**1. UMA-мультипроцессоры** UMA (Uniform Memory Access) – однородный доступ к памяти, когда все процессоры имеют связь со всеми модулями памяти;

**2. NUMA-мультипроцессоры** NUMA (NonUniform Memory Access) – неоднородный доступ к памяти, у каждого процессора имеется промежуточная память (например кэш);

**3. Мультипроцессоры UMA с общей шиной** Наиболее популярный вид UMA-мультипроцессоров имеют одну общую шину [3, 4]. От двух и более микропроцессоров и от одного и нескольких модулей памяти используют эту шину для обмена данными. Этот процесс выполняется следующим образом, процессору для считывания слова из памяти, необходимо проверить, что шина не занята. Если на шине не наблюдается обмен данными, процессор отправляет адрес необходимого слова на шину, устанавливает некоторые управляющие сигналы и ожидает реакцию памяти, которая выдает данные на шину по запрошенному адресу. В случае занятости шины, процессор находится в режиме ожидания до тех пор, пока шина не освободится. В данной схеме имеется существенный недостаток. Когда используется от двух до трех процессоров, контролировать доступ к шине не составляет труда, сложности появляются, при количестве процессоров 32 или 64. Производительность системы в данном варианте в основном характеризуется пропускной способностью шины, и большинство процессоров вынуждены значительное время простаивать.

**4. Мультипроцессоры UMA с координатными коммутаторами** Даже при самом удачном кэшировании использование одной шины сводит масштаб UMA-мультипроцессора всего лишь к 16 или 32 центральным процессорам. Чтобы преодолеть этот барьер, нужен другой тип сети обмена данными. Простейшая схема подключения n центральных процессоров к k модулям памяти — это координатный коммутатор. В каждом пересечении горизонтальной (входной) и вертикальной (выходной) линий стоит элемент коммутации, небольшой выключатель, который может пропускать или не пропускать электрический сигнал, то есть быть включенным или выключенным в зависимости от того, должны или не должны соединяться горизонтальная и вертикальная линии. На рис. 8.3, а изображены три одновременно включенных элемента, позволяющие соединяться следующим парам «центральный процессор — модуль памяти»: (001, 000), (101, 101) и (110, 010). У координатного коммутатора есть одно великолепное свойство: он представляет собой неблокирующуюся сеть (nonblocking network), то есть такую, где ни одному центральному процессору никогда не будет отказано в необходимом ему подключении по причине занятости какого-то элемента коммутации или линии (если предположить, что свободен сам требуемый модуль памяти).

**5. Мультипроцессоры UMA с многоступенчатыми коммутаторными сетями** Принципиально другая архитектура мультипроцессоров базируется на простых коммутаторах 2x2, Рис. 4(а). У такого коммутатора два входа и два выхода. Сообщения, поступающие по любой из входных линий, могут переключаться на любую выходную линию. Сообщения в рассматриваемом нами мультипроцессоре будут состоять из четырех частей 1) Поле Module (модуль) указывает модуль памяти. 2) Поле Address (адрес) указывает адрес внутри модуля. 3)Поле Opcode (код операции) указывает операцию, то есть READ (чтение) или WRITE (запись). 4) Необязательное поле Value (значение) может содержать операнд, например 32- разрядное слово, которое должно быть записано операцией WRITE. С помощью данного коммутатора 2x2 можно построить самые различные многоступенчатые коммутаторные сети.

**6. Основным признаком PVP-систем является**

Наличие специальных векторно-конвейерных процессоров, в которых предусмотрены команды однотипной обработки векторов независимых данных, эффективно выполняющиеся на конвейерных функциональных устройствах.

**7. Много ЦПУ, общая периферия, одна ОС, локализованное расположение, связь через общую память**

**8. Много ЦПУ, общая периферия, несколько ОС, локализованное расположение, связь по выделенным линиям**

**9. Много ЦПУ, периферии, разные ОС, разное расположение, связь по сети**

**10. Каждому ЦП - свою ОС. Такой способ организации ОС в мультипроцессорах имеет недостаток** 1. Системные вызовы в каждом ЦП обрабатываются самостоятельно – дублирование управляющий структур. 2. Невозможность совместного использования процессоров – простаивание ЦП. 3. Невозможность совместного использования ОЗУ – неоптимальное распределение памяти. 4. При использовании дискового КЭШа разделяемых ВнУ, невозможно обеспечить когерентность.

**11. В модели "хозяин-подчиненный"** В памяти находится одна копия операционной системы, выполнять которую может только один центральный процессор

**12. Симметричные мультипроцессоры** Система, состоящая из нескольких однородных процессоров и массива общей памяти (обычно из нескольких независимых блоков). Все процессоры имеют доступ к любой точке памяти с одинаковой скоростью.

**13. В распределенной ОС** Пользователь не должен знать о том, на какой машине выполняется его работа. Каждый компьютер сети выполняет часть функций глобальной ОС, а все компьютеры сети работают в тесной кооперации друг с другом для эффективного использования всех ее ресурсов

**14. Серверная ОС** Серверные операционные системы — специально разработанные или доработанные операционки, служащие для работы веб технологий (ну и не только веб, но в основном). На платформе серверных ОС работает специальное программное обеспечение (программная среда), позволяющее действовать скриптам (оболочкам) сайтов и баз данных, а так же иметь доступ к их файлам через свой компьютер. В серверные ОС, в основном, не ставят ни какого лишнего программного обеспечения, кроме необходимого и зачастую отсутствует визуальная оболочка, а общаться с сервером приходится через командную строку и специальные программы — клиенты SSH (FileZilla и тд.)

**15. Сеть с равноправными узлами называется** Одноранговые - в них все узлы равноправны; поскольку в общем случае под клиентом понимается объект (устройство или программа), запрашивающий некоторые услуги, а под сервером - объект, предоставляющий эти услуги, то каждый узел в одноранговых сетях может выполнять функции и клиента, и сервера.

**16. Сеть, управляемая одним или несколькими серверами, называется** Сеть с выделенным сервером (англ. client/server network) — это локальная вычислительная сеть (LAN), в которой сетевые устройства централизованы и управляются одним или несколькими серверами. Индивидуальные рабочие станции или клиенты (такие, как Персональный компьютер) должны обращаться к ресурсам сети через сервер(а).

**17. Сети, сочетающие в себе качества одноранговых и сетей на основе сервера, называются** Комбинированные сети

**18. Ethernet это** это протокол, который позволяет сетевым устройствам отправлять и получать данные другим устройствам в той же сети. Он обычно используется в сетях, где местоположение является фактором

**19. Сетевая служба это** Совокупность серверной и клиентской частей ОС, предоставляющих доступ к конкретному типу ресурса компьютера через сеть

**20. К сетевым средствам ОС относятся** - средства предоставления локальных ресурсов и услуг в общее пользование —серверная часть ОС; - средства запроса доступа к удаленным ресурсам и услугам — клиентская часть ОС; - транспортные средства ОС, которые совместно с коммуникационной системой обеспечивают передачу сообщений между компьютерами сети.

**21. Гипервизор первого типа это** Гипервизор — инструмент параллельного управления несколькими операционными системами (ОС) на одном компьютере (хосте). Он позволяет создать несколько виртуальных устройств на одном физическом. Такие ОС называются гостевыми. Они могут работать одновременно и независимо друг от друга. Каждая имеет собственную операционную систему, собственные выделенные ресурсы и память. Гипервизоры 1-го типа ставятся прямо поверх «железа». Их собственные встроенные базовые драйверы позволяют работать с хост-компьютером напрямую на аппаратном уровне. Отсюда еще одно название — аппаратные. То есть, гипервизоры типа 1 работают на низком уровне, максимально близкому к ядру хоста. К таким относятся: VMware ESX, Citrix XenServer, Microsoft Hyper-V.

**22. Гипервизор второго типа это** Гипервизор — инструмент параллельного управления несколькими операционными системами (ОС) на одном компьютере (хосте). Он позволяет создать несколько виртуальных устройств на одном физическом. Такие ОС называются гостевыми. Они могут работать одновременно и независимо друг от друга. Каждая имеет собственную операционную систему, собственные выделенные ресурсы и память. Гипервизоры 2-го типа отличаются от первых наличием «прослойки» между ними и хостом. Они устанавливаются поверх базовой операционной системы. Или же с помощью нее они предоставляют гостевому коду доступ к ресурсам. Грубо говоря, чтобы запустить такой гипервизор, вам надо установить его поверх имеющейся ОС. Пример гипервизоров: Microsoft Virtual PC, VMware Workstation, QEMU, Parallels, VirtualBox.

**23. …aaS означает** В облачных технологиях применяется единая классификация моделей вычислений с суффиксом «…aaS» (…as a Service), что в переводе значит: «…как услуга».

**24. IaaS модель** IaaS (Infrastructure as a Service) — это инфраструктура как услуга. Инфраструктура включает вычислительные ресурсы: виртуальные серверы, хранилища, сети. Их можно представить в виде виртуальных «компьютеров», на которые можно установить операционную систему, программное обеспечение, приложения и.т.д. Поставщик IaaS уже все купил и подготовил, а его клиентам нужно только подключиться к этим вычислительным мощностям через интернет и использовать их для своих задач.

**25. PaaS в сравнении с IaaS**

**26. Основной недостаток SaaS** Минусы модели: -необходимость в постоянном и бесперебойном подключении к сети; -высокий уровень защиты данных только при сотрудничестве с надежными сервисами; -ограниченный функционал в сравнении с классическим программным обеспечением.

**27. Такой модели развертывания нет**

есть следующие модели развертывания: частное облако, публичное облако, гибридное облако и общественное облако.

**28. Частные облака** это облачная инфраструктура, которая принадлежит одной компании. Она развернута на базе собственной физической инфраструктуры компании или на арендованном оборудовании. Компания, которая использует частное облако, не делит ни с кем физические и виртуальные ресурсы, они принадлежат ей.

**29. Google Cloud или Microsoft Azure предоставляют (?)** облачная платформа, которая предоставляет пользователям инструменты для вычислений, хранения информации, размещения приложений.

**30. IaaS — это** IaaS (Infrastructure as a Service) — это инфраструктура как услуга. Инфраструктура включает вычислительные ресурсы: виртуальные серверы, хранилища, сети. Их можно представить в виде виртуальных «компьютеров», на которые можно установить операционную систему, программное обеспечение, приложения и.т.д. Поставщик IaaS уже все купил и подготовил, а его клиентам нужно только подключиться к этим вычислительным мощностям через интернет и использовать их для своих задач.

**31. PaaS — это** PaaS, или «Платформа как услуга», в первую очередь предназначена для разработчиков и компаний, создающих собственное программное обеспечение. Такое решение позволяет разрабатывать, запускать и управлять ПО, не беспокоясь об управлении инфраструктурой. С помощью PaaS компаниям нужно управлять только приложениями и данными. Провайдер PaaS управляет операционной системой, средой выполнения, промежуточным ПО, виртуализацией, серверами, хранилищем и сетью. Другими словами, разработчикам не нужно начинать с нуля при разработке программного обеспечения. Таким образом, разработчики могут больше сосредоточиться на таких аспектах процесса разработки приложения, как тестирование или развертывание приложения.

**32. SaaS — это** Сервисная модель, в рамках которой подписчик получает онлайн-доступ к нужному софту, но не отвечает за его установку, обновление и поддержку работоспособности.

**33. К характеристикам облачных технологий не относится**

**34. Arduino — это** Это небольшая плата с собственным процессором и памятью. На плате также есть пара десятков контактов, к которым можно подключать всевозможные компоненты: лампочки, датчики, моторы, чайники, роутеры, магнитные дверные замки и вообще всё, что работает от электричества. В процессор Arduino можно загрузить программу, которая будет управлять всеми этими устройствами по заданному алгоритму. Таким образом можно создать бесконечное количество уникальных классных гаджетов, сделанных своими руками и по собственной задумке. Торговая марка аппаратно-программных средств для построения и проектирования простых систем, моделей и экспериментов в области электроники, автоматики, автоматизации процессов и робототехники.

**35. Industry 4.0 характеризуется** Эффективностью, использованием цифровых технологий, соответствием принципам устойчивого развития, обеспечением постоянной связи между всеми участниками процесса и «умными» идеями. Четвёртая промышленная революция (The Fourth Industrial Revolution) — прогнозируемое событие, массовое внедрение кибер физических систем в производство и обслуживание человеческих потребностей, включая быт, труд и досуг.

**36. IoT – это** Inthernet of Things (Интернет вещей) — это сеть физических объектов, которые имеют встроенные технологии, позволяющие осуществлять взаимодействие с внешней средой, передавать сведения о своем состоянии и принимать данные извне.

**37. RFID-метка – это** (Radio Frequency IDentification, радиочастотная идентификация) — способ автоматической идентификации объектов, в котором посредством радиосигналов считываются или записываются данные, хранящиеся в так называемых транспондерах, или RFID-метках. Любая RFID-система состоит из считывающего устройства (считыватель, ридер или интеррогатор) и транспондера (он же RFID-метка, иногда также применяется термин RFID-тег).

**38. К предпосылкам развития Интернета вещей можно отнести** Оптимизация напрямую зависит от скорости передачи и обработки данных. Когда сбором и обработкой данных занимаются люди, то на это могут уйти дни.

**39. К примерам проектов IoT можно отнести** Облачные платформы IoT, усовершенствованная измерительная инфраструктура (AMI), мониторинг производственного потока, умные дверные замки, приложения для умного дома, термостаты WiFi, домашние роботы, домашняя система раннего оповещения о землетрясениях, транспорт, здравоохранение, отслеживание персонала, умный офис, промышленное производство и т.д.

**40. К типичным примерам применения технологии IoT относится** носимые устройства (wearables) • умный дом (smart home) • умная одежда • smart TV • умные девайсы для животных • умный транспорт (connected cars) и беспилотники • умный город (smart city) • страховая телематика (usage-based insurance) • умные рабочие места • умные электросети (smart grid) • умные заводы (smart factory, industrial internet, IIoT) • точное земледелие • умные скважины smart well • геолокационный маркетинг, в том числе биконы • умные склады

**41. Ключевая особенность технологии IoT – это** Обмен информацией, взаимодействие устройств друг с другом без участия человека

**42. Основной недостаток технологии IoT** o Необходимость огромной предварительной подготовки. Нужно не только научить устройства идентифицировать и маркировать различные объекта, но и организовать общение устройств разной марки. o Из первого пункта вытекает проблема совместимости и интеграции данных. Сотни стандартов передачи и обработки данных ограничивают возможности взаимосвязи. Сейчас всё еще не установлено, должны ли устройства работать по одному стандарту или правила надо разрабатывать исходя из конкретного поставщика интернет-услуг. o Наиболее важным недостатком в отношении внедрения IoT является вопрос о конфиденциальности. Умные домашние устройства получают много данных о пользователе. Эта информация включает в себя личные графики, потребительские привычки, расписание приема лекарств и даже местоположение пользователя в любой момент времени. Если эти данные попадут не в те руки, людям может быть нанесен большой вред и ущерб. Поэтому шифрование – важный момент развития Интернета вещей. o Данная концепция относится к сложным системам. И у них всегда есть вероятность отказа от работы или совершения ошибки. К примеру, устройство может случайно уведомить всех членов семьи об отсутствии продуктов в холодильнике или сломанной лампочке. Или неправильно воспримет сигналы принтера и закажет цветные чернила вместо черно-белых.

**43. Технология IIoT – это** Industrial Internet of Things (Промышленный Интернет Вещей) — это система объединенных компьютерных сетей и подключенных к ним промышленных (производственных) объектов со встроенными датчиками и программным обеспечением для сбора и обмена данными, с возможностью удаленного контроля и управления в автоматизированном режиме, без участия человека.

**44. К основным компонентам технологии IoT не относится** В архитектуре IoT обычно выделяют три основных элемента. Вещи (Things) – Оборудование, которое имеет устройства для подключения (проводного или беспроводного) к сети. Сеть (Network) – Аналогично роутеру в вашем доме сеть объединяет и подключает любое количество устройств к облаку. Облако (Cloud) – Удаленный сервер в дата-центре, консолидирующий и надежно хранящий ваши данные. Вещи создают потоки данных – байты собранной с помощью датчиков такой простой информации, как температура, влажность или положение. Часто эту информацию называют Малыми данными (Little Data), так как она небольшая по объему. Большой массив разных устройств передает огромное количество различных Малых данных через сеть в облако, где они консолидируются, со временем данных становятся все больше и больше. И тогда их называют уже термином Большие данные (Big Data). Благодаря IoT появилось много языков и протоколов – от уже традиционных Wi-Fi и Bluetooth до новых LoRaWAN™ и Sigfox. Каждый из них подходит для разных применений в зависимости от нескольких факторов: Скорость обмена данными – Какой объем информации необходимо передавать? Энергопотребление – Например, носимая электроника может работать небольшое количество времени без подзарядки Дальность – На какое расстояние данные будут передаваться – пара метров или несколько километров? Частота – Какая частота для нас доступна? Базовые элементы делятся на несколько типов: сенсоры, актуаторы и гейты. Сенсоры – получение информации Актуаторы – воздействие на внешнюю среду Гейты – отвечают за логику Как вариант – встроенные технологии, позволяющие обеспечивать взаимодействие с внешней средой

**45. Arduino Uno – это** Самая популярная плата. На ней 14 цифровых и 6 аналоговых входов, 32 КБ постоянной и 2 КБ оперативной памяти, процессор частотой 16 МГц, порт USB. Не сравнить с современными смартфонами и компьютерами, но для знакомства с конструктором и создания простых систем этого вполне достаточно.

**46. Устройства IoT получают данные от внешней среды с помощью** Базовые элементы делятся на несколько типов: сенсоры, актуаторы и гейты. Сенсоры – получение информации Актуаторы – воздействие на внешнюю среду Гейты – отвечают за логику Как вариант – встроенные технологии, позволяющие обеспечивать взаимодействие с внешней средой

**47. Ключевая особенность микроконтроллера ESP8266 это** - Удобное подключение к компьютеру – через USB кабель, питание от него же; - Наличие встроенного преобразователя напряжения 3,3В; - Наличие 4 Мб флеш-памяти; - Встроенные кнопки для перезагрузки и перепрошивки; - Все порты выведены на плату на две гребенки с шагом 2,5 мм.

**48. Микроконтроллер – это** Микросхема, предназначенная для управления электронными устройствами. Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и ПЗУ. По сути, это однокристальный компьютер, способный выполнять относительно простые задачи.

**49. К исполнительной системе Windows не относится** относится служба сервера всё чего нет в скобках (диспетчер процессов и потоков, диспетчер виртуальной памяти, диспетчер ввода/вывода, диспетчер КЭШа, диспетчер объектов, библиотеки времени выполнения, Диспетчер конфигурации, Справочный монитор защиты).

**50. Ветвь HKEY\_CURRENT\_USER реестра хранит сведения о** пользователе, вошедшем в систему в настоящий момент. (меню Пуск)

**51. К ветвям реестра Windows-систем не относится** HKEY\_USER всё чего нет в скобках (HKEY\_CLASSES\_ROOT, HKEY\_CURRENT\_USER, HKEY\_LOCAL\_MACHINE, HKEY\_USERS, HKEY\_CURRENT\_CONFIG).

**52. Для закрытия объектов ядра Windows применяют функцию** CloseHandle

**53. Венгерская нотация — соглашение об именовании переменных, суть которого сводится к тому, что** (Добавляем префиксы) имена идентификаторов предваряются заранее оговорёнными префиксами, состоящими из одного или нескольких символов. При этом, как правило, ни само наличие префиксов, ни их написание не являются требованием языков программирования, и у каждого программиста они могут быть своими.

**54. К объектам ядра не относится** Критические разделы всё чего нет в скобках (Процессы, Потоки, Файлы, События, Семафоры, Мьютексы, Каналы, Файлы, проецируемые в память).

**55. Для описания дескриптора объекта в Windows API используется тип** HANDLE

**56. К системным процессам Windows не относится** Winword.exe всё чего нет в скобках (winlogon.exe, sms.exe, smss.exe, explorer.exe, alg.exe, spoolsv.exe, svchost.exe, ctfmon.exe, csrss.exe, Lsass.exe, wuauclt.exe, hkcmd.exe).

**57. Ядро ОС Windows находится в файле** Ntoskrnl.exe

**58. Большинство внутренних текстовых строк в Windows хранится в кодировке** Unicode UTF16

**59. Ветвь HKEY\_LOCAL\_MACHINE реестра хранит сведения о** аппаратной части аппаратной части ПК, о драйверах устройств, сведения о загрузке Windows.

**60. Загружаемый модуль режима ядра, предоставляющий низкоуровневый интерфейс для аппаратной платформы, на которой выполняется Windows** HAL (Он скрывает от операционной системы специфику конкретной аппаратной платформы (интерфейсов ввода-вывода, контроллеров прерываний, механизмов взаимодействия между процессорами и т. д.), то есть, все функции, которые зависят от аппаратной архитектуры и конкретной ЭВМ.)

**61. К DLL подсистемы Windows не относится** Psxdll.dll всё чего нет в скобках (Kernel32.dll, Advapi32.dll, User32.dll, Gdi32.dll).

**62. Гипервизор – это** (или монитор виртуальных машин) — программа или аппаратная схема, обеспечивающая или позволяющая одновременное, параллельное выполнение нескольких операционных систем на одном и том же хост-компьютере. Это процесс, который отделяет операционную систему компьютера и приложения от базового физического оборудования. Обычно представляет собой программное обеспечение, хотя создаются и встроенные гипервизоры, например, для мобильных устройств.

**63. Прием сообщений в главном цикле обработки сообщений в программах ОС Windows осуществляет функция** GetMessage

**64. Для инициации механизма перерисовки окна в Windows-приложениях нужно вызвать функцию** Чтобы спровоцировать перерисовку окна необходимо вызвать функцию: void InvalidateRect( HWND, //handle окна RECT\* //эта область требует перерисовки, BOOL); //нужно ли перед перерисовкой очищать область обновления Очистка производится сообщением WM\_ERASE\_BACKGROUND. После вызова функции InvalidateRect, окно не перерисовывается сразу (до WM\_PAINT). Перерисовка произойдет только при опросе программой очереди сообщений. Когда перерисовка требуется немедленно, то вслед за InvalidateRect вызывается функция: void UpdateWindow(HWND);

**65. Назначение оконной функции — это** обработка сообщений Windows. Организации адекватной реакции со стороны Windows-приложения на действия пользователя и поддержания в актуальном состоянии того окна приложения, сообщения которого она обрабатывает.

**66. Точкой входа в Windows-приложение является функция** WinMain

**67. Выберите неверное утверждение об асинхронных сообщениях в ОС Windows** Возврат к вызывающему коду происходит только после окончания обработки сообщения оконной процедурой всё чего нет в скобках (Передаются непосредственно окну, асинхронные посылаются прямо в оконную процедуру (send), если сообщение отправляется с помощью функции SendMessage, то оно является асинхронным).

**68. Какая структура данных используется для хранения оконных сообщений?** MSG ОЧередь

**69. Выберите неверное утверждение о синхронных сообщениях в ОС Windows** всё чего нет в скобках (Помещает в очередь сообщений программы, извлекаются и диспетчеризуются в цикле обработки сообщений, синхронные сообщения помещаются в очередь сообщений (post)). Сообщения ставятся в очередь синхронных сообщений вызовом функции PostQuitMessage

**70. К стандартным видам ресурсов Windows-программ не относится** всё чего нет в скобках ( Menu, Accelerator, Icon, Cursor, Bitmap, String Table, Dialog Box, Font, Version Info). Bitmap Table

**71. К типам оконных сообщений Windows не относится** относится всё чего нет в скобках (WM\_ACTIVATE, WM\_ACTIVATEAPP, WMJDREATE, WM\_DESTROY, WM\_ENABLE, WM\_KILLFOCUS, WM\_MOUSEACTIVATE, WM\_MOVE, WM\_SETFOCUS, WM\_S1ZE). WM\_CLICK

**72. Загрузка ресурса из внешнего исполняемого файла для Windows-приложений выполняется с помощью функции** LoadResource

**73. Тип HINSTANCE представляет** Дескриптор экземпляра приложения. От фразы handle instance. Это Windows-тип который означает экземпляр приложения. Первый параметр при этом содержит указатель (в смысле handle) на текущий экземпляр приложения, второй - всегда равен NULL.

**74. Тип HWND представляет** Дескриптор окна. От фразы handle window. HWnd определяет тип описателя для описателей окна.

**75. К состояниям процесса не относится** всё чего нет в скобках (активное, готовое, заблокированное).

**76. Каждый процесс имеет как минимум один** Один поток

**77. Мьютекс — это объект синхронизации** который устанавливается в особое сигнальное состояние, когда не занят каким-либо потоком. Используемый для обеспечения монопольного доступа к некоторому ресурсу со стороны нескольких потоков

**78. Переключение контекста происходит при** истечение кванта времени, который был выделен потоку, когда поток с более высоким приоритетом стал готовым исполнить код, когда запущенный поток должен ждать.

**79. Событие — это объект синхронизации** который позволяет одному потоку информировать другой об окончании какой-либо операции, наступлении чего-либо

**80. Состояние взаимоблокировки возникает** при борьбе за невыгружаемый ресурсы

**81. Состояние гонки возникает** когда два или более потоков могут получить доступ к общим данным, и они пытаются изменить их одновременно.

**82. Критическая секция — это объект синхронизации** это объект синхронизации, который представляет собой участок кода, в котором поток (thread) получает доступ к ресурсу (например переменная), который доступен из других потоков, позволяющий предотвратить одновременное выполнение некоторого набора операций несколькими потоками

**83. Семафор — это объект синхронизации** который используется для контроля доступа нескольких потоков до общего ресурса. В общем случае это какая-то переменная, состояние которой изменяется каждым из потоков. в основе которого лежит счётчик, над которым можно производить две атомарные операции: увеличение и уменьшение значения на единицу, при этом операция уменьшения для нулевого значения счётчика является блокирующейся.

**84. Процесс в WinAPI создается функцией** CreateProcess

**85. Разделение памяти в ОС Windows позволяет** Техника разделяемой памяти позволяет осуществлять обмен информацией через общий для процессов сегмент памяти без использования системных вызовов ядра. Сегмент разделяемой памяти подключается в свободную часть виртуального адресного пространства процесса. Таким образом, два разных процесса могут иметь разные адреса одной и той же ячейки, подключенной разделяемой памяти.

**86. Технология processor affinity позволяет** Осуществлять привязку вычислительных процессов к ядрам процессора

**87. При использовании статической библиотеки** статические библиотеки представляют собой архивы, которые содержат объектный код для библиотеки, при подключении к приложению, который компилируется в исполняемый. Программа компилируется 1 раз и может запускаться на любом пк без доп библиотек

**88. Функция входа в DLL — это функция** это функция может использоваться для выполнения задач простых инициализации и очистки. DLLMain – работает как switch, в зависимости от передаваемого параметра выполняют ту или иную ветку очистки или инициализации.

**89. Для указания экспортируемых из DLL функций применяют конструкцию** \_\_declspec(dllexport)

**90. При загрузке DLL c явной компоновкой** Явную загрузку динамических библиотек осуществляет функция HINSTANCE LoadLibrary(LPCTSTR lpLibFileName) или ее расширенный аналог HINSTANCE LoadLibraryEx(LPCTSTR lpLibFileName, HANDLE hFile, DWORD dwFlags). Обе они экспортируются из KERNEL32.DLL, следовательно, каждое приложение требует неявной компоновки по крайней мере этой библиотеки. В случае успешной загрузки DLL возвращается линейный адрес библиотеки в памяти. Приложения должны вызывать функцию для явной загрузки библиотеки DLL во время выполнения

**91. При отключении DLL от адресного пространства процесса вызывается ее функция DllMain** **со значением параметра fdwReason равным** DLL\_PROCESS\_DETACH – отключение

**92. Для указания импортируемых из DLL функций применяют конструкцию** \_\_declspec(dllimport) \_\_declspec(dllimport) void func1(void);

**93. Динамическая библиотека содержит** файл, содержащий машинный код. Загружается в память процесса загрузчиком программ операционной системы либо при создании процесса, либо по запросу уже работающего процесса, то есть динамически.

**94. Отличительный признак динамической библиотеки** - позволяет многократное использование различными программными приложениями(процессами). Общие библиотеки отличаются тем, что они не скомпилированы в исполняемый файл. Вместо этого динамический компоновщик ищет в некоторых каталогах нужные библиотеки, а затем загружает их в память.

**95. Расширение динамических библиотек в Windows** .dll (сокр. от «dynamic link library» = «библиотека динамической компоновки»)

**96. Для загрузки DLL-библиотеки в память не применяют функцию GetProcAddress – применяют LoadLibraryQ или LoadLibraryExQ.**

**97. Расширение статических библиотек в Windows** .lib (сокр. от «library»)

**98. Изменение динамической библиотеки, влияет на программы которые ее используют?** если нужно обновить код, который вынесен в динамическую библиотеку, то достаточно обновить только библиотеку, и все программы, что её используют, получат новую версию, их не надо пересобирать. Влияет на все работающие

**99. Код статической библиотеки** Статическая библиотека — объектный файл в виде файла (нередко может быть поставлен вместе с исходным кодом), код из которого выборочно или полностью вставляется в программу на этапе компоновки. Библиотеки, распространяемые в виде исходного кода, преобразуются транслятором в объектные файлы. Затем компоновщик соединяет объектный из объектных файлов библиотек и объектные файлы вашей программы в один исполняемый файл. Включается в исполняемый

**100. При создании нового потока в процессе, система для всех спроецированных в адресное** **пространство DLL вызывает функцию DllMain с параметром fdwReason равным** DLL\_THREAD\_ATTACH

**101. При загрузке DLL c неявной компоновкой** при неявной компоновке функции загружаемой DLL добавляются в секцию импорта вызывающего файла. При запуске такого файла загрузчик операционной системы анализирует секцию импорта и подключает все указанные библиотеки

**102. При нормальном завершении потока в процессе, система для всех спроецированных в** **адресное пространство DLL вызывает функцию DllMain с параметром fdwReason равным** DLL\_THREAD\_DETACH

**103. При проецировании DLL в адресное пространство процесса первый раз система вызывает DllMain со значением параметра fdwReason равным** DLL\_PROCESS\_ATTACH

**104. Процесс может использовать функции динамической библиотеки** если она загружена в память

**105. Расширение статических библиотек в Linux** .a (Archive)

**106. Расширение динамических библиотек в Linux** .so (Shared Object)

**107. Уровень IRQL с самым низким приоритетом – это** PASSIVE LEVEL

**108. Прерывания, поступающие от источников с IRQL, равными или ниже текущего уровня…** маскируются до тех пор, пока выполняемый поток не понизит IRQL.

**109. Для выполнения APC пользовательского режима в контексте целевого потока** Нужно разрешение со стороны этого потока

**110. Cпециальная процедура, вызываемая по прерыванию для выполнения его обработки** Обработчик Прерываний. ISR

**111. Для выполнения APC режима ядра в контексте целевого потока** не нужно «разрешение» со стороны этого потока

**112. Прерывания, поступающие от источника с IRQL, превышающим текущий уровень…** то обработка с более низким приоритетом приостанавливается. Прерывают работу процессора

**113. Этот уровень IRQL используется при взаимодействии процессоров и запроса на** **выполнение какой-то операции** IPI LEVEL

**114. Уровень IRQL с самым высоким приоритетом – это** HIGH LEVEL

**115. DIRQL — это** это уровни IRQL выше dispatch\_level, имеют ли они конкретные мнемонические имена или нет, отражают приоритеты аппаратных прерываний. Таким образом, эти аппаратные IRQL часто упоминаются как уровни IRQL Устройства (или DIRQL).

**116. К возникновению исключения не приводит** подключение флешки либо всё чего нет в скобках (Нарушение защиты памяти, Целочисленное деление на ноль, Целочисленное переполнение, Переполнение/потеря значимости числа с плавающей точкой, Деление на ноль в арифметике с плавающей точкой, Ненормализованный операнд с плавающей точкой, Отладочная точка останова, Неверное выравнивание данных,Недопустимая машинная команда, Привилегированная машинная команда,Отладочное пошаговое исполнение, Нарушение сторожевой страницы,Ошибка чтения страницы, Исчерпание квоты файла подкачки ).

**117. В ОС семества Windows NT х64 и IA64 количество уровней прерываний равно** 16 (0-15)

**118. В ОС семества Windows NT x86 количество уровней прерываний равно** 32 (0-31)

**119. Для поиска обработчика прерываний в Windows используется таблица** IDT

**120. При DPC-вызове передача управления вызываемой процедуре** не происходит – вместо этого адрес вызываемой процедуры и параметры помещаются в специальную очередь

**121. IRQL — это** это механизм программно-аппаратной приоритизации, применяемый для синхронизации в операционных системах семейства Windows NT.

**122. Фильтрующие драйверы нужны для** Драйвер — это набор функций, которые вызываются операционной системой при наступлении некоторых событий, приходящих от устройства или пользовательского режима. Фильтрующие драйверы — драйверы, которые используются для мониторинга или изменения логики другого драйвера путем изменения данных, которые идут к нему. Существует два типа фильтрующих драйверов: 1) верхние фильтрующие драйверы 2) нижние фильтрующие драйверы

**123. DPC-точка входа драйвера нужна для** точки входа вызовов отложенных процедур (DPC - Deferred Procedure Call). Два типа DPC: DpcForlsr и CustomDpc. Драйвер использует эти точки входа, чтобы завершить работу, которая должна быть сделана в результате появления прерывания или другого специального условия. для завершения работы, которое должно быть сделано в результате появления прерывания или другого специального условия.

**124. DriverEntry — это** диспетчер Ввода/вывода вызывает эту функцию драйвера при первоначальной загрузке драйвера. Внутри этой функции драйверы выполняют инициализацию как для себя, так и для любых устройств, которыми они управляют. Эта точка входа требуется для всех NT драйверов.Процедура инициализации виндовс-драйвера

**125. DriverEntry вызывается на IRQL равном** PASSIVE\_LEVEL – код 0

**126. WDM — это** это Windows Driver Model, разработана с учетом технологии Plug and Play и ACPI. Дает возможность загружать драйверы налету, без необходимости перезагрузки системы, подключать их в виде фильтров к другим драйверам, гибкое управление энергосбережением и конфигурации устройств.

**127. Драйвер NT располагается в файле с расширением** .sys и имеет стандартный РЕ-формат (РЕ - Portable Executable).

**128. После удаления последнего объекта устройства и отсутствии ссылок на драйвер диспетчер ввода-вывода производит** выгрузка драйвера (Драйвер освобождает захваченные ресурсы — память, файлы, устройства и т. п.)

**129. Какая процедура драйвера выполняет основную часть обработки прерывания, оставшейся после вызова ISR-процедуры** Процедура отложенного вызова

**130. Драйверы, управляющие устройствами конкретного типа** Функциональный драйвер

**131. ISR-точка входа драйвера нужна для** для поддержки обработки прерываний. (Эта точка входа присутствует, только если драйвер поддерживает обработку прерывания. Как только драйвер подключен к прерываниям от своего устройства, его ISR будет вызываться всякий раз, когда одно из его устройств запрашивает аппаратное прерывание)

**132. С какой компонентой исполнительной системы Windows чаще всего взаимодействует** **драйвер** Диспетчер ввода/вывода

**133. Драйверы PnP должны реализовывать эту процедуру** Процедура добавления устройства (PnP).

**134. Что из перечисленного не является этапом трансляции программы** Трансляция программы производится в несколько этапов. 1. Лексический анализ (выделение лексем) 2. Синтаксический анализ. 3. Генерация кода. скорее всего ответ - морфлогический анализ

**135. Что такое препроцессор** это транслятор, на вход которого поступает исходная программа, содержащая макроопределения и макрокоманды, результирующей программой является макрорасширение на исходном языке. Без создания исполняемого файла или выполнения программы

**136. К общим синтаксическим критериям не относится** Правильный ответ: Малое потребление ресурсов Неправильные ответы: Общ. Синт. Критерии: 1) легкость чтения, 2) легкость написания, 3) легкость верификации, 4) легкость трансляции 5) отсутствие неоднозначности

**137. Для того, чтобы выполнить программу, необходимо** Интерпретатор – это программа, которая воспринимает исходную программу на входном (исходном) языке и выполняет её. Разместить ее в памяти компьютера

**138. Программа, которая собирает исполняемый файл из нескольких объектных модулей,** **называется** Компоновщик (редактор связей, сборщик, линкер)

**139. Исходный код это** это текст компьютерной программы на каком-либо языке программирования или языке разметки, который может быть прочтён человеком. Исходный код – это некоторая последовательность предложений входного языка. программа, написанная на языке программирования, в текстовом формате. А также текстовый файл, содержащий исходный код.

**140. Какое утверждение является верным (вопрос об отношении между компилятором,** **транслятором и интерпретатором)** Возможные варианты ответа: 1. Транслятор – это программа, принимающая на вход программу на одном языке и преобразующая её в программу на другом языке 2. (+) Компилятор – это частный случай (разновидность) трансляторов. А именно, такой транслятор, который преобразует программу на некотором языке программирования в машинный код 3. Интерпретатор – это программа, "просматривающая" код на некотором языке программирования и "выполняющая" одну его инструкцию за другой.

**141. Интерпретатор это** это программа, которая воспринимает исходную программу на входном (исходном) языке и выполняет её.

**142. На последнем этапе трансляции исходный код превращается в** - исполняемый файл .exe, выполняемый код

**143. Какое утверждение является неверным (вопрос об отношении между компилятором,** **транслятором и интерпретатором)** смотреть вопрос 140 возможный вариант ответа - Интерпритатор – частный случай компилятора

**144. g++ это** Компилятор

**145. Компиляция это** это трансляция программы, составленной на исходном языке высокого уровня, в эквивалентную программу на низкоуровневом языке, близком машинному коду.

**146. Программа, позволяющая управлять процессом выполнения программы и обнаруживать в** **ней ошибки** Отладчик

**147. Ассемблер - это частный случай** компилятора

**148. Необязательным этапом трансляции программы является** оптимизация

**149. В формальных языках лексема это -** элементарная составляющая программы (идентификатор, разделитель, символ операций, число, ключевое слово, необязательное слово, пробел, комментарий и т. д.) или, иначе говоря, программная единица, полученная в результате лексического анализа. Или конструкция, которая состоит из элементов алфавита языка и не содержит в себе других конструкций.

**150. Что такое грамматика** Это математическая система, определяющая язык посредством порождающих правил. Это описание способа построения предложений некоторого языка. Или это способ описания формального языка, т. е. выделения некоторого подмножества из множества всех слов некоторого конечного алфавита. Или любой способ конечного задания языка. Наука, изучающая способ построения языка.

**151. В формальных языках и грамматиках цепочка ω называется подцепочкой γ, если γ=** если γ= αωβ, где α, ω, β - это цепочки (Пример: цепочка абвг. Подцепочка может быть а, аб, абв, б, бв и т.д.)

**152. Слова в формальных языках – это** это конечная последовательность символов некоторого алфавита. Все последовательности терминалов, выводимые (порождаемые) из начального нетерминала по правилам вывода, заданных грамматикой.

**153. Язык L над алфавитом А – это** это произвольное множеств цепочек, составленных из символов алфавита A или язык над любым алфавитом содержащим A. Подмножество цепочек А\*

**154. В формальных языках синтаксис это -** это набор правил, определяющий допустимые конструкции языка (правила образования структур из слов и словосочетаний).

**155. В формальных языках лексика это -** это совокупность слов (словарный запас) языка.

**156. Все существующие формальные языки строятся на основе** конечных алфавитов

**157. Символ (буква) в формальных языках – это** элемент конечного непустого множества, из которого состоит алфавит (или это символы из ASCII) Простой неделимый знак

**158. Что можно описать в виде строгого набора правил?** Только формальные языки

**159. Алфавит в формальных языках – это** это счётное множество допустимых символов языка или конечное непустое множество символов.

**160. Цепочка символов (строка) – это** это произвольная упорядоченная конечная последовательность символов некоторого алфавита, записанных один за другим.

**161. В формальных языках семантика это -** это раздел языка, определяющий значение предложений языка. Семантика определяет “содержание языка” – задает смысл для всех допустимых цепочек языка.

**162. Для задания ЯП необходимо** 1) Определить множество допустимых символов языка 2) Определить множество правильных программ языка 3) Задать смысл для каждой правильной программы

**163. Формальная грамматика – это** это описание способа построения предложений некоторого языка или это математическая система, определяющая язык. Способ описания формального языка, выделение некоторого подмножества из множества всех слов некоторого конечного алфавита.

**164. Форма Бэкуса-Наура используется для** используется для описания контекста-свободных формальных грамматик, а именно: 1) Символ “::=” 2) Нетерминалы “<>” 3) Терминалы – это символы используемые в описываемом языке 4) Порождение нескольких альтернативных цепочек “|”.

**165. Конфигурация распознавателя называется конечной, если** если управляющее устройство находится в одном из заранее выделенных заключительных состояний, а входная головка сошла с правого конца входной ленты. Содержимое вспомогательной памяти удовлетворяет некоторому установленному условию.

**166. По типу считывающего устройства распознаватели разделяются на** 1) Односторонние – это значит, что считывающая головка допускает перемещение по ленте входных символов только в одном направлении. 2) Двусторонние – это значит, что считывающая головка может перемещаться относительно ленты входных символов в обоих направлениях.

**167. Язык распознавателя это** – это множество всех цепочек, которые допускает распознаватель.

**168. К структурным компонентам распознавателя не относится** Всё то, чего нет в скобках (входная лента и положение входной головки, устройство управления, внешней памяти) Конфигурации

**169. Выберите действие, которое не выполняется распознавателем** Неправильные варианты: 1) чтение очередного символа из входной цепочки 2) сдвиг входной цепочки на заданное количество символов (вправо или влево) 3) доступ к рабочей памяти для чтения или записи информации 4) преобразование информации в памяти УУ(устройство управления), изменение состояния УУ Правильный ответ: Удаление данных с ленты

**170. К метасимволам языка РБНФ не относится** Правильный ответ:// Неправильные варианты:

1) = или ::= 2) . 3) | 4) { } 5) [ ] 6) ( | )

**171. Распознаватель – это** – это специальный автомат, который позволяет определить принадлежность цепочки символов некоторому языку. Задача распознавателя, чтобы на основании исходной цепочки дать ответ на вопрос, принадлежит ли она заданному языку или нет.

**172. Управляющее устройство называется детерминированным, если** если для каждой допустимой конфигурации распознавателя, которой возникла на некотором шаге его работы, существует единственно возможная конфигурация, в которую распознаватель перейдёт на следующем шаге работы. Для каждой конфигурации распознавателя существует не более одного возможного следующего шага.

**173. Конфигурация распознавателя называется начальной, если** управляющее устройство находится в заданном начальном состоянии, входная головка обозревает самый левый символ на входной ленте и вспомогательная память имеет заранее установленное начальное состояние

**174. Формальная грамматика определяется с помощью** 1) Множества терминальных символов грамматики 2) Конечное множество нетерминальных символов грамматики 3) Множество правил вывода грамматики, являющиеся конечным подмножеством множества (ΣT∪ΣN) в степени (+) умноженное на (ΣT∪ΣN) в степени (\*). 4) Начальный символ грамматики.

**175. К способам задания языка не относится** Точно не будут такие варианты: При использовании нотации Бэкуса-Наура нетерминальные символы (подлежащие дальнейшему определению) языка заключаются в угловые скобки вида < ... > . Последовательность символов : : = означает «определяется как», а символ | означает "или". Пример описания в нотации Бэкуса-Наура арифметических выражений, содержащих переменные a, b,приведен ниже. < выражение > : : = < терм > | < терм > + < выражение > | < терм > - < выражение > < терм > : : = < множитель > | < множитель > \* < терм > | < терм > / < множитель > < множитель > : : = a|b| ( < выражение > )

**176. Деревом вывода грамматики G(T,N,P,S) называется дерево (граф), которое соответствует** **некоторой цепочке вывода и удовлетворяет следующим условиям:** Все перечисленное:

**177. При восходящем разборе дерево строится** дерево строится от терминальных вершин вверх к корню дерева

**178. При нисходящем разборе дерево строится** дерево строится от корня (начального символа) вниз к висячим вершинам

**179. Сентенциальной формой грамматики называется** называется цепочка, выводимая из ее начального символа

**180. Непосредственная выводимость означает** что в множестве правил вывода грамматики содержится правило и, следовательно, в системе команд конечного автомата есть команда

**181. Грамматики по Хомскому типа 0 это -** неограниченные — любая цепочка символов из алфавита. – грамматики с фразовой структурой

**182. Определите тип языка, заданного правилами вывода: 1) S -> 1B; 2) B -> B0 | 1.** Грамматика возможно правая регулярная, значит тип языка 3 (регулярный язык)

**183. Грамматики по Хомскому типа 2 это -** контекстно-свободные

**184. Определите тип языка, заданного правилами вывода: 1) S -> aSBC | abC; 2) CB -> BC; 3) bB -> bb; 4) bC -> bc; 5) cC -> cc.** Контекстно-зависимый язык. Тип 1

**185. Языки классифицируются в соответствии с(со)** с типами грамматик, которыми они задаются

**186. Регулярные грамматики включают в себя** Возможный ответ: простейшие конструкции языков программирования: идентификаторы, константы, строки, комментарии и т. д. Возможный ответ: леволинейные (левые) и праволинейные (правые)классы грамматик

**187. Определите тип языка, заданного правилами вывода: 1) S -> APA; 2) P -> + | -; 3) A -> a | b.** Тип 2

**188. Языки типа "3" по классификации Хомского это -** регулярные языки

**189. Языки типа "1" по классификации Хомского это -** – контекстно-зависимые языки

**190. Грамматики по Хомскому типа 3 это -** регулярные (автоматные) грамматики

**191. В случае, когда язык может быть задан с помощью большого количества грамматик,** **относящихся к разным типам, то из всех этих грамматик берется** та, которая имеет максимальный номер класса

**192. Определите тип языка, заданного правилами вывода: 1) S -> aaCFD; 2) AD -> D; 3) F -> AFB| AB; 4) Cb -> bC; 5) AB -> bBA; 6) CB -> C; 7) Ab -> bA; 8) bCD -> e, где e - цепочка нулевой длины** Тип 0

**193. Определите тип языка, заданного правилами вывода: 1) S -> At | Bt; 2) A -> a | Ba; 3) B -> b | Bb | Ab.** Тип 3

**194. Языки типа "0" по классификации Хомского это -** языки с фразовой структурой

**195. Языки типа "2" по классификации Хомского это -** контекстно-свободные языки

**196. Грамматики по Хомскому типа 1 это -** - контекстно-зависимая грамматика

**197. Грамматики, определяющие один и тот же язык, называются…** эквивалентные

**198. Грамматики, для которых определяемые ими языки различаются не более чем на пустую цепочку символов, называются…** почти эквивалентные

**199. Cтруктурная единица языка, которая состоит из элементарных символов языка это** лексема

**200. Распознавание и выделение лексем из входной последовательности символов производится** лексическим анализатором

**201. Лексический анализ** это начальный этап трансляции, который состоит в выделении в исходной программе элементарных составляющих: идентификаторов, разделителей, комментариев и т.д. Является одним из первых этапов трансляции

**202. Причиной, по которой в состав практически всех компиляторов включают лексический** **анализ, является** Правильные варианты: Все перечисленные 1) сокращение объема информации, обрабатываемой на этапе синтаксического разбора 2) некоторые задачи, требующие использования сложных вычислительных методов на этапе синтаксического анализа, могут быть решены более простыми методами на этапе лексического анализа 3) лексический анализатор отделяет сложный по конструкции синтаксический анализатор от работы непосредственно с текстом исходной программы, структура которого может варьироваться в зависимости от архитектуры вычислительной системы, где выполняется компиляция, - при такой конструкции компилятора для перехода на другую вычислительную систему достаточно только перестроить относительно простой лексический анализатор 4) в современных системах программирования лексический анализатор может выполнять обработку текста исходной программы параллельно с его подготовкой пользователем - это даёт системе программирования принципиально новые возможности, которые позволяют снизить трудоёмкость разработки программ

**203. В ЯП Java слово "if" является** является основным оператором выбора в Java и позволяет выборочно изменять ход выполнения программы - и это одно из основных отличий между программированием и простым вычислением.

**204. В ЯП Java слово "goto" является** Список ключевых слов в Java включает слово goto. Однако данный оператор помечен как not used (не используется). Дело в том, что Джеймс Гослинг, создатель ЯП Java изначально заложил в JVM поддержку оператора goto. Однако впоследствии эту фичу выпилили.

**205. Функции, выполняемые лексическим анализатором и тип лексем, которые он должен выделять, определяются** изменчивостью в зависимости от версии компилятора. То, какие функции будут выполняться лексическим анализатором, а какие будут оставлены для синтаксического разбора - решают разработчики компилятора

**206. Список лексем содержит** весь текст исходной программы, обработанный лексическим анализатором. Содержит все возможные типы лексем, любая лексема может встречаться несколько раз. Каждой лексеме в таблице лексем соответствует некий уникальный условный код, зависящий от типа лексемы, и дополнительная служебная информация. Таблица лексем в каждой строке должна содержать информацию о виде лексемы, ее типе и, возможно, значении. Обычно структуры данных, служащие для организации такой таблицы, имеют два поля: первое – тип лексемы, второе – указатель на информацию о лексеме.

**207. Если измениться кодировка исходной программы, то это повлияет** сканер (лексический анализатор) отделяет сложный по конструкции синтаксический анализатор от работы непосредственно с текстом исходной программы, структура которого может варьироваться в зависимости от версии входного языка — при такой конструкции компилятора для перехода от одной версии языка к другой дос­таточно только перестроить относительно простой лексический анализатор.

**208. Какая из таблиц заполняется при лексическом анализе** Относительная таблица адресов

**209. Выберите таблицы, используемые на этапе лексического анализа** Ключевых слов, идентификаторов, разделителей, констант, терминальных символов, имен, кодов лексем

**210. К типам лексем ЯП не относится** = существуют шесть классов лексем: идентификаторы, ключевые слова, константы, стринговые литералы (в доке Крощенко нет этого), операторы (знаки операций) и разделители.

**211. Теоретической моделью, используемой при построении лексического анализатора,** является конечный автомат

**212. Синтаксический анализатор – это** это часть компилятора, которая отвечает за выявление и проверку синтаксических конструкций входного языка. Основная часть компилятора на этапе анализа. Без его выполнения работа компилятора бессмысленна

**213. Одним из самых простых методов синтаксического анализа является** метод рекурсивного спуска (Recursive descent parser) — алгоритм нисходящего синтаксического анализа, реализуемый путём взаимного вызова процедур, где каждая процедура соответствует одному из правил контекстно-свободной грамматики или БНФ. Применения правил последовательно, слева-направо поглощают токены, полученные от лексического анализатора. Это один из самых простых алгоритмов синтаксического анализа, подходящий для полностью ручной реализации.

**214. На этапе трансляции синтаксический анализ** следует сразу после лексического анализа

**215. Синтаксис языков программирования описывают с помощью** синтаксис языка можно описать, например, с помощью правил Бэкуса — Наура. Синтаксис программного языка является комплексом правил, которые описывают различные комбинационные варианты алфавитных символов. Они считаются программой или её компонентом с правильной структурой.Контекстно-свободных грамматик

**216. Метод рекурсивного спуска применим к** подклассу КС-грамматик - алгоритм нисходящего синтаксического анализа, реализуемый путём взаимного вызова процедур, где каждая процедура соответствует одному из правил контекстно-свободной грамматики или БНФ

**217. Характерной особенностью автомата с магазинной памятью является** является наличие доп памяти в виде стека

**218. МПА при построении синтаксического анализатора – это** - автомат с магазинной памятью - это распознаватель текста исходной программы, построенный на основе грамматики входного языка. Синтаксические конструкции могут быть описаны с помощью КС (контекстно-свободные)-грамматик и с помощью регулярных грамматик. Распознаватель для КС-языков – это МПА.

**219. Что подается на вход синтаксического анализатора?** подается выход лексического анализатора (список лексем), строка токенов.

**220. Недетерминированный МПА** Конечный автомат называется детерминированным, если из каждой его конфигурации возможно не более одного перехода в следующую конфигурацию. В противном случае он недетерминированный.

**221. Детерминированный МПА** Конечный автомат называется детерминированным, если из каждой его конфигурации возможно не более одного перехода в следующую конфигурацию. В противном случае он недетерминированный.

**222. Установление однозначного соответствия между лексическими единицами и их именами** **в тексте программы – это** идентификация

**223. Кодогенерация** часть процесса компиляции, когда специальная часть компилятора, кодогенератор, конвертирует синтаксически корректную программу в последовательность инструкций, которые могут выполняться на машине. Может быть промежуточным либо финальным этапом трансляции

**224. Для формирования внутреннего представления программы может применяться** Все другие варианты Все внутренние представления программы обычно содержат в себе две принципиально различные вещи – операторы и операнды. Различия между формами внутреннего представления заключаются лишь в том, как операторы и операнды соединяются между собой. Также операторы и операнды должны отличаться друг от друга, если они встречаются в любом порядке. За различение операндов и операторов, как уже было сказано выше, отвечает разработчик компилятора, который руководствуется семантикой входного языка. Известны следующие формы внутреннего представления программ: - связочные списочные структуры, представляющие синтаксические деревья - многоадресный код с явно именуемым результатом (тетрады) - многоадресный код с неявно именуемым результатом (триады) - обратная (постфиксная) польская запись операций(полиз) - Q ассемблерный код или машинные команды В каждом конкретном компиляторе может использоваться одна из этих форм, выбранная разработчиками

**225. Назовите неправильное утверждение** синтаксическое дерево – граф в виде дерева, вершины которого – операнды а листья - операции

**226. Преимущество триад при промежуточном представлении программы перед тетрадами в** том, что что делает их применение удобным а) Занимает меньше памяти, чем тетрады б) Формирует линейную последовательность операция, которая в отличии от синтаксического дерева легче интерпретируется (проще переводится в результирующий код) в) Дает больше возможностей по оптимизации, чем ПОЛИЗ г) Явно отражает связь операций между собой д) Промежуточные результаты триад могут сохраняться в регистрах процессора е) Близко к языку ассемблера (двухстороннее представление)

**227. Что из перечисленного не является промежуточным языком** возможно будет что-то типа «результат работы семантического анализатора»(да), «результат работы лексического анализатора»(нет) С++

**228. ПОЛИЗ — это** это обратная польская запись — форма записи математических и логических выражений, в которой операнды расположены перед знаками операций. Также именуется как обратная бесскобочная запись, постфиксная нотация, бесскобочная символика Лукасевича, польская инверсная запись, ПОЛИЗ. (Польская инверсная запись) Способ бесскобочного представления выражений (не только арифметических), в которых операнды предшествуют операции

**229. Дерево операций** синтаксическое дерево, в котором внутренние узлы – операции, а листья – операнды. Если синтаксическому дереву соответствует последовательность операций, то говорят о дереве операций. Может быть построено на основании дерева синтаксического разбора путем его упрощения. Может быть построено из дерева вывода

**230. Ассемблер – это** это транслятор программы из текста на языке ассемблера, в программу на машинном языке. Как и сам язык, ассемблеры, как правило, специфичны для конкретной архитектуры, операционной системы и варианта синтаксиса языка, поскольку работают с мнемониками машинных инструкций определённого процессора. 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. Является существенно платформо-зависимым: языки ассемблера для различных аппаратных платформ несовместимы, хотя могут быть в целом подобны. Язык низкого уровня

**231. Выберите ПОЛИЗ для выражения (2+5) \* (5-8) / 3**

25+58-\*3/

**232. Выберите ПОЛИЗ для выражения (4-2) \* 4 / 5**

42-4\*5/

**233. К достоинствам триадного представления промежуточного кода не относится** Достоинства: а) Занимает меньше памяти, чем тетрады б) Формирует линейную последовательность операция, которая в отличии от синтаксического дерева легче интерпретируется (проще переводится в результирующий код) в) Дает больше возможностей по оптимизации, чем ПОЛИЗ г) Явно отражает связь операций между собой д) Промежуточные результаты триад могут сохраняться в регистрах процессора е) Близко к языку ассемблера (двухстороннее представление)

**234. К недостаткам тетрадного представления промежуточного кода относится** а) Необходимость большого количества памяти для своего представления б) Не отражает взаимосвязь операций между собой в) Есть сложности с непосредственным преобразованием в машинный код

**235. Промежуточный код** байт-код или байткод (англ. byte-code), иногда также используется термин псевдокод — машинно-независимый код низкого уровня, генерируемый транслятором и исполняемый интерпретатором. Большинство инструкций байткода эквивалентны одной или нескольким командам ассемблера. Трансляция в байткод занимает промежуточное положение между компиляцией в машинный код иинтерпретацией. В этапах трансляции: 3 этап: дерево грамматического разбора подается семантическому анализатору (получаем промежуточный код), 4 этап: оптимизация, 5 этап: промежуточный оптимизированный код попадает к генератору кода (получаем объектный код).

**236. К приемам оптимизации линейных участков программы не относится** вынесение инвариантных вычислений из циклов (относятся: • удаление бесполезных присваиваний • исключение избыточных вычислений (лишних операций) • свертка операций объектного кода • перестановка операций • арифметические преобразования)

**237. К приемам оптимизации циклов не относится** свертка операций объектного кода (относится: - развертка циклов – объединение – разрезание – перестановка - разделение циклов на блоки - разрешение зависимости - вынесение инвариантных вычислений из циклов - замена операций с индуктивными переменными - слияние и развертывание циклов)

**238. Линейный участок программы** это выполняемая по порядку последовательность операций, имеющая один вход и один выход. Чаще всего линейный участок содержит последовательность вычислений, состоящих из арифметических операций и операторов присваивания значений переменным.

**239. К приемам оптимизации не относится** извлечение метода (относится:Принципиально различаются два основных вида оптимизирующих преобразований: - преобразования исходной программы (в форме ее внутреннего представления в компиляторе), не зависящие от результирующего объектного языка - преобразования результирующей объектной программы Оптимизируют: - линейные участки программы - логические выражения - вызовы процедур и функций - другие конструкции входного языка Используют как машинно-зависимые, так и машинно-независимые методы)

**240. Что является критериями оптимизации кода** размер и скорость

**241. Обработка, связанная с переупорядочиванием и изменением операций в компилируемой** программе с целью получения более эффективной результирующей объектной программы оптимизация

**242. Для линейных участков не могут выполняться следующие виды оптимизации** развертка объектного кода (могут: • удаление бесполезных присваиваний;• исключение избыточных вычислений (лишних операций);• свертка операций объектного кода;• перестановка операций;• арифметические преобразования)

**243. Особенностью оптимизации как этапа трансляции является** существует оптимизирующий транслятор, оптимизирующий промежуточный код (удаляются избыточные команды и упрощается реализуемый алгоритм), который далее попадает к генератору кода.

**244. Сравните файлы ls и ln, находящиеся в каталоге /bin, используя абсолютные имена файлов** diff /bin/ls /bin/ln

**245. Вы находитесь в каталоге ~/. Просмотрите первые 10 строк файла /etc/group, используя** относительное имя файла

head ../../etc/group

**246. Вы находитесь в каталоге /var/opt. Просмотрите первые 10 строк файла /home/user/mytxt,** используя относительное имя файла

head ../../home/user/mytxt

**247. Вы находитесь в каталоге /opt/pycharm. Просмотрите первые 10 строк файла ~/mytxt,** используя абсолютное имя файла

head ~/mytxt

**248. Вы находитесь в каталоге ~. Где вы окажетесь после применения команды cd** ../../../../../../../..

Корень /

**249. Вы находитесь в каталоге ~. Где вы окажетесь после применения команды cd** ../././././././vasia/./.././brother

/home/brother ?? /brother

**250. Вы находитесь в каталоге ~. Где вы окажетесь после применения команды cd** 1/2/3/././4/././..

/3

**251. Вы находитесь в каталоге /usr/local/bin. Где вы окажетесь после применения команды cd** /../../../../etc/././..

Корень /

**252. Вы находитесь в каталоге /usr/local/bin. Где вы окажетесь после применения команды cd** /../var/./././

/var

**253. Поменяйте владельца и группу файла ~/mytxt на root**

(sudo) chown root:root ~/mytxt

**254. Установите, используя символы прав доступа следующие права на файл ~/mytxt: для себя** все права, для группы и остальных — никаких

chmod -rwx—— ~/mytxt

**255. Установите при помощи восьмеричных масок следующие права на файл ~/mytxt: для себя** все права, для группы и остальных — никаких

chmod 700 ~/mytxt

**256. Установите при помощи восьмеричных масок следующие права на файл ~/mytxt: для себя** исполнение и запись, для группы никаких, для остальных чтение

chmod 304 ~/mytxt

**257. Просмотрите руководство по команде more**

man more

**258. Создайте жесткую ссылку для файла /home/user1/mytxt с тем же именем, поместив ее в** каталог /home/user1/work

ln /home/user1/mytxt.txt /home/user1/work/mytxt.txt

**259. Создайте символьную ссылку для файла /home/user1/mytxt с тем же именем, поместив ее** **в каталог /home/user1/work**

ln -s /home/user1/mytxt.txt /home/user1/work/mytxt.txt

**260. Получите подробную информацию о файлах в текущем каталоге**

ls

**261. Получите список файлов текущего каталога, отсортированный по размеру**

ls -lS /path/to/folder/

**262. Выведите на консоль имя самого большого (по размеру) файла текущего каталога**

Чтобы найти самые большие файлы в определенном месте, просто добавьте путь к команды find:

find /home/yurii-type f -exec du -Sh {} + | sort -rh | head -n 5

find /home/yurii -type f -printf "%s %p\n" | sort -rn | head -n 5

Вышеупомянутая команда отобразит самый большой файл из каталога /home/yurii.

**263. Найдите все текстовые файлы (с расширением txt) текущего каталога, содержащие в** названии "pro" Чтобы найти все файлы с расширением .txt, находящиеся в текущем каталоге, следует воспользоваться такой командой: $ find . -name “\*.txt” –print

**264. Выведите список пользователей в алфавитном порядке**

less /etc/passwd //вся инфа

awk -F: '{ print $1}' /etc/passwd //только имя

cut -d: -f1 /etc/passwd

sort //сортировка

**265. Получите информацию о текущем пользователе**

Самый простой способ получить текущего пользователя — это выполнить следующую команду: $ whoami Так же, можно воспользоваться переменным окружением и вывести следующий параметр: $ echo $USER Так же, можно получить пользователя по ID: $ id -u -n Так же, можно получить пользователя по PID: $ ps -o user= -p $$ | awk '{print $1}' Можно использовать утилиту «w» и она покажет пользователя:# w Вот еще один пример хорошей утилиты# who. С ключом -m -- выводит информацию только о текущем пользователе. $ who -m

**266. Выведите список файлов текущего каталога в три колонки по 7 строк на странице**

ls - Выводит список файлов в директории.