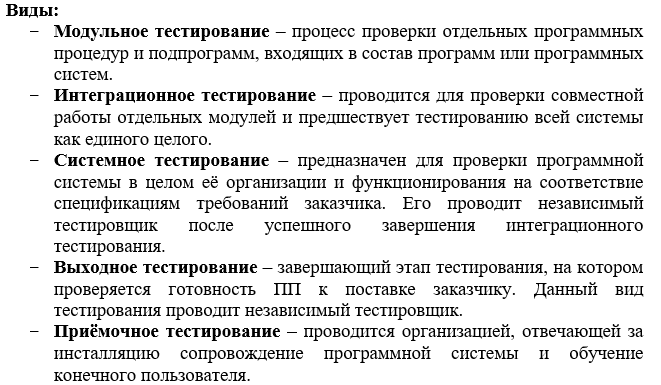
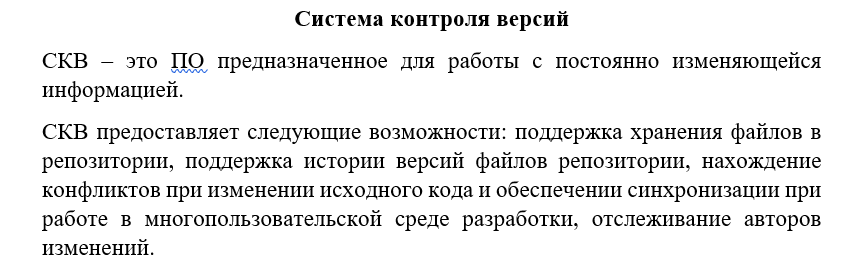
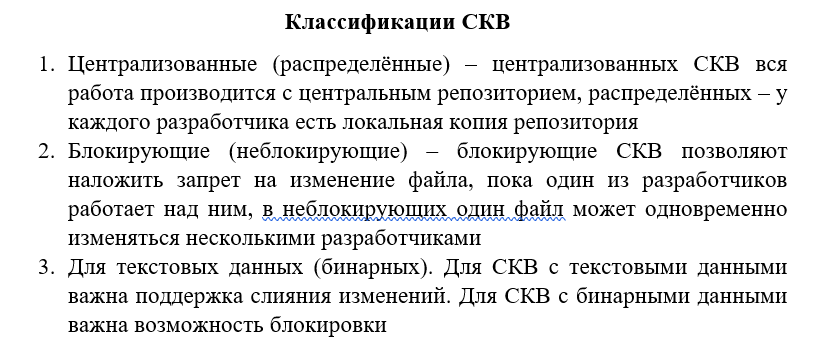
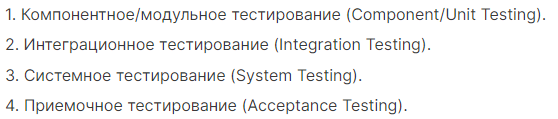
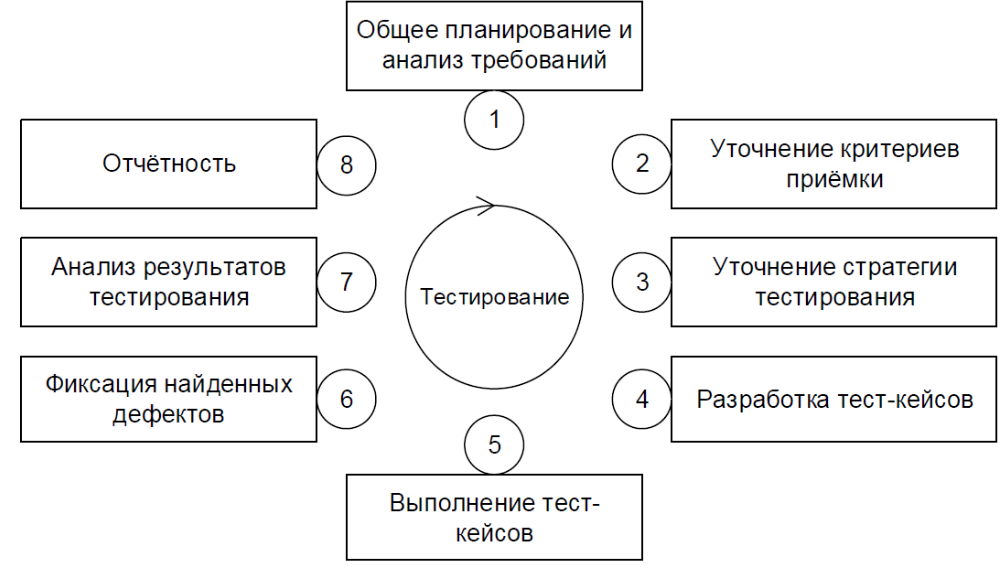
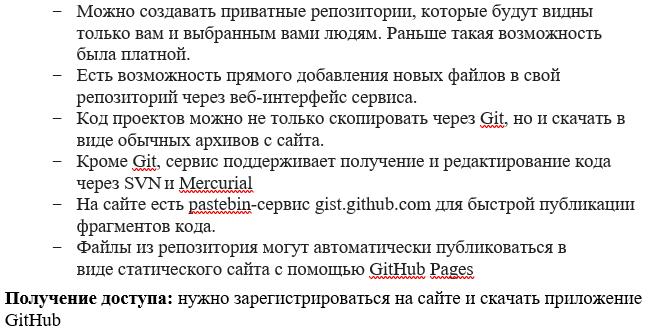
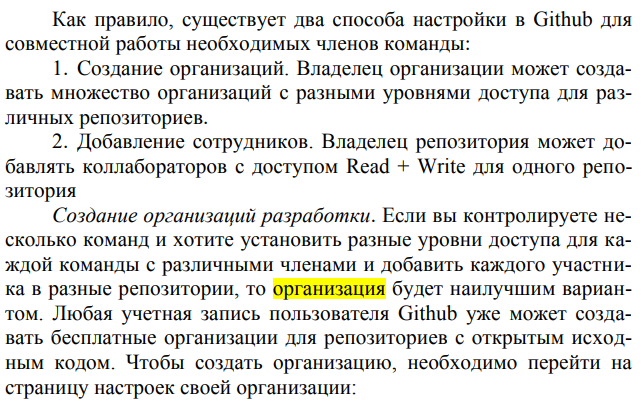
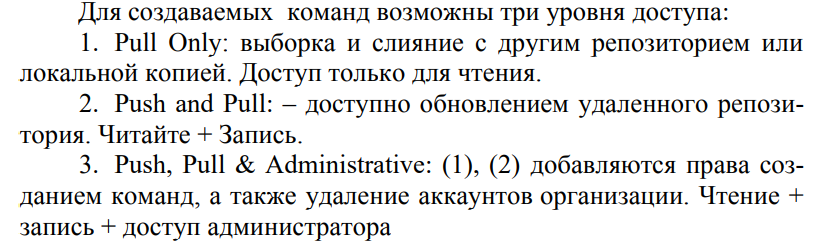
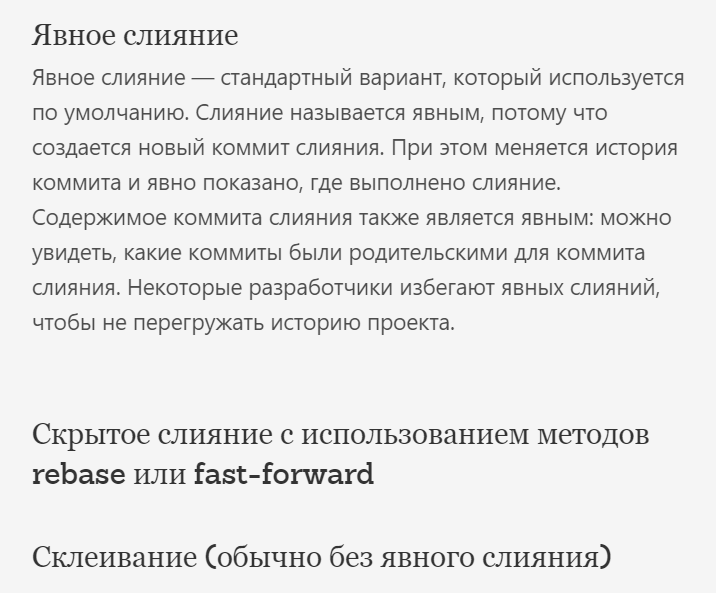
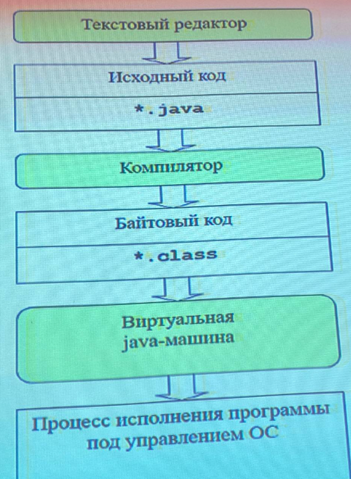
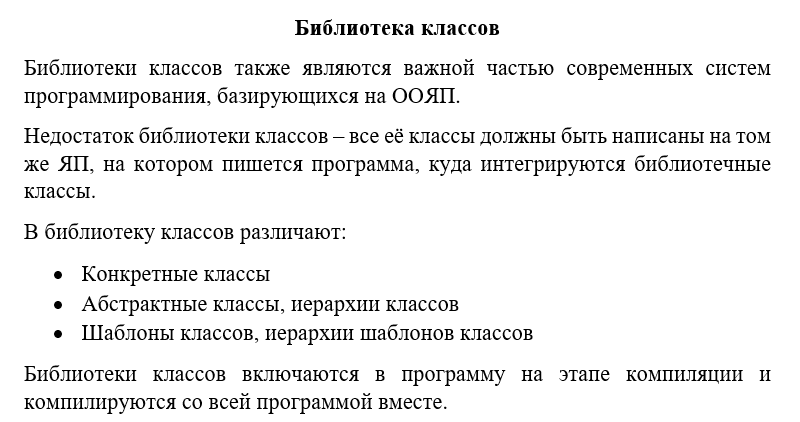
Троянская программа, также известная как «троянский конь» - это тип вредоносного ПО, которое маскируется под обычный файл или программу, чтобы обманным путем заставить пользователей загружать и устанавливать вредоносное ПО. Троян может предоставить злоумышленнику удаленный доступ к зараженному компьютеру. 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 - 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.  
   **
2. **Хранилище (Repository): основные понятия.  
   **
3. **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.
4. **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.  
   
5. **Понятия управления контроля версиями, применение и функции системы контроля версий..  
   **Основной областью применения СКВ является коллективная разработка (это программа и область применения которой не ограничивается по программированию).
6. **Классификация систем контроля версий.  
   **
7. **Функциональное значение контроля версий типа SVN. Современные подходы к анализу ПО(определение анализа ПО + примеры).**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. НЕ ВСЁ ОТВЕТИЛ
8. **Уровни тестирования: описание. Цикл тестирования: иерархическое древо: графическое отображение.  
     
   **
9. **Импорт и экспорт настроек: общие сведения, применение.**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.
10. **Обратное проектирование: основные понятия, виды применяемых инструментов и описание процесса.**Обратное проектирование – исследование некоторого готового устройства или программы, а также документации на него с целью понять принцип его работы.  
    Основными инструментами для исследования программ являются дизассемблеры и отладчики.
11. **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.
12. **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.
13. **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 измененные модули и наборы тестов сохраняются в БД проекта.
14. **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 отчет о ходе тестирования при завершении цикла тестирования.
15. **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 отчет о ходе тестирования.
16. **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
17. **Понятие тестирования методом «серый ящик» и описание процесса.**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.
18. **Получение доступа к GitHub. Основные функции и принцип GitHub.  
    **
19. **Организация разработки в сервисе Github.**Это план развития проекта и регулирование должностных обязанностей команды  
    
20. **Описание уровней доступа GitHub.  
    **
21. **Характеристика способов слияния копий репозитория.  
    **
22. **Модели создания кода в репозитории: описание и графическое представление.**
23. **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.
24. **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.
25. **Java: 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.
26. **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.
27. **Библиотека классов: понятие, доступ к членам класса.  
    **
28. **Язык Ассемблера: возможности языка, преимущества и недостатки.** Язык ассемблера - машинно-ориентированный язык программирования низкого уровня. Представляет собой систему обозначений, используемую для представления в удобно читаемой форме программ, записанных в машинном коде. Его команды прямо соответствуют отдельным командам машины или их последовательностям.

*Достоинства:*  
Обеспечение максимального использования специфических возможностей конкретной платформы, что также позволяет создавать более эффективные программы с меньшими затратами ресурсов.  
Язык ассемблера используется для создания «прошивок» [BIOS](https://www.tadviser.ru/index.php/%D0%A1%D1%82%D0%B0%D1%82%D1%8C%D1%8F:BIOS).

Существует возможность исследования других программ с отсутствующим исходным кодом с помощью дизассемблера.

Обеспечение максимального использования специфических возможностей конкретной платформы, что также позволяет создавать более эффективные программы с меньшими затратами ресурсов.

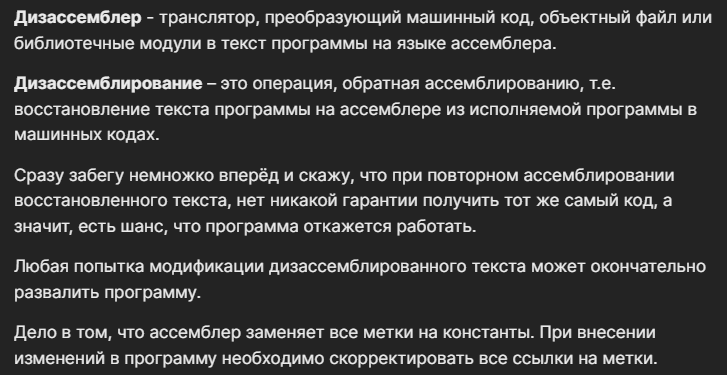
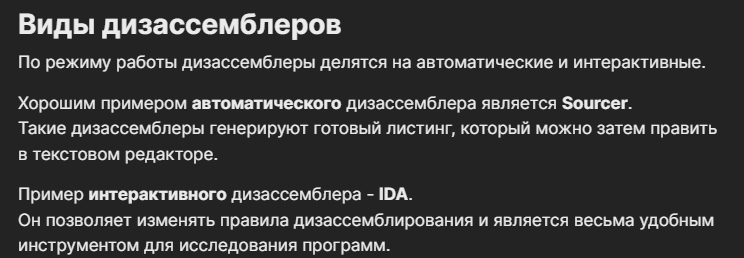
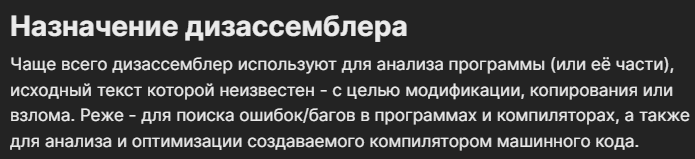
*Недостатки:*

Как правило, меньшее количество доступных библиотек по сравнению с современными индустриальными языками программирования.

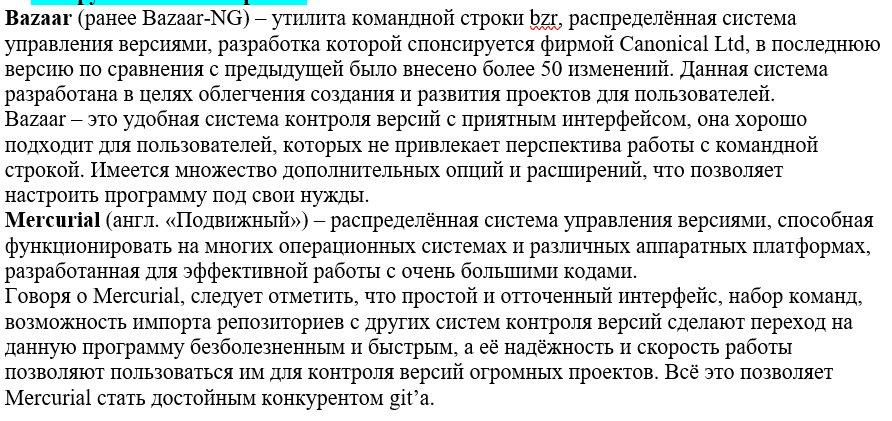
Отсутствует переносимость программ на компьютеры с другой архитектурой и системой команд (кроме двоично-совместимых).

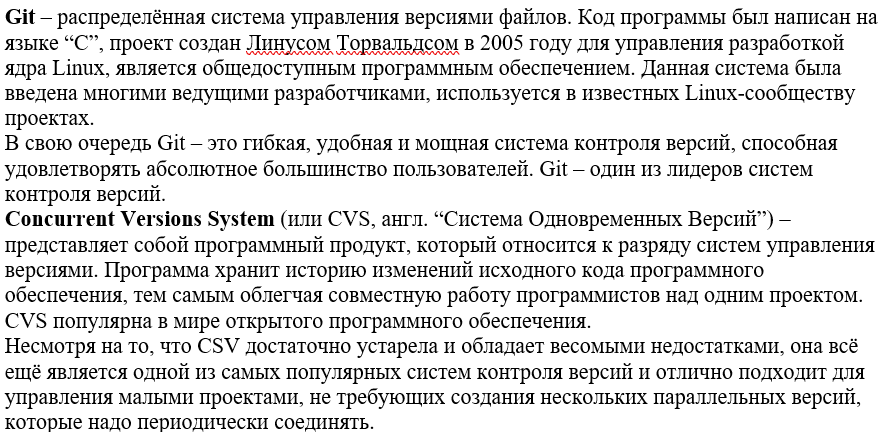
Главное преимущество ассемблера практически полностью нивелируется хорошей оптимизацией в современных компиляторах языков высокого уровня.

1. **Сегментная адресация: определение, примеры применения.  
   **
2. **Дизассемблер: возможности программы, преимущества и недостатки.**

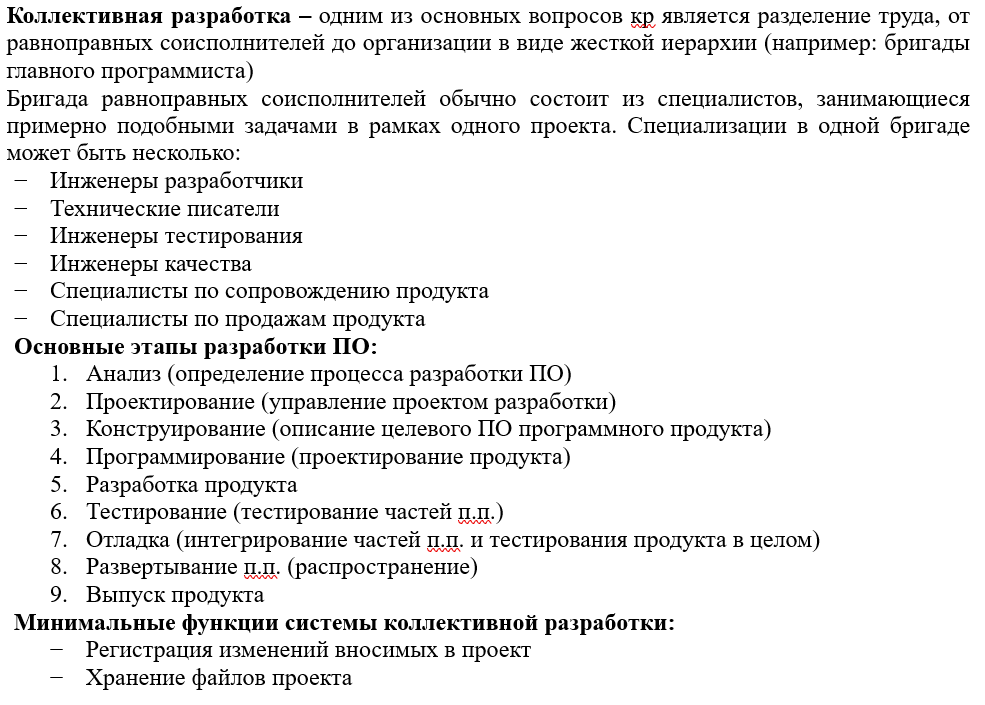
**  
  
**

1. **Мониторинг работоспособности системы контроля версий: назначение и инструменты мониторинга.**

****

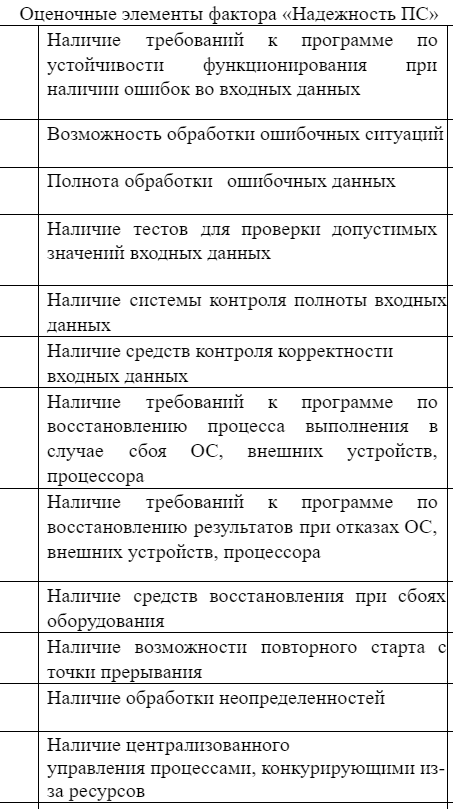
****

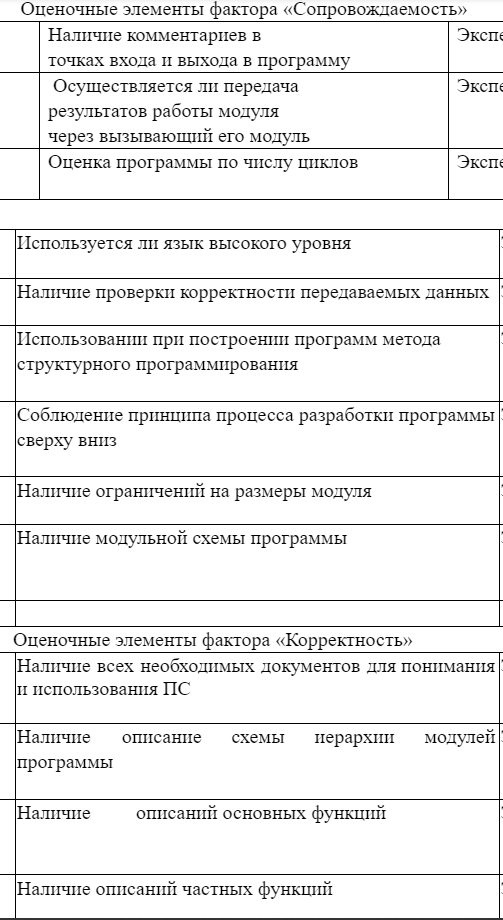
1. **Коллективная разработка: характеристика, структура.**

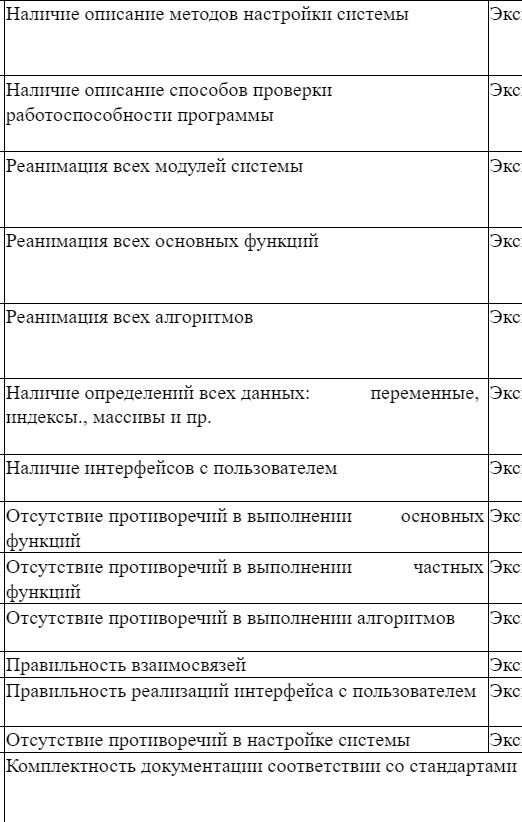
**

***Практические задания:***

Примерные функциональные особенности для сравнения программ или описания одной программы:







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 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

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 “ветви” - для выбора ветки. Далее cp “первый” “второй”. git commit -m “a”. git push.

Далее в самом GitHub нажать на commits.

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

* 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.**

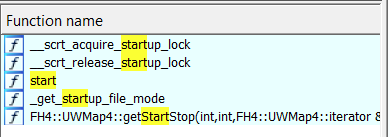
* создаем 2 раза ветки (git branch final и master) git checkout final. загружаем файл(так же по разу с каждой ветки)(git add “”)(git commit -m “”) (без git push) далее git merge “ветка которая не выбрана(final или master)”.

1. **Разработать в репозитории с помощью функции «menu» две ветви проекта.**

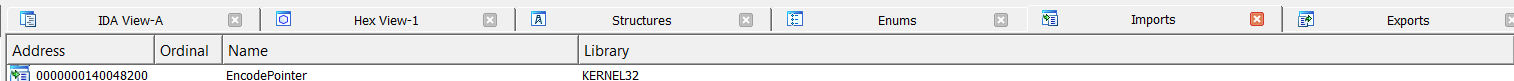
* git branch 1. git branch 2

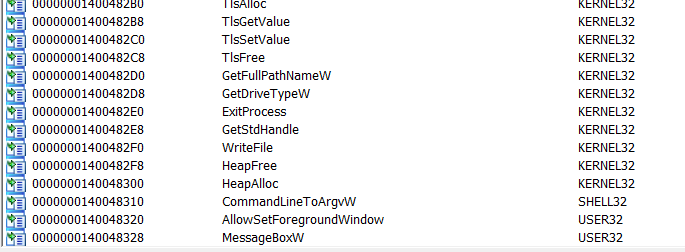
1. **В программе IDA pro 8.0 дизассемблировать файл «6.exe», находящийся в папке «экзамен» на рабочем столе. Отразить в виде таблицы в MS Word библиотеки, к которым он обращается, и указать, где начинается работа программы.**

* 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).**

* создать 2 папки, в одной TortoiseSVN -> create repository here. Далее заходим в папку и repo-browser. Далее создаем там папку. Выбираем эту папку где находится репозиторий и нажимаем SVN Checkout и выбираем 2 папку.

1. **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.**

* git branch “название новой ветки”. git checkout fish - для выбора ветки. git commit -m “a”. git push.

Далее в самом GitHub нажать на commits.

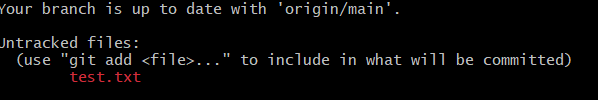
1. **Создать новую ветку СКВ с помощью команды Git Branch, перейти в репозиторий и создать новый текстовый файл.**

* git branch new. git checkout new. git add файл. git commit -m “a”. git push.

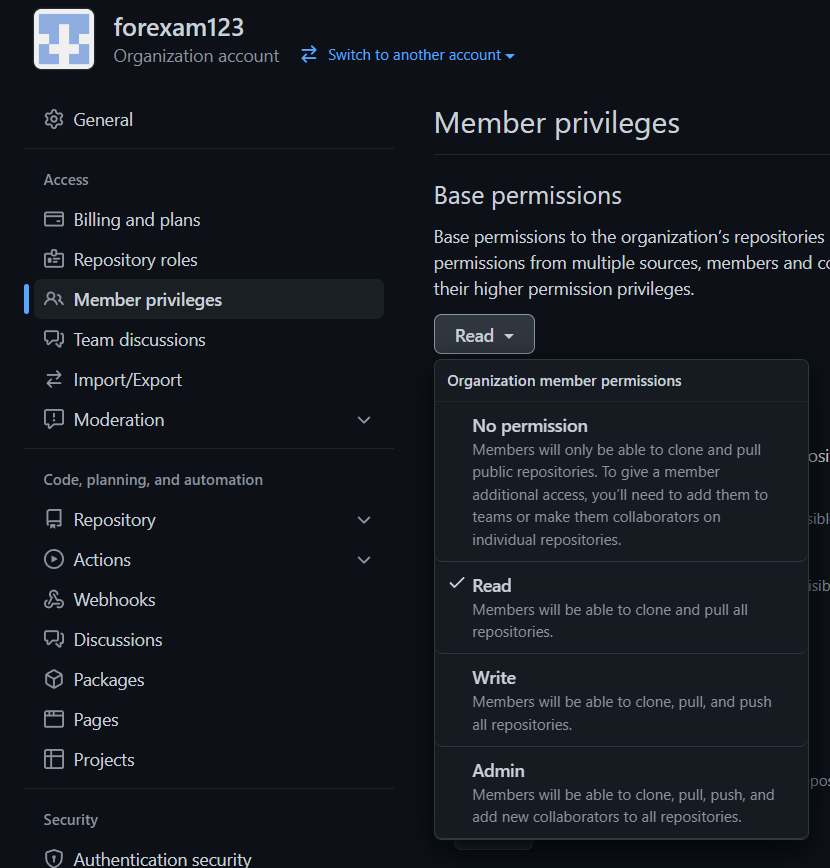
1. **Создать локальный репозиторий, связанный с заданным глобальным репозиторием. Настроить параметры доступа к локальному репозиторию GitHub.**

* git clone ссылка. Про параметры не известно.

1. **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.**

****

1. **Создать приложение MVC в среде Visual Studio, обеспечивающее ввод данных о некотором объекте.**

* как и в 30

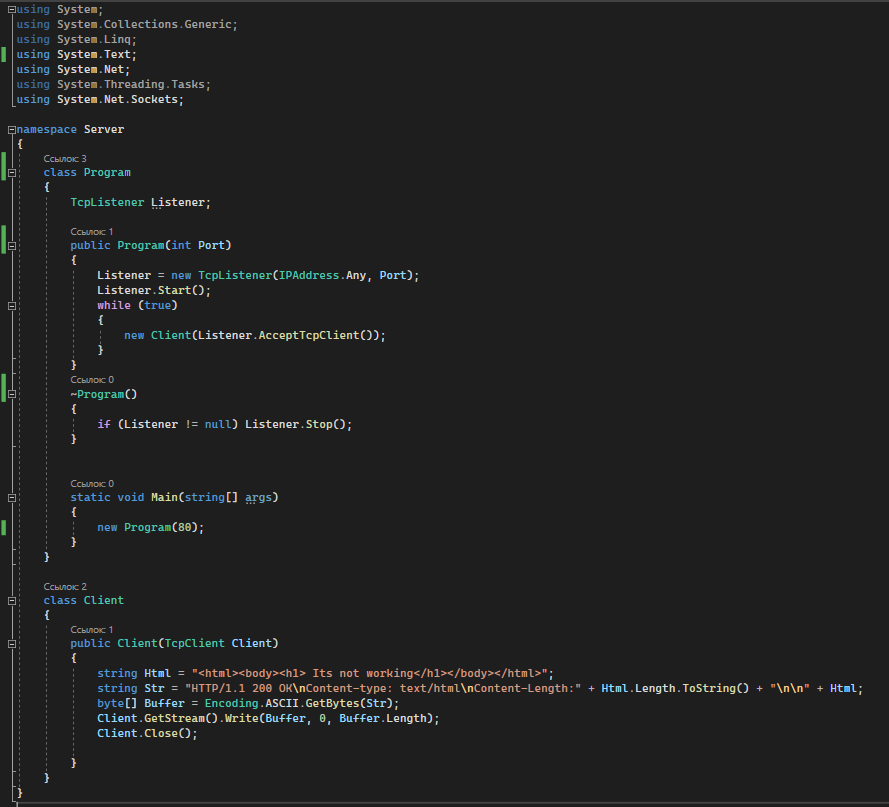
1. **Проверьте правильность ввода данных с помощью отрицательного и положительного теста на стороне клиента.**

* 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();

}

}

Обычное консольное приложение. Не забыть добавить библиотеки (using), как на скрине.

Далее запускаем сайт по ip [127.0.0.1](http://127.0.0.1/)