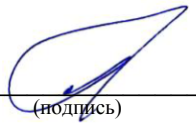
Министерство науки и высшего образования Российской Федерации

ИВАНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Шуйский филиал ИвГУ

Кафедра математики, информатики и методики обучения

ОДОБРЕНО:

Руководитель программы аспирантуры



_____ А.Н. Привалов
(подпись)

«18» марта 2026 г.

Рабочая программа дисциплины
МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, КОМПЛЕКСОВ И КОМПЬЮТЕРНЫХ СЕТЕЙ

Уровень высшего образования:	Подготовка кадров высшей квалификации
Научная специальность:	2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей
Направленность образовательной программы:	
Срок освоения образовательной программы и форма обучения:	3 года

Шуя

1. Цели освоения дисциплины

Целью учебной дисциплины является содействие аспирантам в приобретении знаний, умений и навыков в области математического и программного обеспечения вычислительных машин, комплексов и компьютерных сетей, необходимых в процессе проведения практических работ в рамках диссертационного исследования. В рамках данного курса предполагается изучение современного состояния проблем эффективности и надежности процессов обработки и передачи данных и знаний в вычислительных машинах, комплексах и компьютерных сетях, а также получение преставления о теории программирования, процессах создания и сопровождения программных средств различного назначения.

2. Место дисциплины в структуре ОП

Дисциплина «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей» является обязательной для изучения и относится к дисциплинам образовательной программы, направленным на подготовку к сдаче кандидатского экзамена.

Успешное освоение данной дисциплины будет способствовать готовности аспирантов к подготовке публикаций и (или) заявок на патенты на изобретения, полезные модели, свидетельства о государственной регистрации программ для электронных вычислительных машин, баз данных, топологий интегральных микросхем, подготовке к прохождению промежуточной аттестации по этапам выполнения научного исследования, сдаче кандидатского экзамена и выполнении диссертационного исследования.

Для освоения данной дисциплины обучающийся должен:

Знать:

– основные варианты использования компьютерных технологий в научных исследованиях и образовательных целях;

Уметь:

– эффективно использовать ресурсы Интернета в образовательных целях, по отношению к проблеме собственного диссертационного исследования;

Владеть:

– базовыми теоретическими знаниями для решения профессиональных задач.

3. Планируемые результаты обучения по дисциплине

В результате освоения дисциплины обучающийся должен:

Знать:

– формулировки и доказательства основных положений разделов математики и информатики, соответствующих направлению подготовки;

– постановку проблем математического и информационного моделирования сложных систем;

– основные математические модели, связанные с распределением ресурсов в сложных системах, в телекоммуникации и компьютерных сетях;

– общие закономерности и тенденции развития в области математического и программного обеспечения вычислительных машин, комплексов и компьютерных сетей;

– теоретические основы исследований в области математического и программного обеспечения вычислительных машин, комплексов и компьютерных сетей.

Уметь:

– планировать процесс моделирования и вычислительного эксперимента;

– абстрагироваться от несущественных факторов при моделировании различных объектов и явлений;

– применять математические методы и теории в предметной области, связанной с направлением подготовки;

– использовать специализированные знания в области математического и программного обеспечение вычислительных машин, комплексов и компьютерных сетей для научно-исследовательской работы.

Иметь опыт:

– постановки задач и обработки результатов компьютерного моделирования;

– проектирования и создания баз данных и знаний;

– решения теоретических и прикладных задач в области математического и программного обеспечения вычислительных машин, комплексов и компьютерных сетей.

4. Объем и содержание дисциплины

Объем дисциплины составляет 4 зачетных единицы (144 академических часа)

4.1. Содержание дисциплины по разделам (темам), соотнесенное с видами и трудоемкостью занятий лекционно-семинарского типа

№	Разделы (темы) дисциплины	Семестр	Виды занятий, их объем (в акад. часах, по очной форме обучения)			Формы текущего контроля успеваемости (по очной форме обучения) Формы промежуточной аттестации
			Занятия лекцион-ного типа	Занятия семинар ского типа	Самост оятель ная работа обучаю щихся	
1.	Математические основы программирования	3	2	2 практи ческие	6	Практические задания
2.	Компьютерные сети	3	2	2 практи ческие	6	Практические задания
3.	Языки и системы программирования. Технология разработки программного обеспечения	3	2	2 практи ческие	8	Практические задания
4.	Операционные системы	3	2	2 практи ческие	8	Практические задания
5.	Облачные вычисления	3	2	4 практи ческие	8	Практические задания
6.	Организация баз данных и знаний	3	2	2 практи ческие	8	Практические задания
7.	Искусственные нейронные сети	3	4	4 практи ческие	8	Практические задания
8.	Программная инженерия	3	2	2 практи ческие	8	Практические задания

9.	Заключительный. Подведение и анализ промежуточных результатов освоения дисциплины	3	2	2 практические	8	Практические задания
10.	Итого за семестр:		20	20	68	Кандидатский экзамен

4.2. Развернутое описание содержания дисциплины по разделам (темам)

Тема 1 – Математические основы программирования

Понятие алгоритма и его уточнения. Машины Тьюринга, нормальные алгоритмы Маркова, рекурсивные функции; эквивалентность этих уточнений. Тезис Чёрча-Тьюринга. Понятие алгоритмической неразрешимости; примеры алгоритмически неразрешимых проблем. Понятия недетерминированного, вероятностного и интерактивного алгоритмов.

Понятие сложности алгоритмов. Временная и пространственная сложность алгоритма в худшем случае и в среднем; символы O, Ω, Θ в асимптотических оценках сложности. Сложность в худшем случае сортировки слияниями, быстрой сортировки; сложность в среднем быстрой сортировки. Понятие битовой сложности. Нижние границы сложности класса алгоритмов. Нижние границы сложности классов алгоритмов сортировки и поиска с помощью сравнений. Алгоритм, оптимальный в данном классе, оптимальность по порядку сложности. Понятие сложности вычислительной задачи. Классы P и NP алгоритмов распознавания языков. Сведение по Карпу, по Куку. Полиномиальная сводимость задач; NP-полные задачи (формулировка основных фактов, примеры). Теорема Кука-Левина об NP-полноте задачи выполнимости булевой формулы. Понятие коммуникационная сложности. Понятие алгебраической сложности. Примеры эффективных (полиномиальных) алгоритмов: быстрые алгоритмы поиска и сортировки; полиномиальные алгоритмы для задач на графах и сетях (поиск в глубину и ширину, о минимальном остове, о кратчайшем пути, о назначениях).

Автоматы. Эксперименты с автоматами. Алгебры регулярных выражений. Теорема Клини о регулярных языках. Регулярные языки и их представление конечными автоматами. Задачи распознавания слов конечными автоматами (распознающие автоматы). Конечные автоматы, распознающие структуру дерева (tree finite automata). Альтернирующие конечные автоматы (alternating finite automata).

Тема 2 – Компьютерные сети

Иерархическая организация компьютерных сетей (понятия протокола, иерархии протоколов, сервис и интерфейсы, архитектуры сети), классификация транспортных сред (способы коммутации данных, типы каналов, топология среды), классификация компьютерных сетей (локальные сети, городские сети, региональные сети). Требования, предъявляемые к современным компьютерным сетям (производительность, надежность, безопасность, расширяемость, масштабируемость, управляемость, совместимость).

Способы организации систем передачи данных (на примерах телефонных сетей, стандартов X.25, Frame Relay, SONET, ISDN, ATM, семейство xDSL, GSM сети. GPRS служба), методы уплотнения данных в канале (мультиплексирование), организация и виды коммутаторов. Эталонная модель OSI ISO: понятие открытой системы, основные понятия модели, распределение функций сети между уровнями модели (физический, канальный, сетевой, транспортный, сессии, представления, приложений). Модель TCP/IP.

Теоретические основы передачи данных (данные, сигнал, передача; взаимосвязь пропускной способности канала и ширины его полосы пропускания [теорема Нийквиста-Котельникова, теорема Шеннона]). Основные виды физических сред передачи данных и их свойства (витая пара, коаксиальный кабель, оптоволокно); беспроводные каналы (радио сети, микроволновые сети, сотовые сети, спутниковые сети).

Тема 3 – Языки и системы программирования. Технология разработки программного обеспечения

Языки программирования. Понятие о парадигмах программирования. Процедурные, объектно-ориентированные и функциональные языки программирования. Понятие о логическом программировании. Концепция типа данных. Базисные типы данных в современных языках программирования (булевский, целый, вещественный, символьный, перечислимый, ограниченный, указатели, ссылки, массивы, записи). Динамические структуры данных (списки, бинарные деревья, стек, очередь), основные операции над ними, способы их машинного представления. Таблицы, поиск по ключу. Способы представления таблиц (упорядоченные массивы, деревья поиска, перемешанные таблицы). Оценки сложности (в среднем и в худшем случае) операций поиска по ключу и вставки нового элемента в таблицу. Основные управляющие конструкции в современных языках программирования. Процедуры (функции): вызов процедур, способы передачи параметров, локализация переменных, побочные эффекты. Понятие исключительной ситуации. Обработка исключительных ситуаций. Понятие модуля, раздельная трансляция (зависимая и независимая). Средства инкапсуляции данных. Абстрактные типы данных. Основные принципы объектно-ориентированного программирования. Понятие класса и его реализация в современных языках программирования. Объекты (основные свойства и отличительные признаки). Наследование. Полиморфизм. Динамическое связывание методов. Абстрактные классы и интерфейсы. Понятие о родовых модулях и шаблонах. Обобщенное программирование. Машинно-ориентированные языки, язык ассемблера. Представление машинных команд и констант. Команды транслятору. Их типы, принципы реализации. Макросредства, макровызовы, языки макроопределений, условная макрогенерация, принципы реализации.

Тема 4 – Операционные системы

Основы архитектуры операционных систем. Базовые понятия - процесс, ресурс. Физические, виртуальные (логические) ресурсы. Структура ОС. Ядро. Системные вызовы. Структурная организация ОС. Монолитные, микроядерные ОС. Логические функции ОС. Прерывания, обработка прерываний. Мультипроцессные (мультипрограммные) ОС, требования к аппаратуре компьютера по обеспечению корректного мультипроцессирования. Управление процессами. Управление устройствами. Управление данными. Задачи планирования. Типы ОС: пакетные, разделения времени, реального времени. Сетевые, распределенные ОС. Управление процессами. Определение процесса. Жизненный цикл, состояния процесса. Основные типы процессов: полновесные процессы, легковесные процессы (нити). Контекст процесса. Планирование распределения времени ЦП между процессами -основные подходы: вытесняющие и невытесняющие стратегии; алгоритмы, основанные на квантовании; алгоритмы, использующие приоритет: планирование по наивысшему приоритету, понятие относительного и абсолютного приоритета, алгоритмы, использующие изменяющийся приоритет. Очереди с обратной связью. Смешанные алгоритмы планирования. Взаимодействие процессов. Взаимодействие параллельных процессов и их синхронизация. Классификация средств межпроцессного взаимодействия. Разделяемые ресурсы и синхронизация доступа к ним. Взаимное исключение. Тупики. Механизмы обеспечения взаимного исключения: семафоры Дейкстры, мониторы Хоара, обмен сообщениями. Способы реализации взаимного исключения в однопроцессорных ЭВМ, многопроцессорных ЭВМ с общей памятью, в распределенных вычислительных системах. Классические задачи синхронизации процессов: «обедающие философы», «читатели и писатели», «спящий парикмахер». Типовые средства межпроцессного взаимодействия (базовые средства взаимодействия процессов ОС Unix, IPC ОС Unix, сокеты, MPI). Файловые системы. Файлы, структурная организация файлов. Атрибуты файлов. Типовые программные интерфейсы работы с файлами. Структура файловой системы, подходы в практической реализации. Модели реализации файлов. Понятие индексного узла (дескриптора). Модели реализации каталогов.

Квотирование пространства файловой системы. Надежность файловой системы, целостность файловой системы. Распределенные файловые системы. Алгоритмы обеспечения консистентности кэшей в распределенных файловых системах.

Тема 5 – Облачные вычисления

История основных типов высокопроизводительных вычислений, тенденции развития современных инфраструктурных решений. Виртуализация. Виртуальная машина. Виртуализация серверов. Виртуализация приложений. Виртуализация представлений (рабочих мест). Разновидности архитектуры гипервизора.

Введение в понятия облачных вычислений. Обзор парадигмы облачных вычислений. Архитектура облачных систем. Модели развертывания облаков: частное облако, публичное облако, гибридное облако, общественное облако. Основные модели предоставления услуг облачных вычислений. Различия между облачными и кластерными вычислениями. Достоинства и недостатки облачных вычислений. Обзор существующих сервисов и платформ. Технологии облачных вычислений.

Миграция из стандартной среды в облачные приложения. Концепция миграции. Фазы миграции в облако. Выбор подходящей модели развертывания в соответствии с существующими бизнес-задачами. Выбор подходящего поставщика облачных услуг. Концепция SLA. Производительность облачной инфраструктуры. Концепция вендора. Открытые стандарты для обеспечения облачных услуг.

Тема 6 – Организация баз данных и знаний

Основные понятия структурной части реляционной модели данных. Тип данных, домен, заголовок отношения, кортеж, тело отношения, переменная отношения. Целостная часть реляционной модели данных: возможные и первичный ключи, целостность сущности; понятие внешнего ключа, ссылочная целостность. Манипуляционная часть реляционной модели данных. Реляционная алгебра Кодда: базовые теоретико-множественные и специальные реляционные операции. Общая операция соединения, эквисоединение, естественное соединение. Минимальный набор операций алгебры Кодда. Реляционное исчисление кортежей. Кортежные переменные, правильно построенные формулы, кванторы, свободные и связанные переменные. Выражения исчисления кортежей. Функциональные зависимости. Замыкания функциональных зависимостей, аксиомы Армстронга. Покрытия функциональных зависимостей, Декомпозиция обновлений минимальное покрытие атрибутов отношения. отношения без потерь, теорема Хита. Аномалии и нормализация отношений. Минимальные функциональные зависимости, вторая нормальная форма отношений. Транзитивные функциональные зависимости, третья нормальная форма. Независимые проекции, теорема Риссанена. Атомарные проекции. Перекрывающиеся возможные ключи, нормальная форма Бойса-Кодда. Обобщенная архитектура, компоненты и функции системы управления базой данных (СУБД). Структуры внешней памяти реляционных баз данных, способы организации индексов. Алгоритмы буферизации базы данных в основной памяти. Физическая и логическая целостность баз данных. Понятие транзакции. Способы поддержания логической целостности данных, ограничения целостности и триггеры. Методы восстановления баз данных после сбоев, восстановление физически и логически целостного состояния базы данных. Авторизация доступа к базе данных, механизм привилегий. Понятие ACID-транзакции. Минимальные требования к поддержке атомарности, согласованности, изолированности и сохранности результатов транзакций. Виды конфликтов транзакций. Сериализация транзакций. Методы сериализации транзакций. Двухфазный протокол синхронизационных блокировок. Гранулированные и предикатные блокировки. Сериализация транзакций на основе временных меток. Журнализация изменений базы данных, протокол WAL. Логическая и физическая журнализация. Индивидуальные откаты транзакций. Восстановления базы данных по журналу после мягких и жестких сбоев.

Тема 7 – Искусственные нейронные сети

Формальный нейрон Маккалока-Питтса. Персептрон Розенблатта. Линейная разделимость. Преодоление проблемы линейной разделимости. Обучение с учителем: классификация образов. Обучение с учителем: аппроксимация многомерных функций. Сети прямого распространения. Алгоритм обратного распространения ошибки. Проблемы обучения: ошибка аппроксимации; переобучение; ошибка, связанная со сложностью модели. Оптимизация размера сети. Адаптивная оптимизации архитектуры сети. Валидация обучения. Ранняя остановка обучения. Сети встречного распространения. Обучение без учителя. Структура слоев Кохоненна и Гроссберга. Обучение. Предварительная обработка входных векторов. Выбор начальных значений весовых векторов. Статистические свойства обученной сети.

Тема 8 – Программная инженерия

Основы программной инженерии. Понятие жизненного цикла программного обеспечения (ЖЦ ПО). Стандарт ISO/IEC 12207 (Information Technology - Software Life Cycle Processes). Процессы ЖЦ ПО: основные, вспомогательные и организационные. Понятия модели и стадии ЖЦ ПО. Каскадная и итерационная модели ЖЦ. Методы «быстрой» разработки ПО (Agile Software Development). Понятие зрелости процессов создания ПО (Software Process). Модель оценки зрелости CMM (Capability Maturity Model). Уровни зрелости процессов создания ПО (начальный, воспроизводимый, определенный, управляемый, оптимизируемый). Технологии создания ПО. Понятие технологии создания ПО и ее состав (технологические процессы, технологические операции, рабочие продукты, роли, руководства, инструментальные средства). Требования, предъявляемые к технологии. Факторы оценки и выбора технологии. Технологии создания ПО как программные продукты. Унифицированный процесс разработки ПО (Unified Software Development Process). Метрики ПО. Оценка сложности и размера ПО. Стандартные метрики, применяемые для оценки размера ПО (количество строк кода и функциональных точек (function points)). Методы оценки трудоемкости создания ПО (статистические модели, экспертные оценки, оценки по аналогии). Базовая методика оценки трудоемкости Боэма (COCOMO – Конструктивная модель стоимости). Расчет трудоемкости по методу функциональных точек. Методика выявления функциональных типов и подсчета количества функциональных точек. Качество ПО. Понятие качества ПО. Показатели качества ПО (функциональность, надежность, удобство использования, производительность, адаптируемость). Методы обеспечения качества ПО (отладка, тестирование, верификация и аттестация ПО). Методы, принципы, стратегии и этапы тестирования ПО. Функциональное и нагрузочное тестирование ПО, тестирование пользовательского интерфейса. Модульное, интеграционное и регрессионное тестирование ПО. Организация и планирование тестирования ПО.

Тема 9 – Заключительный. Подведение и анализ промежуточных результатов освоения дисциплины

5. Образовательные технологии

В программе данного курса предусмотрено использование в учебном процессе активных форм проведения занятий и организации самостоятельной работы. В учебном процессе используются как активные, так и интерактивные формы проведения занятий: дискуссия, метод поиска быстрых решений в группе, мозговой штурм.

Аудиторные занятия проводятся в интерактивной форме с использованием мультимедийного обеспечения (ноутбук, проектор) и технологии проблемного обучения.

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине: технологии смешанного обучения, мультимедиа технологии, облачные технологии, технологии визуализации (инфографика, дашборды).

6. Учебно-методическое обеспечение самостоятельной работы обучающихся

Самостоятельная работа обучающихся по дисциплине осуществляется в следующих формах:

−повторение материала с использованием конспектов лекций;

−изучение и анализ предложенных электронных образовательных ресурсов, содержащих информацию по математическому и программному обеспечению вычислительных систем, комплексов и компьютерных сетей;

−выполнение заданий, из которых складывается портфолио студента, включающее обязательные и дополнительные (определяемые студентами совместно с преподавателем) материалы.

Задания и результаты их выполнения составляют часть информационного ресурса кафедры математики, информатики и методики обучения и хранятся на сервере кафедры, доступ к ним обеспечен с компьютеров в компьютерном классе.

Полностью весь методический материал по обеспечению самостоятельной работы студентов приводится в приложении 1 к настоящей программе. Доступ к методическим материалам осуществляется при помощи электронной информационно-образовательной среды Шуйского филиала ИвГУ.

7. Характеристика оценочных средств для текущего контроля успеваемости и промежуточной аттестации обучающихся по дисциплине

Промежуточная аттестация по дисциплине (экзамен) проводится с использованием рейтинговой оценки.

Промежуточная аттестация студентов предполагает экзамен в форме выполнения тестового задания, примерный вариант которого представлен в Приложении 2.

Критерии оценки:

− оценка «отлично» выставляется аспиранту, если аспирант выполнил тестового задание на 85 или более процентов;

−оценка «хорошо» выставляется аспиранту, если аспирант выполнил тестового задание на 71 или более процентов;

−оценка «удовлетворительно» выставляется аспиранту, если аспирант выполнил тестового задание на 61 или более процентов;

−оценка «неудовлетворительно» выставляется аспиранту, если аспирант выполнил тестовое задание на менее чем 61 процент;

8. Учебно-методическое и информационное обеспечение дисциплины

Основная литература:

1. Онтологии и тезаурусы: модели, инструменты, приложения : учебное пособие / Б. В. Добров, В. В. Иванов, Н. В. Лукашевич, В. Д. Соловьев. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ) : Бином. Лаборатория знаний, 2009. – 173 с. : ил.,табл., схем. – (Основы информационных технологий). – ISBN 978-5-9963-0007-5; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=233056

2. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский ; пер. с пол. И. Д. Рудинского. – 2-е изд., стер. – Москва : Горячая линия – Телеком, 2013. – 384 с. : ил. – ISBN 978-5-9912-0320-3; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=253603

3. Городняя, Л. В. Основы функционального программирования: курс : учебное пособие / Л. В. Городняя ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2004. – 217 с.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=233773

4. Мейер, Б. Инструменты, алгоритмы и структуры данных / Б. Мейер. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 543 с. : схем., ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=429033

5. Биллиг, В. А. Параллельные вычисления и многопоточное программирование / В. А. Биллиг. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 311 с. : ил., схем.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=428948

6. Жуматий, С. А. Вычислительное дело и кластерные системы: курс : учебное пособие / С. А. Жуматий, В. В. Воеводин ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2008. – 125 с. : ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=234002

Дополнительная литература:

1. Алдан, А. Введение в генерацию программного кода / А. Алдан. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 189 с. : схем.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=428825

2. Алексеев, В. Е. Графы и алгоритмы: структуры данных. Модели вычислений / В. Е. Алексеев, В. А. Таланов. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 154 с. : ил. – (Основы информационных технологий). – ISBN 5-9556-0066-3; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=428827

3. Афонин, В. Л. Интеллектуальные робототехнические системы: курс лекций / В. Л. Афонин, В. А. Макушкин. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2005. – 208 с. : ил.,табл., схем. – (Основы информационных технологий). – ISBN 5-9556-0024-8; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=232978

4. Гергель, В. П. Технологии построения и использования кластерных систем: курс : учебное пособие / В. П. Гергель ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2009. – 470 с. : ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=233768

5. Зайцева, Е. В. Математическая логика и теория алгоритмов : учебное пособие / Е. В. Зайцева, Л. М. Гурова. – Москва : Московский государственный горный университет, 2006. – 255 с. – (Высшее горное образование). – ISBN 5-7418-0451-9; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=83721

6. Ильясова, Н. Ю. Информационные технологии анализа изображений в задачах медицинской диагностики / Н. Ю. Ильясова, А. В. Куприянов, А. Г. Храмов. – Москва : Издательство Радио и связь, 2012. – 424 с. – ISBN 5-89776-014-4; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=467652

7. Сотник, С. Л. Проектирование систем искусственного интеллекта: курс : учебное пособие / С. Л. Сотник. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2007. – 204 с. : ил., табл., схем.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=234802

8. Павлов, С. И. Системы искусственного интеллекта : учебное пособие / С. И. Павлов. – Томск : Томский государственный университет систем управления и радиоэлектроники, 2011. – Часть 2. – 194 с. – ISBN 978-5-4332-0014-2; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=208939

9. Пентус, А. Е. Математическая теория формальных языков : учебник / А. Е. Пентус, М. Р. Пентус. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ) : Бином. Лаборатория знаний, 2006. – 248 с. – (Основы информатики и математики). – ISBN 5-9556-0062-0; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=233201

10. Шапиро, Л. Компьютерное зрение : учебное пособие / Л. Шапиро, Д. Стокман ; ред. пер. С. М. Соколов ; пер. с англ. А. А. Богуславского. – 4-е изд., электрон. – Москва : Лаборатория знаний, 2020. – 763 с. : ил. – (Лучший зарубежный учебник). – ISBN 978-5-00101-696-0; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=445962

11. Чубукова, И. А. Data Mining : учебное пособие / И. А. Чубукова. – 2-е изд., испр. – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ) : Бином. Лаборатория знаний, 2008. – 383 с. – (Основы информационных технологий). – ISBN 978-5-94774-819-2; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=233055

Ресурсы информационно-телекоммуникационной сети «Интернет»:
Профессиональные базы данных и информационно-справочные системы:
ЭБС «Университетская библиотека онлайн» www.biblioclub.ru
Национальный цифровой ресурс Руконт - https://rucont.ru/default.aspx
Электронная библиотека ИвГУ http://lib.ivanovo.ac.ru
Электронный каталог НБ ИвГУ http://lib.ivanovo.ac.ru/index.php/ek

Программное обеспечение: операционная система Microsoft Windows, пакет офисных программ Microsoft Office и(или) LibreOffice, интернет-браузер Microsoft Edge и(или) Yandex Browser.

9. Материально-техническое обеспечение дисциплины

Учебные аудитории:

– для проведения занятий лекционного типа с комплектом специализированной учебной мебели и техническими средствами обучения, служащими для предоставления учебной информации большой аудитории;

– для проведения занятий семинарского типа, консультаций, текущего контроля и промежуточной аттестации с комплектом специализированной учебной мебели и техническими средствами обучения.

Помещение для самостоятельной работы, оснащенное комплектом специализированной учебной мебели, компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в ЭИОС.

Демонстрационное оборудование и учебно-наглядные пособия для занятий лекционного типа, обеспечивающие тематические иллюстрации: электронные пособия (презентации), аудио-визуальные пособия (видеоматериалы), печатные пособия (таблицы, схемы)).

Автор(ы) рабочей программы дисциплины: профессор кафедры математики, информатики и методики обучения, доктор технических наук, профессор Привалов А. Н.

Программа рассмотрена и утверждена на заседании кафедры математики, информатики и методики обучения «02» марта 2022 г., протокол № 8

Программа обновлена
протокол заседания кафедры № 8 от «22» марта 2023 г.
Согласовано:
Руководитель ОП _____ А.Н. Привалов
(подпись)

Программа обновлена
протокол заседания кафедры № 7 от «9» февраля 2024 г.
Согласовано:
Руководитель ОП _____ А.Н. Привалов
(подпись)

Программа обновлена
протокол заседания кафедры № 9 от «20» марта 2025 г.
Согласовано:
Руководитель ОП _____ А.Н. Привалов
(подпись)

Программа обновлена
протокол заседания кафедры № 9 от «18» марта 2026 г.
Согласовано:
Руководитель ОП _____ А.Н. Привалов
(подпись)

Министерство науки и высшего образования Российской Федерации

ИВАНОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Шуйский филиал ИвГУ

Кафедра математики, информатики и методики обучения

Приложение 1
к рабочей программе дисциплины

Методические указания
для обучающихся по освоению дисциплины
МАТЕМАТИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ, КОМПЛЕКСОВ И КОМПЬЮТЕРНЫХ СЕТЕЙ

Уровень высшего образования:	Подготовка кадров высшей квалификации
Научная специальность:	2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей
Направленность образовательной программы:	
Срок освоения образовательной программы:	3 года

Шуя

1. Методические указания к проработке лекционного материала по дисциплине

Общий план лекций включает следующие составные элементы:

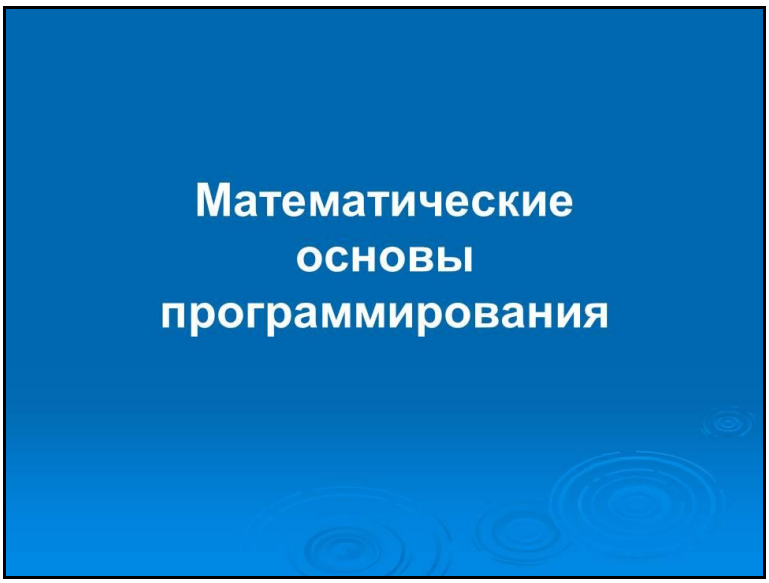
Название этапа	**Содержание**	**Время, мин.**
Вступительная часть	Приветствие. Контроль посещаемости. Постановка темы, цели, основных задач.	4-7
Актуальность и основные понятия	Формулирование проблемы; основные понятия и суть изучаемого вопроса.	8-10
Основная часть	Представление материала лекции в соответствии с программой	60-65
Закрепление полученной информации	Вопрос-ответ, фронтальный опрос, разбор ошибок, допущенных при ответах	5-8
Подведение итогов	Сопоставление результатов лекции с установленной целью и намеченными задачами.	3

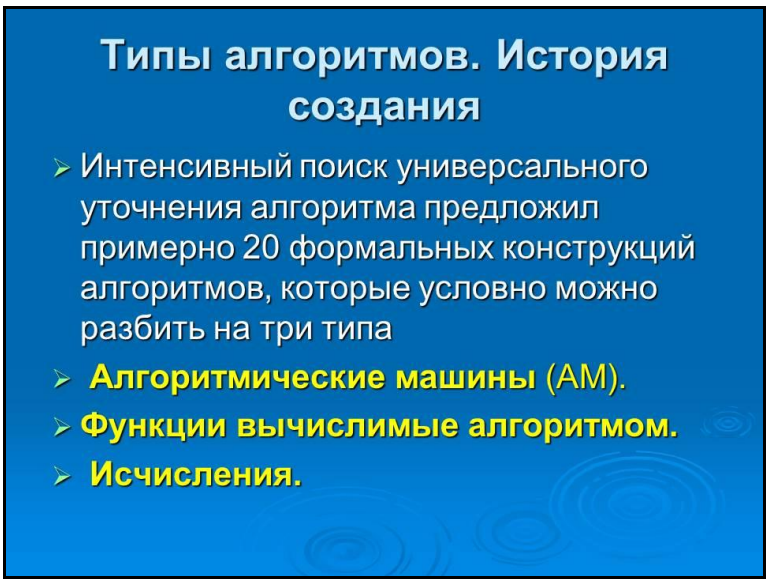
В рамках проведения занятий по дисциплине предполагается чтение следующих лекций:
Лекция 1. Математические основы программирования
Тезисы лекции:

- Понятие алгоритма и его уточнения. Машины Тьюринга, нормальные алгоритмы Маркова, рекурсивные функции; эквивалентность этих уточнений. Тезис Чёрча-Тьюринга. Понятие алгоритмической неразрешимости; примеры алгоритмически неразрешимых проблем. Понятия недетерминированного, вероятностного и интерактивного алгоритмов.
- Понятие сложности алгоритмов. Временная и пространственная сложность алгоритма в худшем случае и в среднем; символы O, Ω, Θ в асимптотических оценках сложности. Сложность в худшем случае сортировки слияниями, быстрой сортировки; сложность в среднем быстрой сортировки.
- Понятие битовой сложности. Нижние границы сложности класса алгоритмов. Нижние границы сложности классов алгоритмов сортировки и поиска с помощью сравнений. Алгоритм, оптимальный в данном классе, оптимальность по порядку сложности.
- Понятие сложности вычислительной задачи. Классы P и NP алгоритмов распознавания языков. Сведение по Карпу, по Куку. Полиномиальная сводимость задач; NP-полные задачи (формулировка основных фактов, примеры). Теорема Кука-Левина об NP-полноте задачи выполнимости булевой формулы. Понятие коммуникационная сложности.
- Понятие алгебраической сложности. Примеры эффективных (полиномиальных) алгоритмов: быстрые алгоритмы поиска и сортировки; полиномиальные алгоритмы для задач на графах и сетях (поиск в глубину и ширину, о минимальном остове, о кратчайшем пути, о назначениях).
- Автоматы. Эксперименты с автоматами. Алгебры регулярных выражений. Теорема Клини о регулярных языках. Регулярные языки и их представление конечными

автоматами. Задачи распознавания слов конечными автоматами (распознающие автоматы). Конечные автоматы, распознающие структуру дерева (tree finite automata). Альтернирующие конечные автоматы (alternating finite automata).

Слайды презентации к лекции:





Алгоритмические машины (АМ)

- имеют единственный процессор, выполняющий небольшой набор очень примитивных действий,
- простую структуру данных (структуру памяти) в виде бесконечной ленты,
- простую логику (правила) управления процессором.

Основные АМ

- Машина Тьюринга (МТ) предложена Тьюрингом в 1937 г.
- Машина Поста (МР) предложена Постом в 1937 г.
- Нормальный алгоритм Маркова (НАМ) предложен Марковым в 1953 г.

Функции вычислимые алгоритмом

- алгоритм не определяется формально, а существует как бы в виде «всем понятной механической процедуры».
- Любая функция, вычислимая на интуитивном (содержательном) уровне, должна быть сконструирована из базовых

- **Рекурсивные функции** на множестве натуральных чисел были предложены Клини в 1938 г.
- Конструктивные механизмы рекурсивных функций очень просты, их применение в процессе построения «функции от функции» позволяет явно выстраивать структуру функции в отличие от АМ, где функция определяется процедурно, через последовательность действий.

Исчисления.

- **Исчисление функций**, вычисляемых на множестве натуральных чисел предложено Эрбраном и Гёделем в 1938 г.
- **λ-исчисление** А.Чёрча также может быть отнесено к этому типу алгоритмов, предложено в 1937 г.
- **Формальные грамматики**, порождающие языки, предложены Хомским в 1953 – 1956 г.

Структура алгоритма (составляющие алгоритма)

- **Процессорная структура**. (Исполнитель алгоритма).
  Во всех теоретических конструкциях алгоритмов и большинстве алгоритмических языках это единственный процессор.
- **Структуры данных**. Структура данных это способ организация записи, хранения и извлечение данных.
- **Данные** – это элементы множеств, которые нужно порождать или распознавать..

- **Информационная структура алгоритма (ИСА)**. Структура функций есть описание конструирования функции от функций из базовых.
- **Логическая структура алгоритма (ЛСА) или программы (ЛСП)**. ЛСА суть описание организации вычислительного процесса, который управляется состоянием памяти.

Интерпретация МТ.

- **Процессор** – в МТ называется управляющей головкой (УГ).
- **Структура данных** (память процессора) бесконечная лента, разбитая на ячейки, в ячейку может быть записан только один символ
- **Процесс вычислений** происходит по тактам
- **Процесс остановки** **(остановка) МТ**.
- ✓ Замечание **Список правил для МТ не упорядочен**



ТЬЮРИНГ Алан Матисон (Turing Alan Mathison)
(1912—1954), английский математик. Основные труды по математической логике, вычислительной математике. В 1936-37 годах ввел математическое понятие абстрактного эквивалента алгоритма, или вычислимой функции, получившее затем название «машины Тьюринга».

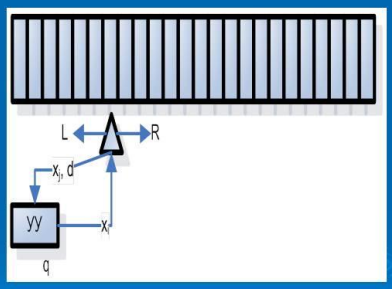
В возрасте 24 лет Тьюринг написал работу "О вычислимых числах", которой суждено было сыграть исключительно важную роль в развитии вычислительной математики и информатики

МТ

- Тьюринг назвал свое абстрактное механическое устройство "универсальной машиной", поскольку она должна была справляться с любой допустимой, то есть теоретически разрешимой задачей — математической или логической.
- Данные должны были вводиться в машину на бумажной ленте, поделенной на клетки — ячейки.
- Каждая такая ячейка либо содержала символ, либо была пустой.
- Машина могла не только обрабатывать записанные на ленте символы, но и изменять их, стирая старые и записывая новые в соответствии с инструкциями, хранимыми в ее внутренней памяти.

Абстрактная модель машины Тьюринга

- МТ = <Q, D, P, q0, qкон>



МТ = <Q, D, P, q0, qкон>

где Г – головка,

УУ – управляющее устройство,

$q \in Q$ – состояние,

L – влево на 1 ячейку, R – вправо на 1 ячейку, S – стоять на месте,

$d \in D$ – команда, $D=\{L,R,S\}$ – множество команд на перемещение головки,

$Q=\{q_0,q_1,\dots q_{кон}\}$ – конечное множество состояний машины Тьюринга,

$\overline{X} = X \cup \{\varepsilon\}$, X – конечное множество входных символов, которые записываются в ячейки ленты,

$P=\{p_{ij}\}$, p_i: $(q, x_i \cup \varepsilon) \rightarrow (q', x_j \cup \varepsilon, d)$

$q_0 \in Q$ – символ начального состояния машины Тьюринга,

$q_{кон} \in Q$ – символ конечного состояния машины Тьюринга.

Вход машины Тьюринга — то, что записано на ленте. За входом следует бесконечное число пробелов.
Выход машины Тьюринга — то, что записано на ленте после того, как машина пришла в конечное состояние.
Если машина Тьюринга проверяет принадлежность языку, то удобно иметь два конечных состояния: q_{yes} и q_{no}.
Машина Тьюринга может:

- закончить работу;
- работать бесконечно.

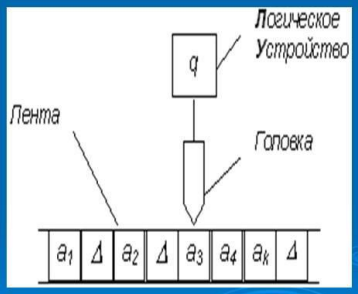
Время

- Временем работы машины Тьюринга на входе x называем количество шагов, которое машина делает, чтобы прийти в конечное состояние.
- Временной сложностью машины Тьюринга называем максимум по всем входам длины n времени работы машины на этих входах.

Память

- Сложностью по памяти работы машины Тьюринга на входе x называем количество ячеек, в которых побывала головка машины.
- Емкостной сложностью машины Тьюринга называем максимум по всем входам длины n сложности по памяти работы машины на этих входах.

- **Машина Тьюринга состоит из трех частей: ленты, считывающе-записывающей головки и логического устройства**



- **Лента выступает в качестве внешней памяти; она считается неограниченной (бесконечной)**

- Головка неподвижна, а лента передвигается относительно нее вправо или влево.
- Машина работает в некотором произвольном конечном алфавите

  A = {, a1…a n} – этот алфавит называется *внешним*.
- В нем выделяется специальный символ – , называемый *пустым знаком* – его посылка в какую-либо ячейку стирает тот знак, который до этого там находился, и оставляет ячейку пустой.

- В каждую ячейку ленты может быть записан лишь один символ.
- Информация, хранящаяся на ленте, изображается *конечной* последовательностью знаков внешнего алфавита, отличных от пустого знака.
- Головка всегда расположена над одной из ячеек ленты.
- Работа происходит тактами (шагами).

- Система исполняемых головкой команд предельно проста:

  на каждом такте она производит замену знака в обозреваемой ячейке *ai* знаком *aj*
- При этом возможны сочетания:
- *j = i* – это означает, что в обозреваемой ячейке знак не изменился;
- *i_0, j = 0* означает, что хранившийся в ячейке знак заменяется пустым, т.е. стирается;
- *i =0, j_ 0* означает, что пустой знак заменяется непустым (с номером j в алфавите), т.е. производится вставка знака;
- *i j_ 0* соответствует замене одного знака другим.

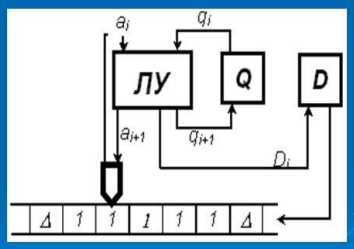
Команды перемещений ленты

- на ячейку влево,
- на ячейку вправо
- остаться на месте, т.е. адрес обозреваемой ячейки в результате выполнения команды может либо измениться на 1, либо остаться неизменным.

✓команда сдвига ленты влево обозначается *R («Right»),*
✓ сдвига вправо – *L («Left»),*
✓ отсутствие сдвига – *S («Stop»).*

Элементарный шаг (такт) работы машины Тьюринга

- головка считывает символ из обозреваемой ячейки и, в зависимости от своего состояния и прочитанного символа, выполняет команду, в которой указано, какой символ записать (или стереть) и какое движение совершить. При этом и головка переходит в новое состояние



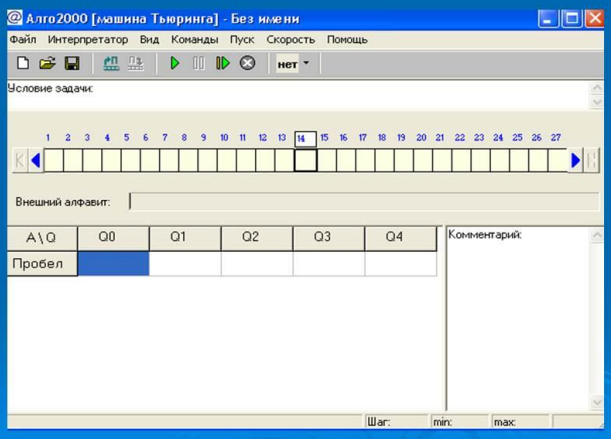
- *Определение Конфигурация машины- совокупность состояний всех ячеек ленты, состояния ЛУ и положение головки*
- В зависимости от начальной конфигурации возможны два варианта :
- после конечного числа тактов машина останавливается по команде остановки; при этом на ленте оказывается конечная конфигурация, соответствующая выходной информации;
- остановки не происходит.
- В первом случае говорят, что данная машина применима к начальной информации, во втором – нет.

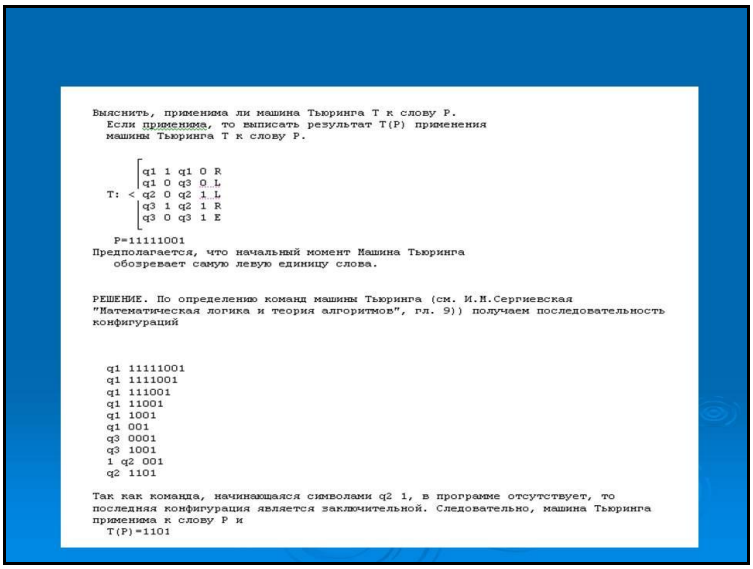
Пример

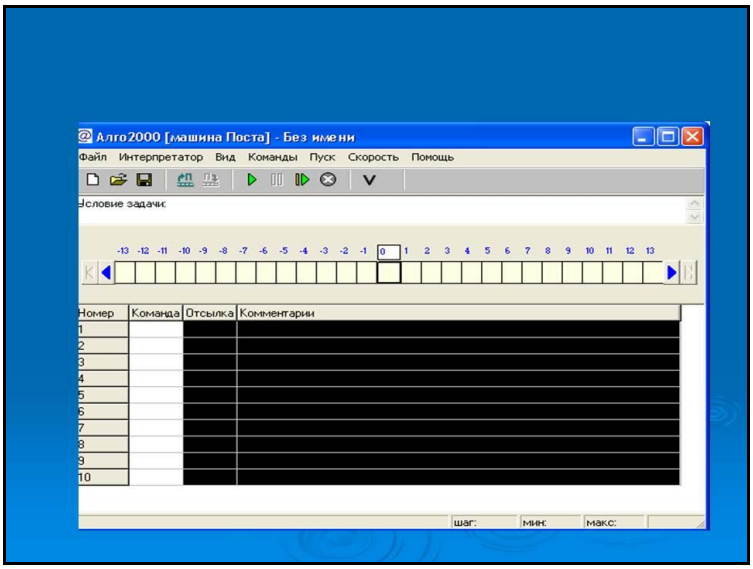
- Пусть начальной является конфигурация *1q1111*.
- Такт 1 Обозревается *1*, в *ЛУ* состояние *q*. Выходная команда *q1R*, что эквивалентно перемещению головки по отношению ленты на 1 шаг вправо. Следовательно, образуется промежуточная конфигурация *11q111*.
- Такт 2 – аналогичным образом осуществляется переход к конфигурации *111q11*.
- Такт 3 – переход к конфигурации *1111q1*.
- Такт 4 –переход к конфигурации *11111q*
- Такт 5 Обозревается , в *ЛУ* состояние *q*. Выходная команда *z1S* – вместо в ячейку записывается *1*, сдвига нет, работа прекращается. Конечная конфигурация *111111z*.
- Задача решена.

Тезис Тьюринга

- ***Всякий алгоритм может быть задан посредством тьюринговой функциональной схемы и реализован в соответствующей машине Тьюринга.***
- ***Машина Тьюринга - это модель компьютера***







Алгоритмическая машина Поста

- Абстрактная машина Поста состоит
- из бесконечной ленты, разделенной на равные секции,
- считывающе-записывающей головки.
- Каждая секция может быть либо пуста (т.е. в нее ничего не записано), либо заполнена (*отмечена* – т.е. в нее записана *метка*).
- Состояние ленты и информация о положении головки характеризуют *состояние машины* Поста.

- За один такт (его называют *шагом*) головка может сдвинуться на одну секцию вправо или влево и поставить или удалить метку.
- Работа машины Поста заключает в переходе от одного состояния машины к другому в соответствии с заданной программой, которая строится из отдельных команд.

Структура команды

- Каждая команда имеет следующую структуру
- *xKy*,
- *x* – номер исполняемой команды;
- *K* – указание о выполняемом действии;
- *y* – номер следующей команды (*наследника*).

Система команд машины

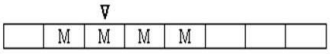
№ п/п	Команда	Запись команды	Описание действий машины
1	Шаг вправо	$X \rightarrow y$	Сдвиг головки на одну секцию вправо
2	Шаг влево	$X \leftarrow y$	Сдвиг головки на одну секцию влево
3	Установить метку	*XMy*	В обозреваемую секцию ставится метка
4	Стереть метку	*XCy*	Из обозреваемой секции удаляется метка

Система команд машины

5	Передача управления	$x \begin{matrix} y_1 \\ y_2 \end{matrix}$	При отсутствии метки в обозреваемой секции управление передается команде y_1, при наличии – команде y_2.
6	Остановка	*x стоп*	Прекращение работы машины

Пример

На ленте записано некоторое число, и головка обозревает одну из помеченных секций (любую). Составить программу прибавления единицы к этому числу. Ситуация иллюстрируется рисунком.

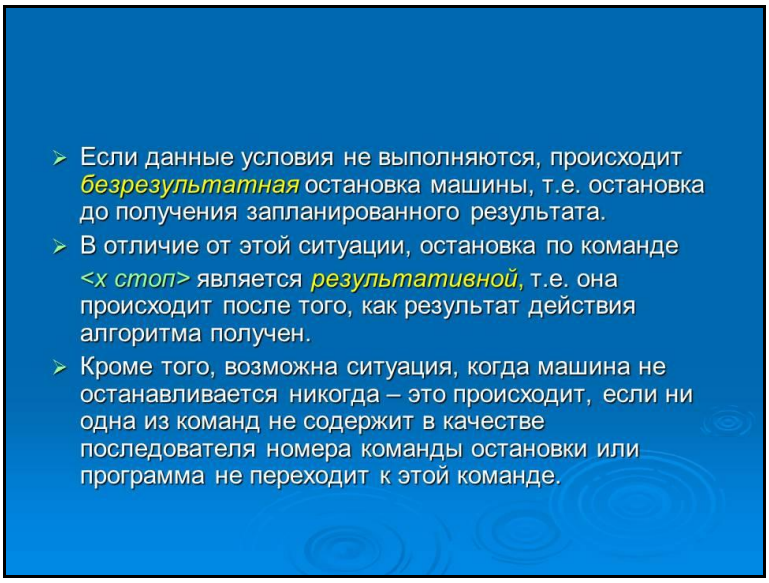


Программа, обеспечивающая решение задачи, состоит из 4-х команд:

1. $\begin{matrix} 3 \\ 2 \end{matrix}$
2. → 1
3. M 4
4. стоп.

Комментарий к примеру

- ✓ Последовательное исполнение команд 1 и 2 приводит к тому, что головка за два такта работы машины сдвигается на одну позицию вправо.
- ✓ Это передвижение продолжается до тех пор, пока после очередного сдвига под головкой не окажется пустой ячейки – тогда по команде 3 в нее будет поставлена метка
- ✓ и по команде 4 машина остановится



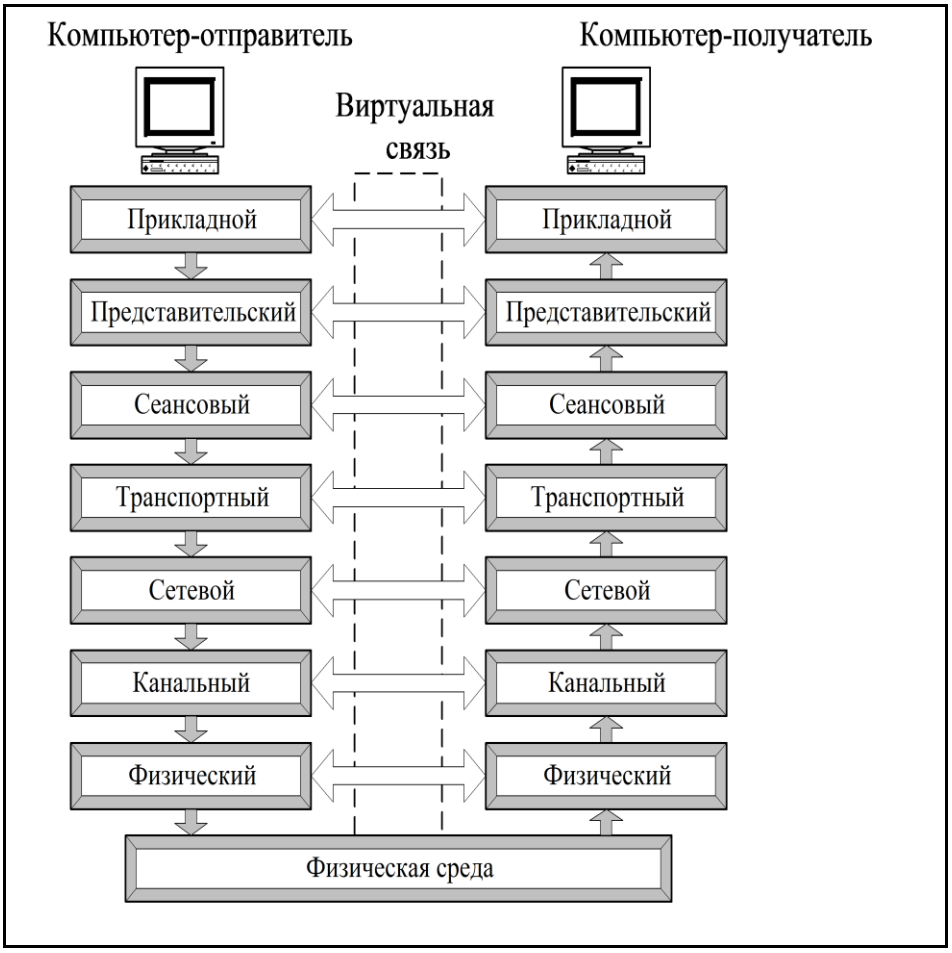
Литература: Мейер, Б. Инструменты, алгоритмы и структуры данных / Б. Мейер. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 543 с. : схем., ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=429033

Лекция 2. Компьютерные сети

Тезисы лекции:

- Иерархическая организация компьютерных сетей (понятия протокола, иерархии протоколов, сервис и интерфейсы, архитектуры сети), классификация транспортных сред (способы коммутации данных, типы каналов, топология среды), классификация компьютерных сетей (локальные сети, городские сети, региональные сети). Требования, предъявляемые к современным компьютерным сетям (производительность, надежность, безопасность, расширяемость, масштабируемость, управляемость, совместимость).
- Способы организации систем передачи данных (на примерах телефонных сетей, стандартов X.25, Frame Relay, SONET, ISDN, ATM, семейство xDSL, GSM сети. GPRS служба), методы уплотнения данных в канале (мультиплексирование), организация и виды коммутаторов.
- Эталонная модель OSI ISO: понятие открытой системы, основные понятия модели, распределение функций сети между уровнями модели (физический, канальный, сетевой, транспортный, сессии, представления, приложений). Модель TCP/IP.
- Теоретические основы передачи данных (данные, сигнал, передача; взаимосвязь пропускной способности канала и ширины его полосы пропускания [теорема Нийквиста-Котельникова, теорема Шеннона]).
- Основные виды физических сред передачи данных и их свойства (витая пара, коаксиальный кабель, оптоволокно); беспроводные каналы (радио сети, микроволновые сети, сотовые сети, спутниковые сети).

Иллюстративный материал к лекции:





Теорема Котельникова - Найквиста

Любой непрерывный сигнал **с ограниченным частотным спектром** может быть представлен значениями этого сигнала, взятыми в дискретные моменты времени nT

Аналитическое представление дискретизированного сигнала как суммы δ–импульсов:

$$u(nT) = \sum_{n=-\infty}^{\infty} u(t)\delta(t - nT)$$

$u(t)$ – аналоговый сигнал

$u(nT)$ – дискретизированный сигнал

T - интервал дискретизации,

$n = \pm\infty$

Формула Шеннона

$$I = -\sum_{i=1}^{n} p(i) * \log_i (p(i))$$

где

I – количество информации (бит);

N – число возможных состояний системы;

p(i) – априорная вероятность каждого состояния системы.





Литература: Жуматий, С. А. Вычислительное дело и кластерные системы: курс : учебное пособие / С. А. Жуматий, В. В. Воеводин ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2008. – 125 с. : ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=234002

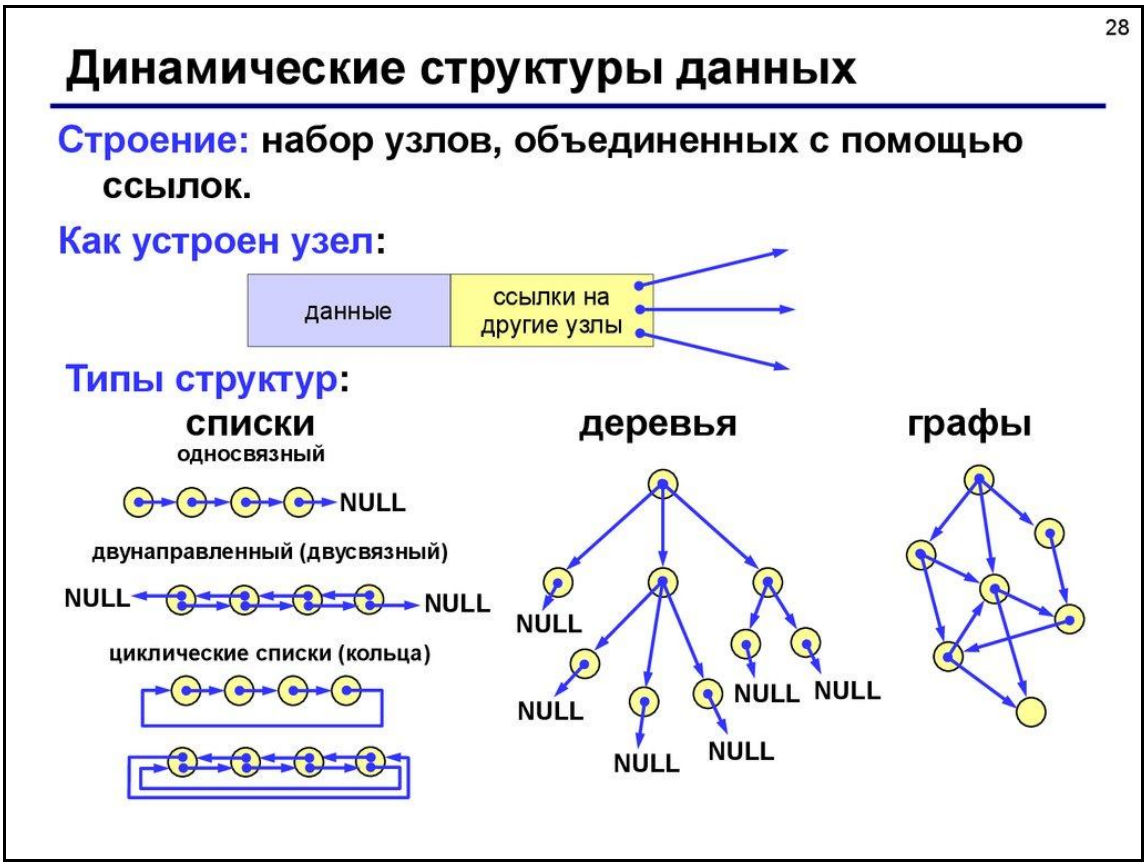
Лекция 3. Языки и системы программирования. Технология разработки программного обеспечения

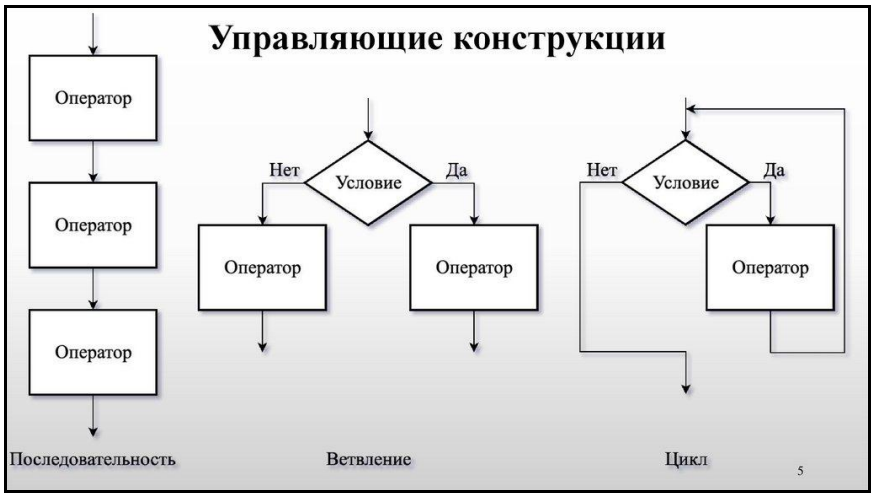
Тезисы лекции:

- Языки программирования. Понятие о парадигмах программирования. Процедурные, объектно-ориентированные и функциональные языки программирования. Понятие о логическом программировании.
- Концепция типа данных. Базисные типы данных в современных языках программирования (булевский, целый, вещественный, символьный, перечислимый, ограниченный, указатели, ссылки, массивы, записи).

- Динамические структуры данных (списки, бинарные деревья, стек, очередь), основные операции над ними, способы их машинного представления. Таблицы, поиск по ключу. Способы представления таблиц (упорядоченные массивы, деревья поиска, перемешанные таблицы). Оценки сложности (в среднем и в худшем случае) операций поиска по ключу и вставки нового элемента в таблицу.
- Основные управляющие конструкции в современных языках программирования. Процедуры (функции): вызов процедур, способы передачи параметров, локализация переменных, побочные эффекты.
- Понятие исключительной ситуации. Обработка исключительных ситуаций. Понятие модуля, раздельная трансляция (зависимая и независимая).
- Средства инкапсуляции данных. Абстрактные типы данных. Основные принципы объектно-ориентированного программирования. Понятие класса и его реализация в современных языках программирования. Объекты (основные свойства и отличительные признаки). Наследование. Полиморфизм. Динамическое связывание методов. Абстрактные классы и интерфейсы. Понятие о родовых модулях и шаблонах. Обобщенное программирование.
- Машинно-ориентированные языки, язык ассемблера. Представление машинных команд и констант. Команды транслятору. Их типы, принципы реализации. Макросредства, макровызовы, языки макроопределений, условная макрогенерация, принципы реализации.

Иллюстративный материал к лекции:





Основные понятия ООП

Основные механизмы (постулаты, принципы) ООП

1. **Инкапсуляция – механизм, связывающий вместе код и данные, которыми он манипулирует, и одновременно защищающий их от произвольного доступа со стороны другого кода, внешнего по отношению к рассматриваемому. Доступ к данным жестко контролируется интерфейсом.**
2. **Наследование – механизм, с помощью которого один объект (производного класса) приобретает свойство другого объекта (родительского, базового класса).**
3. **Полиморфизм – механизм, позволяющий использовать один и тот же интерфейс для общего класс действий.**

Язык ассемблера

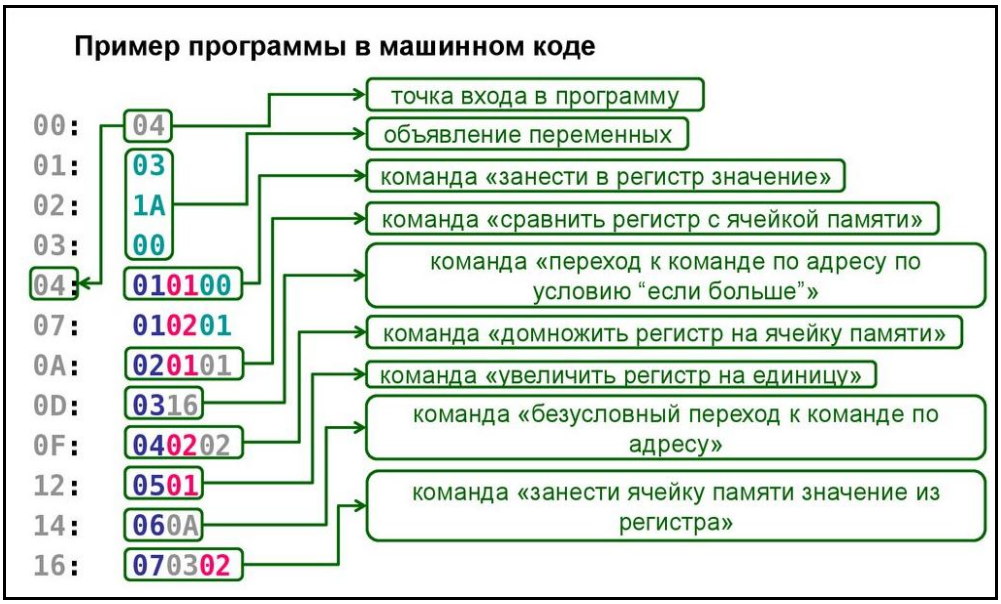
Язык ассемблера - это *символическое представление* машинного языка.

Язык образован в результате мнемонического (символического) кодирования.

Каждая машинная команда обозначается символом, представляющая собой сокращённую форму полной записи наименования данной команды на английском языке. При записи полного набора команд программы мнемонические обозначения применяют как для кода операции, так и для операндов и их адресов.

Для преобразования ассемблера в двоичные коды служит специальная программа - транслятор.

Программа на языке ассемблера - это последовательность команд (операторов), описывающая решение задачи.



Структура программы на языке Ассемблера

```
.data
   ; входные данные
   x db 3
   y dw 26

.data?
   ; выходные данные
   z dd ?

.code

   start:
   mov AX, x
   add AX, y
   mov z, AX

end start
```

Структура программы на языке Ассемблера

Программа на ассемблере представляет собой совокупность блоков, называемых *сегментами*.

Сегменты программы имеют назначение, соответствующее типам сегментов памяти (кода, данных и стека).

Каждый сегмент состоит из совокупности отдельных строк, называемых *предложениями языка Ассемблера*.

```
.486
.model flat, stdcall
option casemap: none

include windows.inc
include user32.inc
include kernel32.inc

includelib user32.lib
includelib kernel32.lib

.data
title    db "Message",0
message  db "Hello, World!",0
```

```
.code

start:

    push MB_OK
    push offset title
    push offset message
    push 0
    call MessageBox

    push 0
    call ExitProcess

end start
```

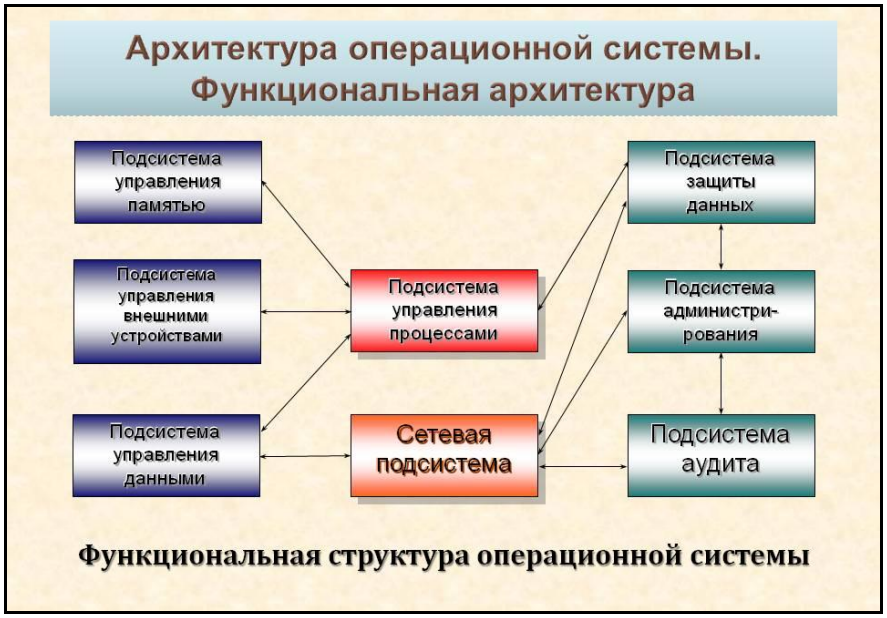
Литература: Мейер, Б. Инструменты, алгоритмы и структуры данных / Б. Мейер. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 543 с. : схем., ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=429033

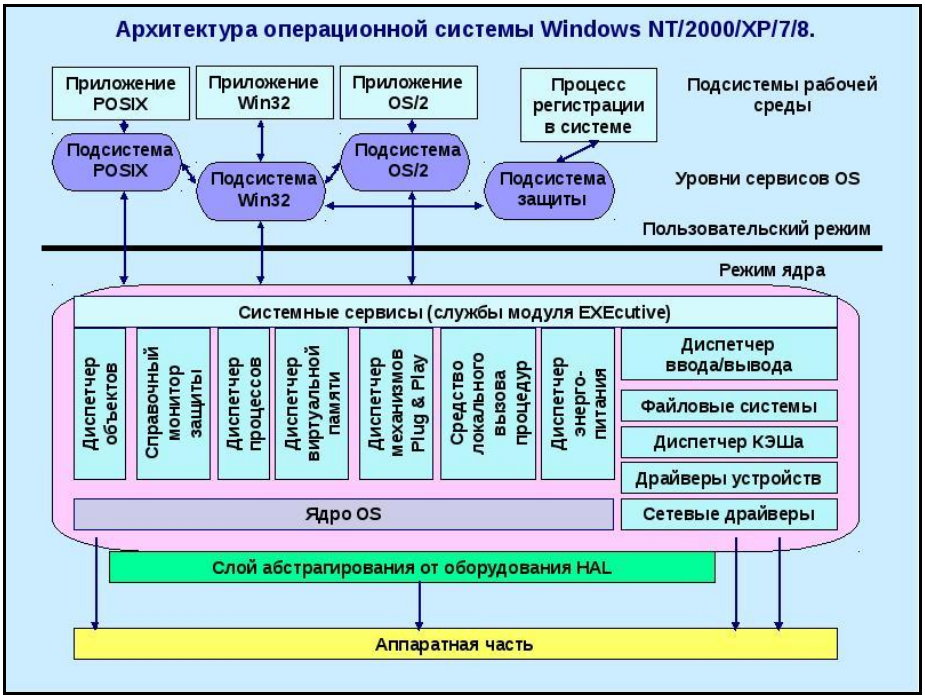
Лекция 4. Операционные системы

Тезисы лекции:

- Основы архитектуры операционных систем. Базовые понятия - процесс, ресурс. Физические, виртуальные (логические) ресурсы. Структура ОС. Ядро. Системные вызовы. Структурная организация ОС. Монолитные, микроядерные ОС. Логические функции ОС. Прерывания, обработка прерываний. Мультипроцессные (мультипрограммные) ОС, требования к аппаратуре компьютера по обеспечению корректного мультипроцессирования. Управление процессами. Управление устройствами. Управление данными. Задачи планирования.
- Типы ОС: пакетные, разделения времени, реального времени. Сетевые, распределенные ОС. Управление процессами. Определение процесса. Жизненный цикл, состояния процесса. Основные типы процессов: полновесные процессы, легковесные процессы (нити). Контекст процесса. Планирование распределения времени ЦП между процессами -основные подходы: вытесняющие и невытесняющие стратегии; алгоритмы, основанные на квантовании; алгоритмы, использующие приоритет: планирование по наивысшему приоритету, понятие относительного и абсолютного приоритета, алгоритмы, использующие изменяющийся приоритет. Очереди с обратной связью. Смешанные алгоритмы планирования.
- Взаимодействие процессов. Взаимодействие параллельных процессов и их синхронизация. Классификация средств межпроцессного взаимодействия. Разделяемые ресурсы и синхронизация доступа к ним. Взаимное исключение. Тупики. Механизмы обеспечения взаимного исключения: семафоры Дейкстры, мониторы Хоара, обмен сообщениями. Способы реализации взаимного исключения в однопроцессорных ЭВМ, многопроцессорных ЭВМ с общей памятью, в распределенных вычислительных системах. Классические задачи синхронизации процессов: «обедающие философы», «читатели и писатели», «спящий парикмахер». Типовые средства межпроцессного взаимодействия (базовые средства взаимодействия процессов ОС Unix, IPC ОС Unix, сокеты, MPI).
- Файловые системы. Файлы, структурная организация файлов. Атрибуты файлов. Типовые программные интерфейсы работы с файлами. Структура файловой системы, подходы в практической реализации. Модели реализации файлов. Понятие индексного узла (дескриптора). Модели реализации каталогов. Квотирование пространства файловой системы. Надежность файловой системы, целостность файловой системы. Распределенные файловые системы. Алгоритмы обеспечения консистентности кэшей в распределенных файловых системах.

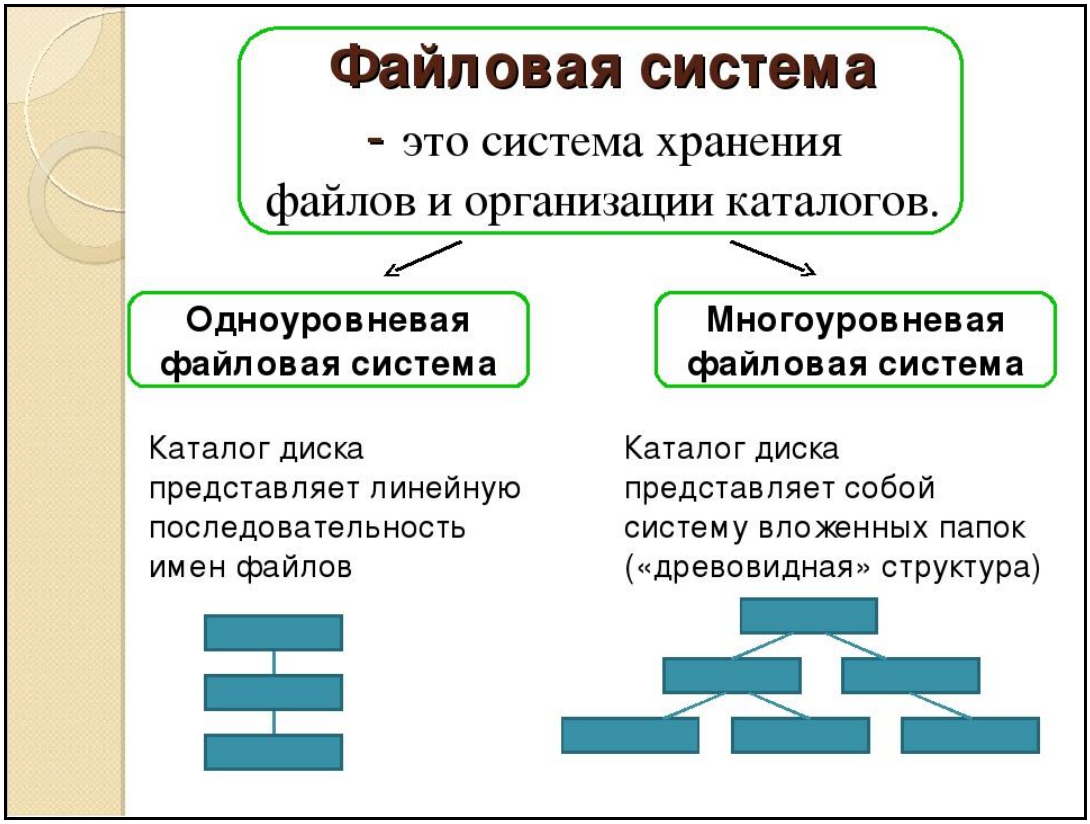
Иллюстративный материал к лекции:

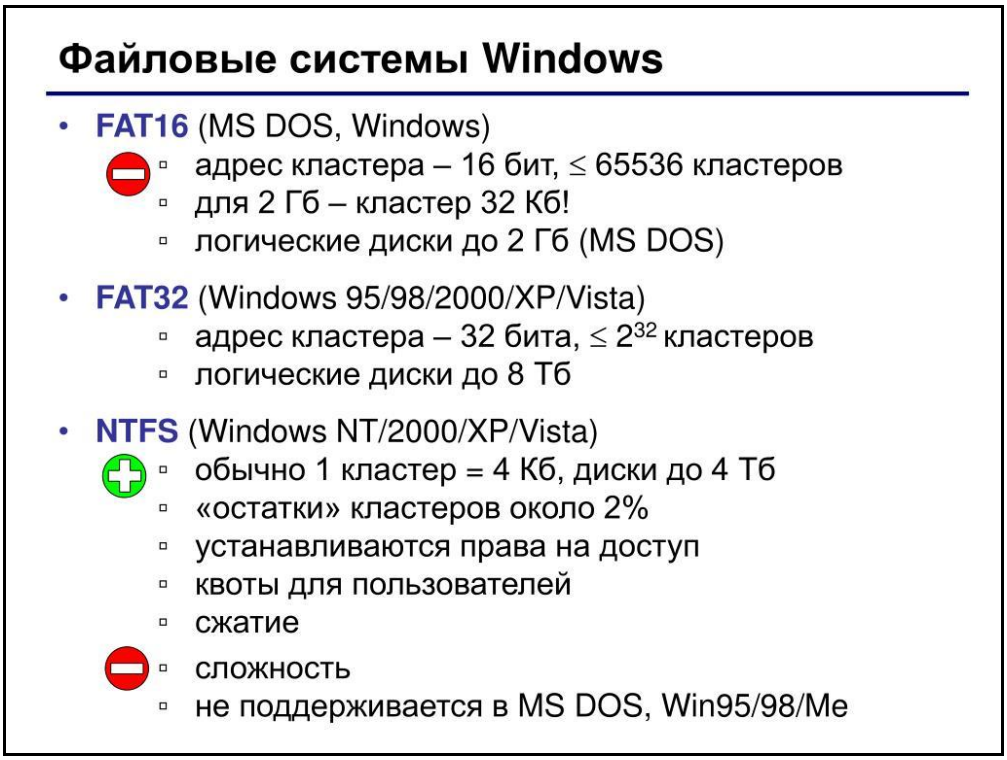






ОС пакетной обработки - из программ, которые нужно выполнить формируется пакет (набор) заданий, вводимых в ЭВМ и выполняемых в порядке очередности с возможным учетом приоритетности.
ОС разделения времени обеспечивают одновременный диалоговый режим доступа к ЭВМ нескольких пользователей на разных терминалах
ОС реального времени обеспечивают определенное гарантированное время ответа машины на запрос пользователя с управлением им какими-либо внешними по отношению к ЭВМ событиями, процессами или объектами.





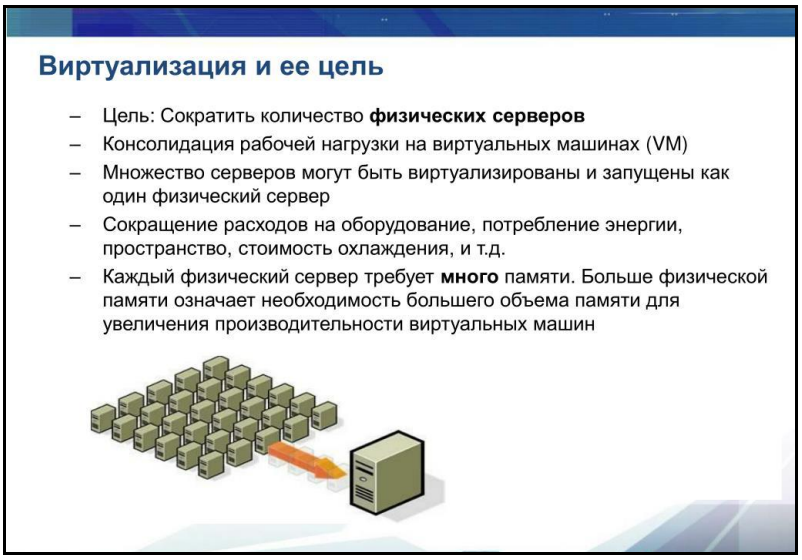
Литература: Жуматий, С. А. Вычислительное дело и кластерные системы: курс : учебное пособие / С. А. Жуматий, В. В. Воеводин ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2008. – 125 с. : ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=234002

Лекция 5. Облачные вычисления

Тезисы лекции:

- История основных типов высокопроизводительных вычислений, тенденции развития современных инфраструктурных решений. Виртуализация. Виртуальная машина. Виртуализация серверов. Виртуализация приложений. Виртуализация представлений (рабочих мест). Разновидности архитектуры гипервизора.
- Введение в понятия облачных вычислений. Обзор парадигмы облачных вычислений. Архитектура облачных систем. Модели развертывания облаков: частное облако, публичное облако, гибридное облако, общественное облако. Основные модели предоставления услуг облачных вычислений. Различия между облачными и кластерными вычислениями. Достоинства и недостатки облачных вычислений. Обзор существующих сервисов и платформ. Технологии облачных вычислений.
- Миграция из стандартной среды в облачные приложения. Концепция миграции. Фазы миграции в облако. Выбор подходящей модели развертывания в соответствии с существующими бизнес-задачами. Выбор подходящего поставщика облачных услуг. Концепция SLA. Производительность облачной инфраструктуры. Концепция вендора. Открытые стандарты для обеспечения облачных услуг.

Иллюстративный материал к лекции:









Литература: Жуматий, С. А. Вычислительное дело и кластерные системы: курс : учебное пособие / С. А. Жуматий, В. В. Воеводин ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2008. – 125 с. : ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=234002

Лекция 6. Организация баз данных и знаний

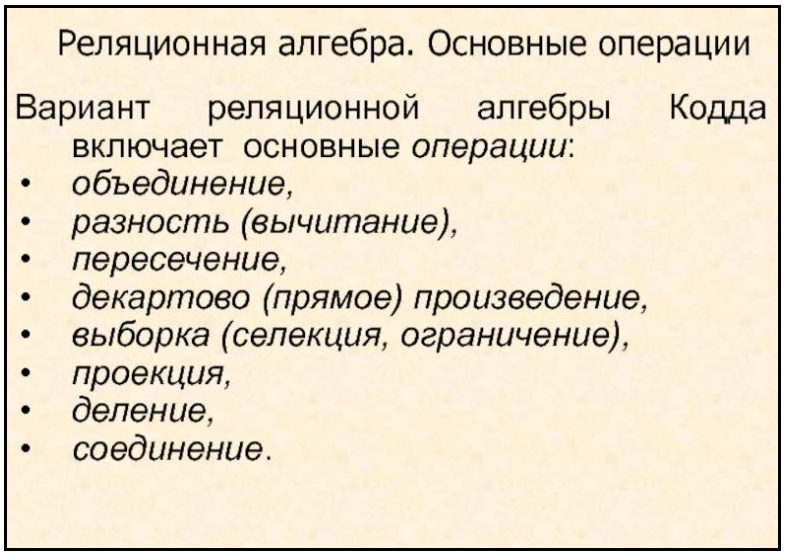
Тезисы лекции:

- Основные понятия структурной части реляционной модели данных. Тип данных, домен, заголовок отношения, кортеж, тело отношения, переменная отношения. Целостная часть реляционной модели данных: возможные и первичный ключи,

целостность сущности; понятие внешнего ключа, ссылочная целостность. Манипуляционная часть реляционной модели данных.

- Реляционная алгебра Кодда: базовые теоретико-множественные и специальные реляционные операции. Общая операция соединения, эквисоединение, естественное соединение. Минимальный набор операций алгебры Кодда.
- Реляционное исчисление кортежей. Кортежные переменные, правильно построенные формулы, кванторы, свободные и связанные переменные. Выражения исчисления кортежей. Функциональные зависимости. Замыкания функциональных зависимостей, аксиомы Армстронга. Покрытия функциональных зависимостей, Декомпозиция обновлений минимальное покрытие атрибутов отношения. отношения без потерь, теорема Хита. Аномалии и нормализация отношений. Минимальные функциональные зависимости, вторая нормальная форма отношений. Транзитивные функциональные зависимости, третья нормальная форма. Независимые проекции, теорема Риссанена. Атомарные проекции. Перекрывающиеся возможные ключи, нормальная форма Бойса-Кодда.
- Обобщенная архитектура, компоненты и функции системы управления базой данных (СУБД). Структуры внешней памяти реляционных баз данных, способы организации индексов. Алгоритмы буферизации базы данных в основной памяти. Физическая и логическая целостность баз данных.
- Понятие транзакции. Способы поддержания логической целостности данных, ограничения целостности и триггеры. Методы восстановления баз данных после сбоев, восстановление физически и логически целостного состояния базы данных.
- Авторизация доступа к базе данных, механизм привилегий. Понятие ACID-транзакции. Минимальные требования к поддержке атомарности, согласованности, изолированности и сохранности результатов транзакций. Виды конфликтов транзакций. Сериализация транзакций. Методы сериализации транзакций. Двухфазный протокол синхронизационных блокировок. Гранулированные и предикатные блокировки. Сериализация транзакций на основе временных меток.
- Журнализация изменений базы данных, протокол WAL. Логическая и физическая журнализация. Индивидуальные откаты транзакций. Восстановления базы данных по журналу после мягких и жестких сбоев.

Иллюстративный материал к лекции:



ACID – требования

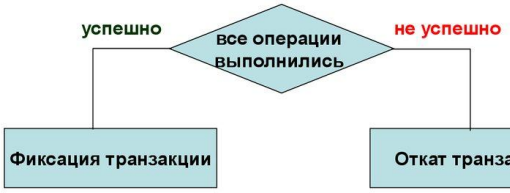
ACID – требования гарантируют правильность и надежность работы системы

Atomic (атомарность)	**Транзакция не может выполниться частично, либо все, либо ничего**
Consistency (согласованность)	**После выполнения транзакции все данные должны находиться в согласованном состоянии**
Isolation (изолированность)	**Транзакция должна быт автономной и воздействовать на другие транзакции или завысить от них**
Durability (устойчивость)	**После завершения транзакции, внесенные изменения останутся неизменными**

ACID - фундаментальные свойства систем обработки транзакций

Варианты завершения транзакций

2 варианта завершения транзакций



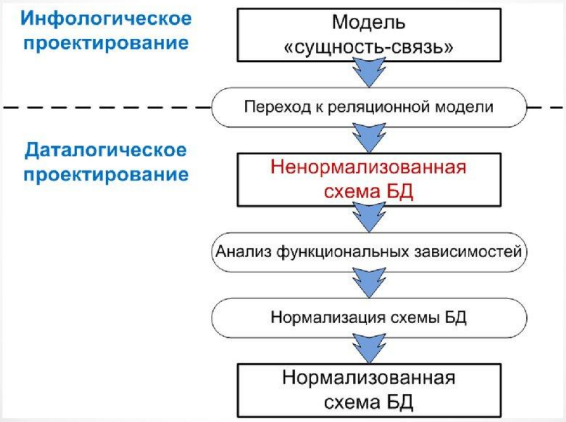
Фиксация транзакции – это **действия**, обеспечивающие **сохранение** на диске изменений БД, сделанные в процессе выполнения транзакции

Откат транзакции – это **действия**, обеспечивающие аннулирование **всех** изменений БД, сделанные в процессе выполнения транзакции

Критерии выбора СУБД

- тип модели данных, которую поддерживает данная СУБД;
- характеристики производительности СУБД;
- запас функциональных возможностей для дальнейшего развития ИС;
- степень оснащенности СУБД;
- удобство и надежность СУБД в эксплуатации;
- стоимость СУБД и дополнительного ПО.

Даталогическое проектирование



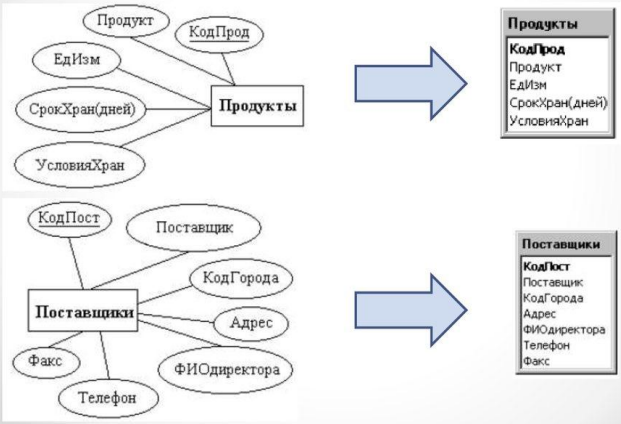
Преобразование диаграммы «сущность-связь» в реляционную БД

1. Каждая сущность превращается в таблицу.
2. Каждый атрибут становится столбцом таблицы с тем же именем.
3. Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы.

Преобразование диаграммы «сущность-связь» в реляционную БД

4. Связи 1:1 становятся внешними ключами.
5. Для поддержания связи M:N между типами сущности A и B создается дополнительная таблица C.
6. Индексы создаются для первичного ключа, внешних ключей и тех атрибутов, на которых предполагается базировать запросы.

Переход к реляционной модели



Физическая модель данных

Описывает данные средствами конкретной СУБД, предполагает выбор эффективного размещения БД на внешних носителях.

Критерии выбора СУБД

- тип модели данных, которую поддерживает данная СУБД;
- характеристики производительности СУБД;
- запас функциональных возможностей для дальнейшего развития ИС;
- степень оснащенности СУБД;
- удобство и надежность СУБД в эксплуатации;
- стоимость СУБД и дополнительного ПО.

Проектирование БД на основе принципов нормализации

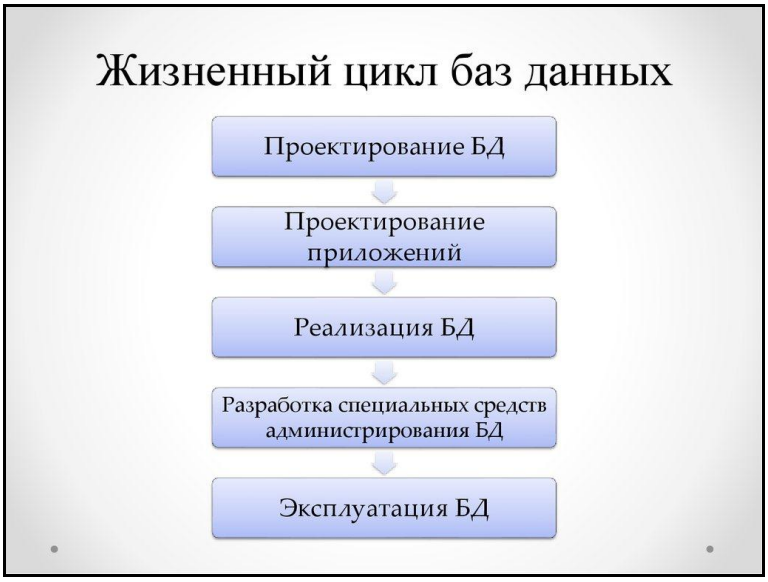
Нормализация – процесс реорганизации данных путем ликвидации избыточности данных и иных противоречий с целью приведения таблиц к виду, позволяющему осуществлять непротиворечивое и корректное редактирование данных.

Нормальные формы

1. первая нормальная форма (1НФ);
2. вторая нормальная форма (2НФ);
3. третья нормальная форма (3НФ);
4. усиленная третья нормальная форма, или нормальная форма Бойса –Кодда (БКНФ);
5. четвертая нормальная форма (4НФ);
6. пятая нормальная форма или нормальная форма проекции-соединения (5НФ).

Виды аномалий

1. аномалия удаления;
2. аномалия обновления;
3. аномалия ввода.



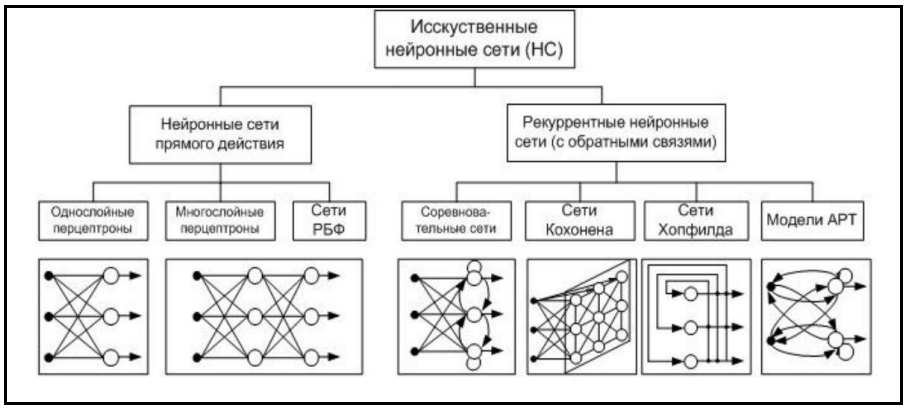
Литература: Жуматий, С. А. Вычислительное дело и кластерные системы: курс : учебное пособие / С. А. Жуматий, В. В. Воеводин ; Национальный Открытый Университет "ИНТУИТ". – Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2008. – 125 с. : ил.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=234002

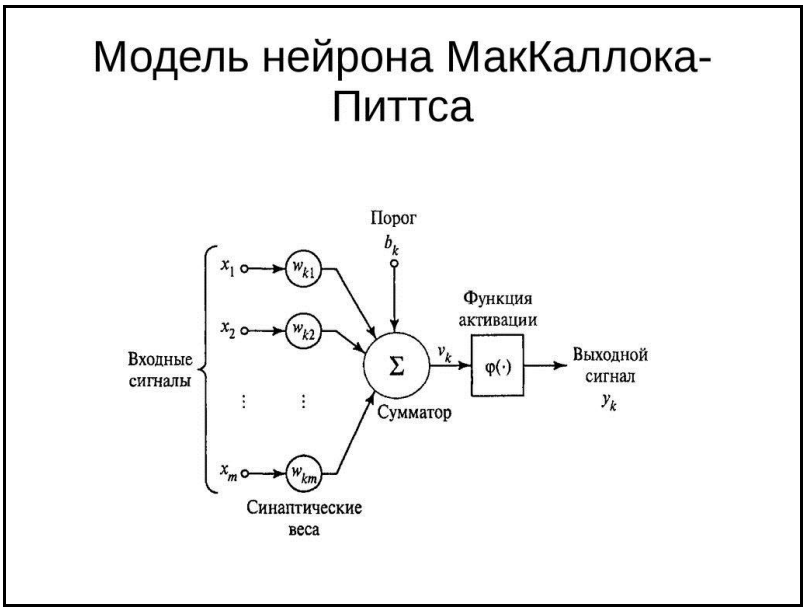
Лекция 7. Искусственные нейронные сети

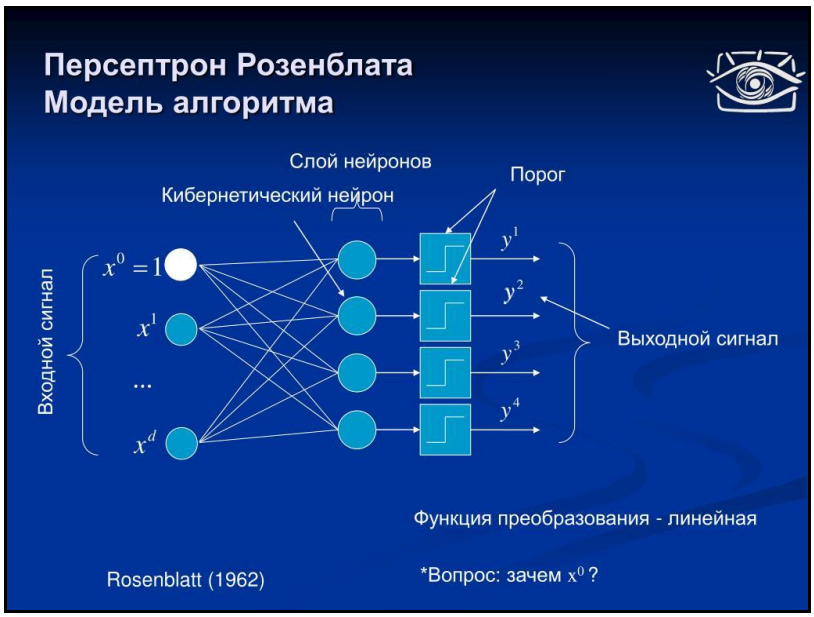
Тезисы лекции:

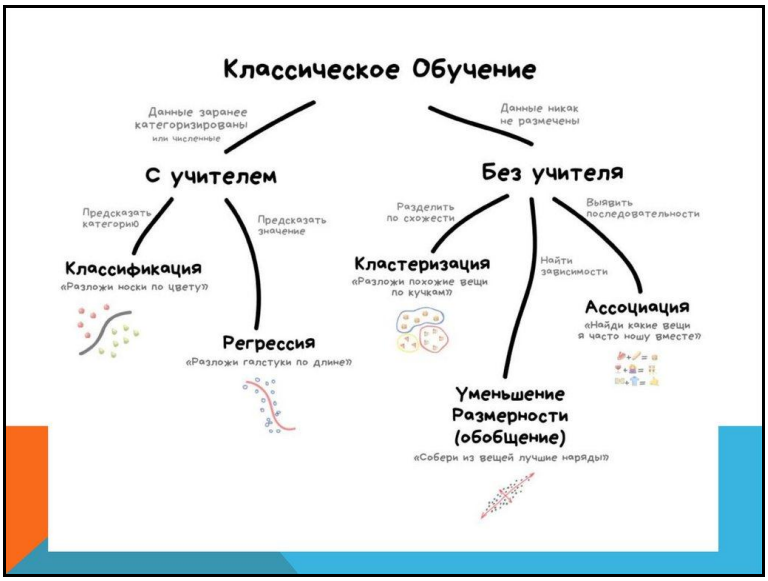
- Формальный нейрон Маккалока-Питтса.
- Персептрон Розенблатта.
- Линейная разделимость. Преодоление проблемы линейной разделимости.
- Обучение с учителем: классификация образов.
- Обучение с учителем: аппроксимация многомерных функций.
- Сети прямого распространения.
- Алгоритм обратного распространения ошибки.
- Проблемы обучения: ошибка аппроксимации; переобучение; ошибка, связанная со сложностью модели.
- Оптимизация размера сети.
- Адаптивная оптимизации архитектуры сети.
- Валидация обучения.
- Ранняя остановка обучения.
- Сети встречного распространения.
- Обучение без учителя.
- Структура слоев Кохоненна и Гроссберга.
- Предварительная обработка входных векторов.
- Выбор начальных значений весовых векторов.
- Статистические свойства обученной сети.

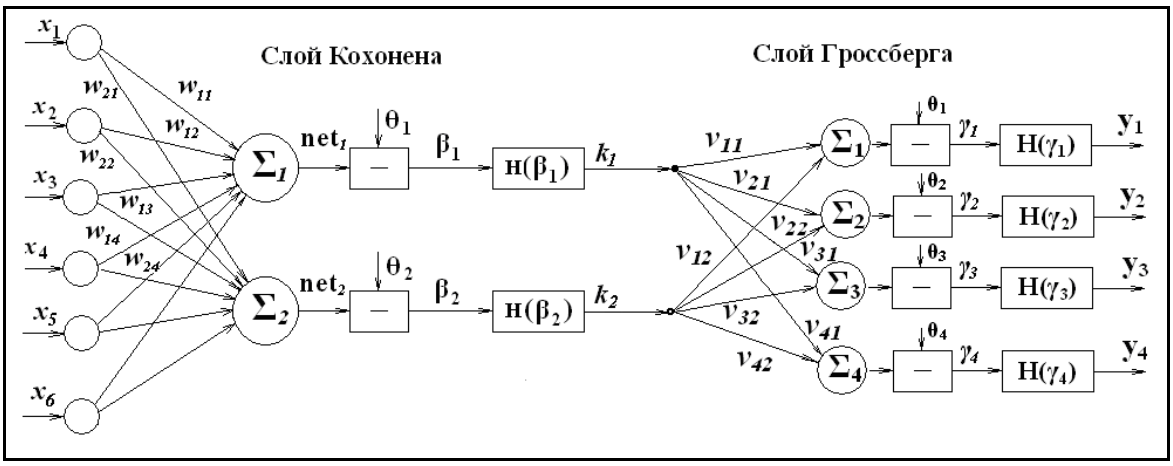
Иллюстративный материал к лекции:

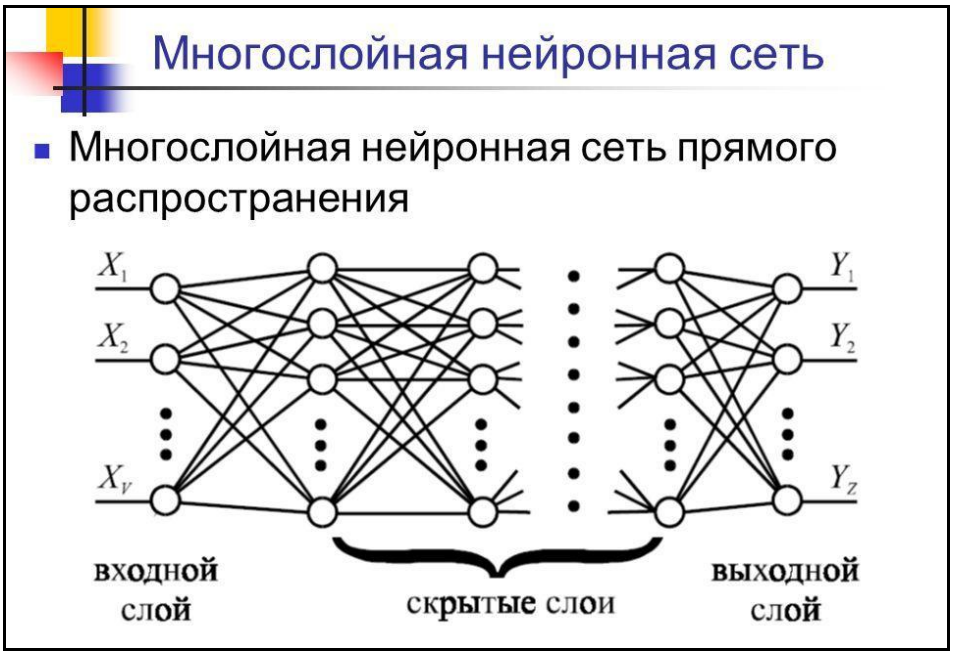


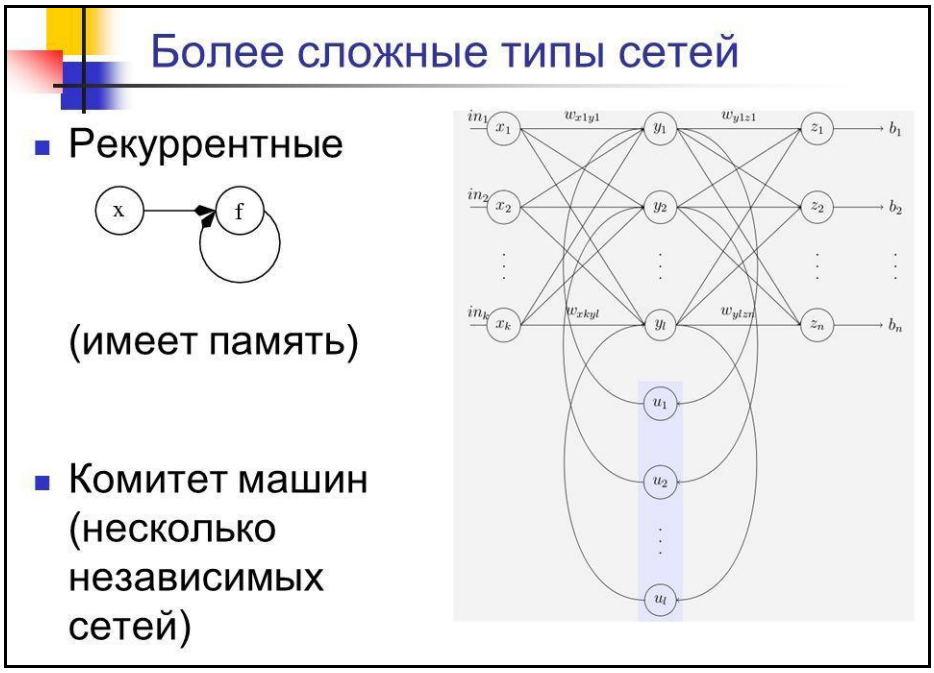





Классическое Обучение
Данные заранее категоризированы или численные
Данные никак не размечены
С учителем
Без учителя
Предсказать категорию
Предсказать значение
Разделить по схожести
Выявить последовательности
Найти зависимости
Классификация
«Разложи носки по цвету»
Регрессия
«Разложи галстуки по длине»
Кластеризация
«Разложи похожие вещи по кучкам»
Ассоциация
«Найди какие вещи я часто ношу вместе»
Уменьшение Размерности (обобщение)
«Собери из вещей лучшие наряды»


Слой Кохонена
Слой Гроссберга
x1
x2
x3
x4
x5
x6
w11
w12
w13
w14
w21
w22
w24
Σ1
Σ2
net1
net2
θ1
θ2
β1
β2
H(β1)
H(β2)
k1
k2
v11
v12
v21
v22
v31
v32
v41
v42
Σ1
Σ2
Σ3
Σ4
θ1
θ2
θ3
θ4
γ1
γ2
γ3
γ4
H(γ1)
H(γ2)
H(γ3)
H(γ4)
y1
y2
y3
y4


Многослойная нейронная сеть
Многослойная нейронная сеть прямого распространения
X1
X2
XV
Y1
Y2
YZ
входной слой
скрытые слои
выходной слой



Литература: Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский ; пер. с пол. И. Д. Рудинского. – 2-е изд., стер. – Москва : Горячая линия – Телеком, 2013. – 384 с. : ил. – ISBN 978-5-9912-0320-3; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=253603

Лекция 8. Программная инженерия

Тезисы лекции:

- Основы программной инженерии. Понятие жизненного цикла программного обеспечения (ЖЦ ПО). Стандарт ISO/IEC 12207 (Information Technology - Software Life Cycle Processes).
- Процессы ЖЦ ПО: основные, вспомогательные и организационные. Понятия модели и стадии ЖЦ ПО. Каскадная и итерационная модели ЖЦ. Методы «быстрой» разработки ПО (Agile Software Development). Понятие зрелости процессов создания ПО (Software Process). Модель оценки зрелости CMM (Capability Maturity Model). Уровни зрелости процессов создания ПО (начальный, воспроизводимый, определенный, управляемый, оптимизируемый).
- Технологии создания ПО. Понятие технологии создания ПО и ее состав (технологические процессы, технологические операции, рабочие продукты, роли, руководства, инструментальные средства). Требования, предъявляемые к технологии. Факторы оценки и выбора технологии. Технологии создания ПО как программные продукты. Унифицированный процесс разработки ПО (Unified Software Development Process).
- Метрики ПО. Оценка сложности и размера ПО. Стандартные метрики, применяемые для оценки размера ПО (количество строк кода и функциональных точек (function points)). Методы оценки трудоемкости создания ПО (статистические модели, экспертные оценки, оценки по аналогии). Базовая методика оценки трудоемкости Боэма (COCOMO – Конструктивная модель стоимости). Расчет трудоемкости по методу функциональных точек. Методика выявления функциональных типов и подсчета количества функциональных точек.

- Качество ПО. Понятие качества ПО. Показатели качества ПО (функциональность, надежность, удобство использования, производительность, адаптируемость). Методы обеспечения качества ПО (отладка, тестирование, верификация и аттестация ПО).
- Методы, принципы, стратегии и этапы тестирования ПО. Функциональное и нагрузочное тестирование ПО, тестирование пользовательского интерфейса. Модульное, интеграционное и регрессионное тестирование ПО. Организация и планирование тестирования ПО.

Иллюстративный материал к лекции:





- **Качество программного обеспечения** - это совокупность свойств, характеризующих способность программного обеспечения удовлетворять потребностям пользователя в соответствии с предназначением.
- Для количественной оценки качества ПО используют **метрики и модели измерений**.
- **Метрики** определяются по модели измерения атрибутов ПО на всех этапах ЖЦ:

1. *метрики программного продукта*, которые используются при измерении его характеристик - свойств;
2. *метрики процесса*, которые используются при измерении свойства процесса ЖЦ создания продукта.
3. *метрики использования*.



Цикломатическая мера Мак-Кейба

- **Математически цикломатическая сложность** структурированной программы определяется с помощью ориентированного графа, узлами которого являются блоки программы, соединенные рёбрами, если управление может переходить с одного блока на другой.

Тогда сложность определяется как:

$$M = E - N + 2P, \text{ где:}$$

- M = цикломатическая сложность,
- E = количество рёбер в графе,
- N = количество узлов в графе,
- P = количество компонент связности.

Мера Дж. Майерса (интервальная мера)

- В качестве оценки используется интервал [V1(G),V2(G)+h], где V1 – цикломатическая мера, а V2 – число отдельных условий плюс единица.
- Данный метод позволяет различать разные по сложности предикаты (логические высказывания).

Метрика граничных значений

Вычисляется как S0=1-(v-1)/Sa , где

S0 – относительная граничная сложность программы,

Sa – абсолютная граничная сложность программы,

v – общее число вершин графа потока управления.

Абсолютная граничная сложность определяется как сумма приведенных сложностей всех вершин графа.

Метрика Чепина

- Оценивает информационную прочность отдельно взятого модуля по характеру использования переменных из списка ввода-вывода. Всё множество переменных разбивается на 4 группы:
- **P** (вводимые для расчетов и для обеспечения вывода),
- **M** (модифицируемые, или создаваемые внутри программы),
- **C** (управляющие),
- **T** (паразитные, то есть неиспользуемые). Переменные, выполняющие несколько функций, учитываются в каждой функциональной группе.
- Метрика Чепина выражается формулой

$$Q = a1*P + a2*M + a3*C + a4*T,$$

- где a1, a2, a3, a4 – весовые коэффициенты равные 1, 2, 3 и 0.5

Выполнение оценки проекта на основе loc- и fp-метрик

- **Размерно-ориентированные метрики прямо измеряют программный продукт и процесс его разработки.** Основываются на LOC-оценках.
- **LOC-оценка** — это количество строк в программном продукте. Исходные данные для расчета этих метрик сводятся в таблицу

Работа	Затраты, чел./мес.	Стоимость, тыс. $	KLOC, тыс. LOC	Прогр. док- ты, страниц	Ошибки	Люди
Закупка	25	168	15,2	200	30	4
Оборудование	63	350	12,1	16354	90	7
Отчет	48	420	25,5	25640	64	10

Выполнение оценки проекта на основе loc- и fp-метрик

$$\text{Производительность} = \frac{\text{Длина}}{\text{Затраты}}\left[\frac{\text{тыс.LOC}}{\text{чел.-мес}}\right];$$

$$\text{Качество} = \frac{\text{Ошибки}}{\text{Длина}}\left[\frac{\text{Единиц}}{\text{тыс.LOC}}\right];$$

$$\text{Удельная стоимость} = \frac{\text{Стоимость}}{\text{Длина}}\left[\frac{\text{Тыс\$}}{\text{LOC}}\right];$$

$$\text{Документированность} = \frac{\text{Страниц Документа}}{\text{Длина}}\left[\frac{\text{Страниц}}{\text{тыс.LOC}}\right].$$

Выполнение оценки проекта на основе loc- и fp-метрик

Функционально-ориентированные метрики косвенно измеряют программный продукт и процесс его разработки. Вместо подсчета LOC- оценки рассматривается не размер, а функциональность или полезность продукта.

Используется 5 информационных характеристик.

1. **Количество внешних вводов.**
2. **Количество внешних выводов.**
3. **Количество внешних запросов.**
4. **Количество внутренних логических файлов.**
5. **Количество внешних интерфейсных файлов.**



Литература: Биллиг, В. А. Параллельные вычисления и многопоточное программирование / В. А. Биллиг. – 2-е изд., испр. – Москва : Национальный Открытый Университет «ИНТУИТ», 2016. – 311 с. : ил., схем.; То же [Электронный ресурс]. - URL: https://biblioclub.ru/index.php?page=book&id=428948

Лекция 9. Итоговая лекция. Подведение и анализ промежуточных результатов освоения дисциплины

Итоговая лекция завершает изучение дисциплины, обобщает пройденное за весь период. При ее подготовке учитываются пробелы в знаниях аспирантов, выявленные на практических занятиях и при оценке самостоятельной работы. Аспиранты получают установку на их устранение и на дальнейшее совершенствование своей подготовки в данной области. На лекции подводятся итоги изучения дисциплины, показывается ее значение в профессиональной сфере и формировании научного мировоззрения, обсуждается вопрос о подведении итогов изучения дисциплины – прохождения тестирования и сдаче кандидатского экзамена по специальности.

2. Методические указания по подготовке к занятиям семинарского типа

2.1. Планы и (или) методические рекомендации к практическим занятиям

В рамках проведения занятий по дисциплине предполагается организация следующих практических занятий:

Практическое занятие 1. Математические основы программирования

Вопросы для обсуждения в рамках практического занятия:

1. Приведите различные интуитивные понятия алгоритма. Что в этих определениях общего?
2. Каким требованиям должен удовлетворять алгоритм? Какие свойства важны с практической точки зрения при компьютерной реализации алгоритма?
3. Приведите структуру алгоритмической системы. Охарактеризуйте взаимосвязи между элементами этой системы.
4. Каким образом связаны понятия алгоритма и вычислимой функции?
5. Что называется порождающей грамматикой? Какие элементы составляют порождающую грамматику?
6. Приведите неформальное определение машины Тьюринга. Дайте характеристику элементам машины Тьюринга. Каким реальным устройствам соответствуют компоненты машины?

Задачи для решения:

1. Опишите алгоритмическую систему, которая может быть использована при описании следующих алгоритмов:
   нахождение корней квадратного уравнения;
   нахождение наименьшего общего делителя двух натуральных чисел;
   нахождение наименьшего общего кратного двух чисел.
2. В алфавите V = { a,b,c,d,e } найдите:
   а) лексикографический номер слов dabcce, bbacdb, eaebca ;
   б) слова, соответствующие номерам 6736, 12346, 27737.

Шкала оценивания:

Оценка «отлично» за практическую работу выставляется, если оба задания выполнены не меньше, чем на 85%.

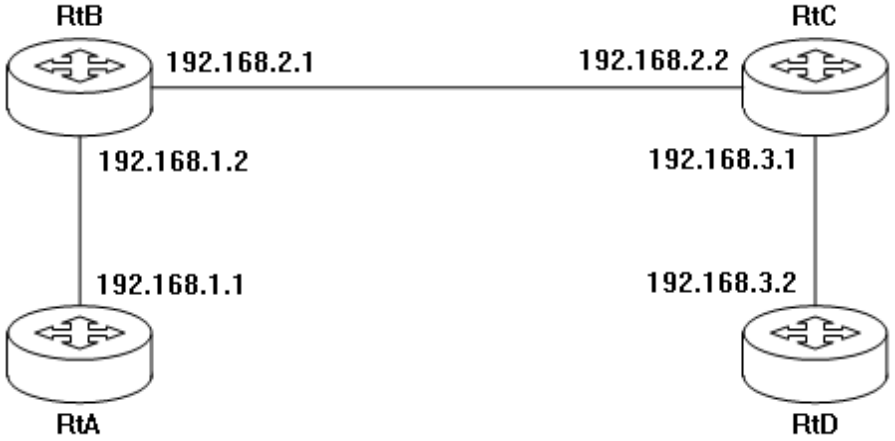
Оценка «хорошо» за практическую работу выставляется, если оба задания выполнены не меньше, чем на 71%.

Оценка «удовлетворительно» за практическую работу выставляется, если оба задания выполнены не меньше, чем на 61%.

Оценка «неудовлетворительно» за практическую работу выставляется, если любое из заданий выполнено меньше, чем на 61%.

Практическое занятие 2. Компьютерные сети

Цель работы: Соединить и настроить маршрутизаторы для работы в сети со следующей топологией:



Пошаговая инструкция:

1. Соедините соответствующие порты маршрутизаторов crossover-кабелями.
2. Запустите терминальную программу, например, HyperTerminal и откройте терминальную сессию с нужными параметрами.
3. Подключите консольный кабель к первому маршрутизатору.
4. Пользуясь терминалом:
   a) войдите в режим глобальной конфигурации;
   b) измените имя маршрутизатора на `RtA`;
   c) настройте интерфейс, к которому подключен соседний маршрутизатор:
      i. войдите в режим конфигурирования интерфейса;

ii. задайте IP-адрес для данного интерфейса;
iii. активируйте интерфейс;
iv. выйдите из режима конфигурирования интерфейса;

d) если у маршрутизатора используются другие интерфейсы, то повторите шаг с для каждого из них;
e) пропишите статические пути для каждой сети, которая не является соседней для данного маршрутизатора;
f) выйдите из режима глобальной конфигурации;

5. Повторите пункты 3-4 для каждого маршрутизатора.
6. Выполните проверку связи между маршрутизаторами `RtA` и `RtD` в обоих направлениях с помощью команд `ping` и `traceroute`.
7. В отчете отразите следующую информацию по каждому маршрутизатору:
   a) команды, необходимые для конфигурации, с пояснениями сути каждой команды;
   b) таблицу маршрутизации;
   c) результаты выполнения команд `ping` и `traceroute`.

Шкала оценивания:

Оценка «отлично» за практическую работу выставляется, если работа выполнена полностью правильно и самостоятельно, а аспирант способен уверенно отвечать на вопросы преподавателя о проделанной работе;

Оценка «хорошо» выставляется, если работа выполнена в целом корректна, но имеются неточности в ответах на вопросе преподавателя или некоторые действия выполнены неоптимально;

Оценка «удовлетворительно» выставляется, когда работа не может быть завершена аспирантом самостоятельно, без помощи преподавателя.

Оценка «неудовлетворительно» выставляется, если работа не была выполнена.

Практическое занятие 3. Языки и системы программирования. Технология разработки программного обеспечения

Тематика: Основные управляющие конструкции в современных языках программирования.

Задачи для решения:

1. Даны действительные числа a, b, c, d, e - стороны выпуклого пятиугольника. Найти площадь пятиугольника. (Определить функцию вычисления площади треугольника по его сторонам.)
2. Даны действительные числа x1, y1, x2, y2, …,x10, y10. Найти периметр десятиугольника, вершины которого имеют соответственно координаты (x1, y1), (x2, y2), …, (x10, y10). (Определить функцию вычисления расстояния между двумя точками, заданными своими координатами.)
3. Подсчитать сколько дней в январе было дней с температурой ниже, выше и равной средней температуре месяца (пользователь вводит по одной температуре, соответствующей определённому дню).
4. По фигурному катанию N судей выставляют оценки. Наибольшая и наименьшая отбрасываются, а по остальным выставляется средний бал. Напишите программу, вычисляющую среднюю оценку и выдайте номер судей, которые поставили наибольшую и наименьшую оценки.
5. Дан список целых чисел. Найти сумму максимума среди чётных и минимума среди нечётных.

6. Даны координаты вершин двух треугольников. Определить, какой из них имеет большую площадь. (Определить функцию вычисления площади треугольника по его вершинам.)
7. Написать программу, которая выводит в порядке возрастания все правильные несократимые дроби, знаменатели которых не превосходят N (2<=N<=500).
8. Написать функцию сложения двух чисел, используя только прибавление единицы.
9. Написать функцию умножения двух чисел, используя только операцию сложения.
10. Найти первые N чисел Фибоначчи двумя способами: с помощью рекурсии и с помощью итерации. Сравнить эффективность алгоритмов.
11. Рекурсивно подсчитать количество цифр в заданном числе.

Шкала оценивания:

Оценка «отлично» за практическую работу выставляется, если было решено не меньше, чем 85% всех представленных заданий.

Оценка «хорошо» за практическую работу выставляется, если было решено не меньше, чем 71% всех представленных заданий.

Оценка «удовлетворительно» за практическую работу выставляется, если было решено не меньше, чем 61% всех представленных заданий.

Оценка «неудовлетворительно» за практическую работу выставляется, если было решено менее 61% всех представленных заданий.

Практическое занятие 4. Операционные системы

Тематика: Файловые подсистемы.

Диск делится на разделы. Как именно диск делится на разделы, определяется таблицей разделов. Таблицы разделов бывают двух типов : MBR и GPT.

Структура MBR: Первые 512 байт (первый сектор диска) главного устройства хранения данных занимает MBR (Master Boot Record). В состав MBR входит 446 байт кода загрузчика, четыре записи по 16 байт - это таблица разделов, 2 байта сигнатуры. Таблица разделов может состоять из первичных разделов (до 4) и логических разделов(до 128).

Структура GPT: GUID Partition Table, аббр. GPT — стандарт формата размещения таблиц разделов на физическом жестком диске. Он является частью расширяемого микропрограммного интерфейса (англ. Extensible Firmware Interface, EFI) — стандарта, предложенного Intel на смену BIOS. EFI использует GPT там, где BIOS использует главную загрузочную запись (англ. Master Boot Record, MBR). В GPT нет собственной программы-загрузчика, вместо этого он работает в паре с EFI. Внутри GPT используется адресация логических блоков LBA, которая абстрагирована от физики устройства (в отличие от CHS — «Цилиндр-Головка-Сектор»). Каждый логический блок занимает 512 байт. LBA 0 — первые 512 байт диска, LBA 1 — следующие, и так далее. Отрицательные значения LBA означают смещение в блоках с конца диска. Последний блок имеет смещение «-1» (LBA -1).

Процесс загрузки компьютера является многоступенчатым процессом, и начинается он с инициализации системных устройств набором микропрограмм, называемых BIOS (Basic Input/Output System), которые выполняются при старте системы. После того, как BIOS успешно проверит системные устройства, идет процесс поиска загрузчика в MBR устройств хранения (CD/DVD диски, USB диск, HDD, SSD и др.) или на первом разделе устройства. После того, как загрузчик получил управление, он получает таблицу разделов и готовит к загрузке операционную систему. В семействе загрузчиков GNU/Linux яркими представителями являются GRUB и LILO. В них MBR состоит из небольшой части ассемблерного кода. Стандартный загрузчик Windows/DOS в состоянии проверить только активный раздел, считать несколько секторов с этого раздела и затем передать управление операционной системе. Он не в состоянии загрузить

Linux, так как не наделен необходимым функционалом. GRand Unified Bootloader (GRUB) - это стандартный загрузчик для операционных систем семейства GNU/Linux, и всем пользователям рекомендуется по умолчанию установить его в MBR, для того чтобы иметь возможность загружать операционную систему с любого раздела, первичного или логического.

В рамках этой практической вам потребуются следующие консольные команды:

fdisk <параметры> - Консольная программа для управления дисками (Работает только с MBR).

parted <параметры> - Консольная программа для управления дисками (Работает как с MBR, так и с GPT).

dd <параметры> - Консольная программа копирования данных.

mkfs.<тип файловой системы> <раздел диска> - Класс консольных команд создания файловых систем на разделах.

mount -t <тип файловой системы> <раздел диска> <точка монтирования> - Консольная программа монтирования разделов жесткого диска.

Задания

1. Добавьте в виртуальную машину с операционной системой Linux виртуальный жесткий диск (делается это в настройках виртуальной машины).
2. Запустите виртуальную машину с операционной системой Linux.
3. Ознакомьтесь с командой fdisk и ее возможностями из справочной документации.
4. Создайте таблицу разделов (3 первичных и 1 логический) с помощью команды fdisk на добавленном виртуальном диске (обычно это диск /dev/sdb).
5. Запишите изменения на диск
6. Проверьте факт создания разделов используя команду fdisk. (Так же, создание разделов можно проверить используя команду ls /dev/sd*)
7. Отформатируйте созданные разделы в файловую систему ext4.
8. Ознакомьтесь с командами mount и umount и их возможностями из справочной документации.
9. Смонтируйте созданные разделы и создайте там произвольные файлы.
10. Сделайте резервную копию MBR с помощью утилиты DD.
11. Сотрите таблицу разделов MBR с помощью утилиты DD.
12. Восстановите MBR с помощью утилиты DD.
13. Смонтируйте разделы и проверьте целостность данных.
14. Отмонтируйте разделы.
15. Установите gdisk <sudo apt-get install gdisk>
16. Создайте таблицу разделов GPT (5 первичных разделов) с помощью gdisk.
17. Отформатируйте созданные разделы в файловую систему ext3.
18. Смонтируйте созданные разделы и создайте там произвольные файлы.
19. Сделайте резервную копию GPT с помощью утилиты DD, предварительно определив необходимое количество байт для резервной копии.
20. Сотрите GPT с помощью утилиты DD.
21. Восстановите GPT с помощью утилиты DD.
22. Смонтируйте разделы и проверьте целостность данных.
23. Отмонтируйте разделы.
24. Определите достоинства и недостатки таблиц разделов MBR и GPT.

Вопросы к практической работе

1. Что записано в первом секторе главной загрузочной записи MBR ?
2. Функциональное назначение MBR и GPT ?
3. Структура GPT.
4. Какое максимальное количество первичных разделов можно создать при использовании таблицы разделов MBR ?
5. Какое максимальное количество первичных разделов можно создать при использовании таблицы разделов GPT ?
6. Как сохранить информацию о структуре MBR ?
7. Как создать 10 разделов с файловой системой ext3 на диске в таблице разделов MBR ?
8. Как стереть код загрузчика в MBR ?
9. Как можно смонтировать раздел диска с файловой системой в режиме только для чтения?
10. Как можно осуществить восстановление GPT разделов в случае сбоев?

Шкала оценивания:

Оценка «отлично» за практическую работу выставляется, если работа выполнена полностью правильно и самостоятельно, а аспирант способен уверенно отвечать на вопросы преподавателя о проделанной работе;

Оценка «хорошо» выставляется, если работа выполнена в целом корректна, но имеются неточности в ответах на вопросе преподавателя или некоторые действия выполнены неоптимально;

Оценка «удовлетворительно» выставляется, когда работа не может быть завершена аспирантом самостоятельно, без помощи преподавателя.

Оценка «неудовлетворительно» выставляется, если работа не была выполнена.

Практическое занятие 5. Облачные вычисления

Тематика: Виртуальная машина.

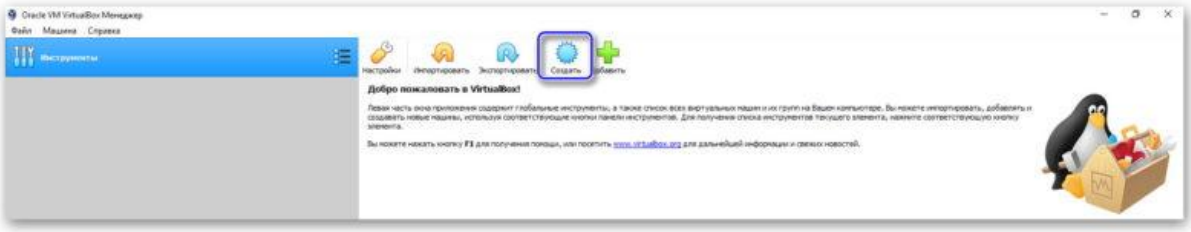
Текст задания: Какие возможности предоставляет виртуальная машина Ubuntu?

- Настройки печати старых принтеров. Например, HP1100 настолько стар, что не работает в Windows. А вот виртуальная Ubuntu работает с ним без проблем и можно настроить печать между двумя машинами (это сложная процедура, которая требует досконального изучения);
- Тестирование ПО. Некоторые программы, которые разрабатываются для нескольких платформ зачастую тестируются в виртуальных машинах;
- Проверка файлов и восстановление съёмных носителей. Очень часто через Ubuntu поднимаются так называемые «мёртвые флешки», которые не определяются в Windows. Также, можно проверять содержимое архивов с большим числом расширений.

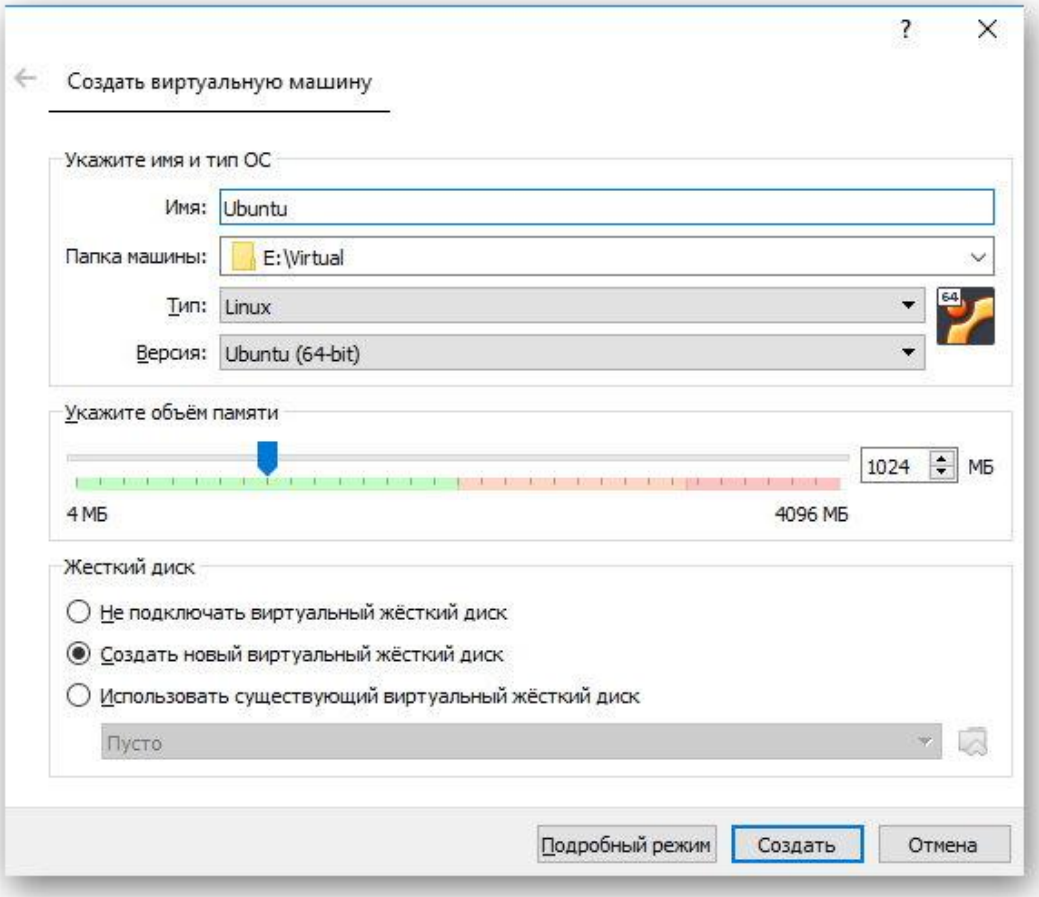
Системные требования Ubuntu включают в себя двухъядерный процессор с частотой 2 ГГц, 2 Гб ОЗУ и 25 Гб свободного места на диске. Данные требования гарантируют комфортную работу в системе, однако систему можно установить и на компьютер с более низкими характеристиками.

Ход работы:

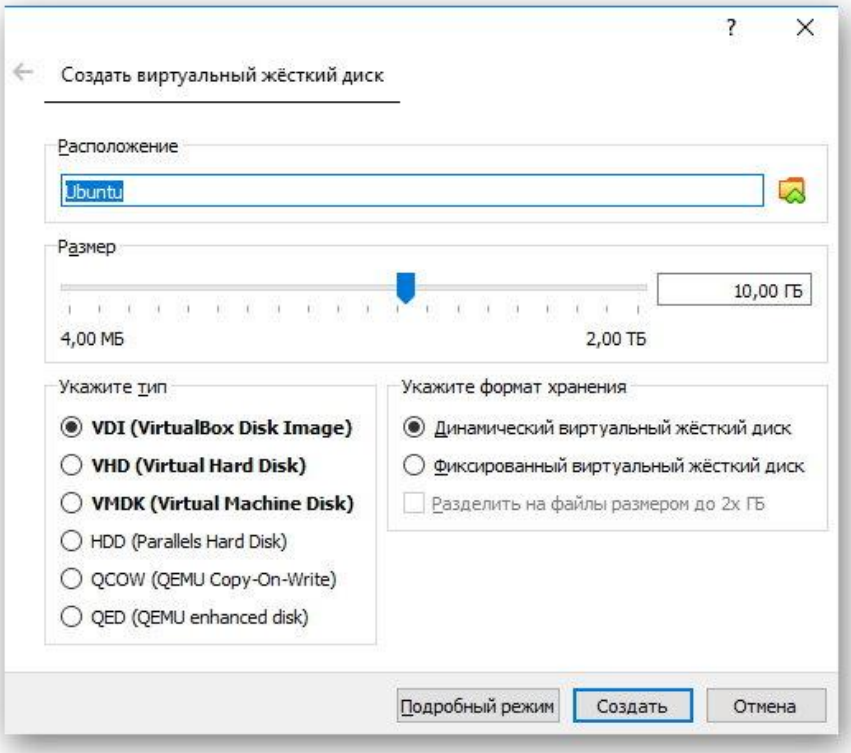
1. Запустите VirtualBox и нажмите «Создать».



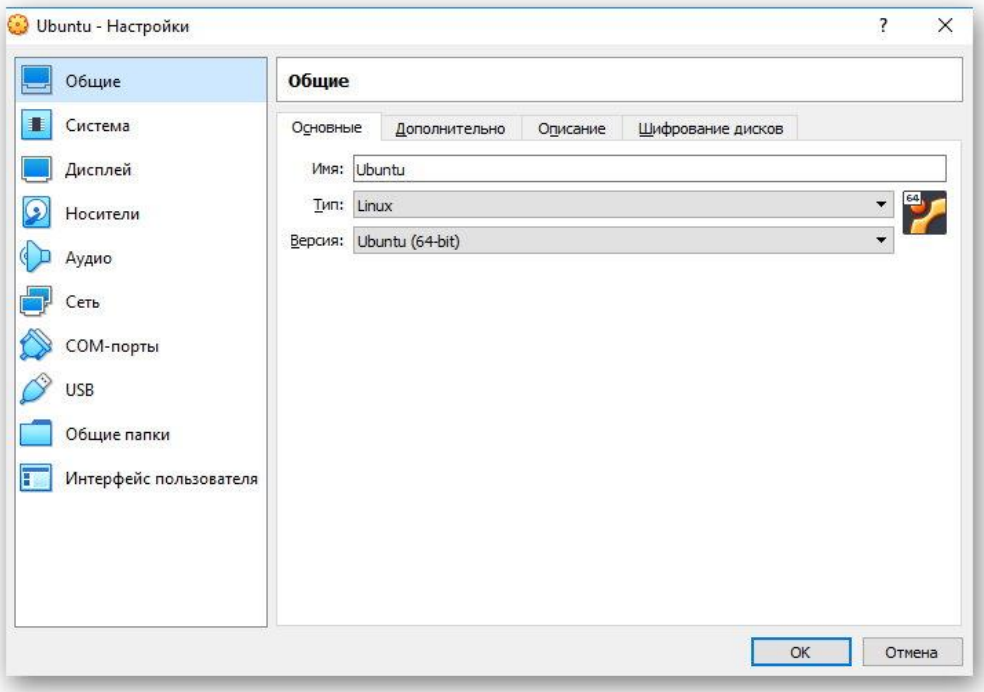
2. Задайте будущей машине: «Имя», «Путь хранения», «Тип ОС», и версию ОС. Фактически нет разницы устанавливать на virtualbox Ubuntu04 или 12.02. Тип и версия (семейство) останутся без изменений. Также, выделите рекомендуем объём ОЗУ.



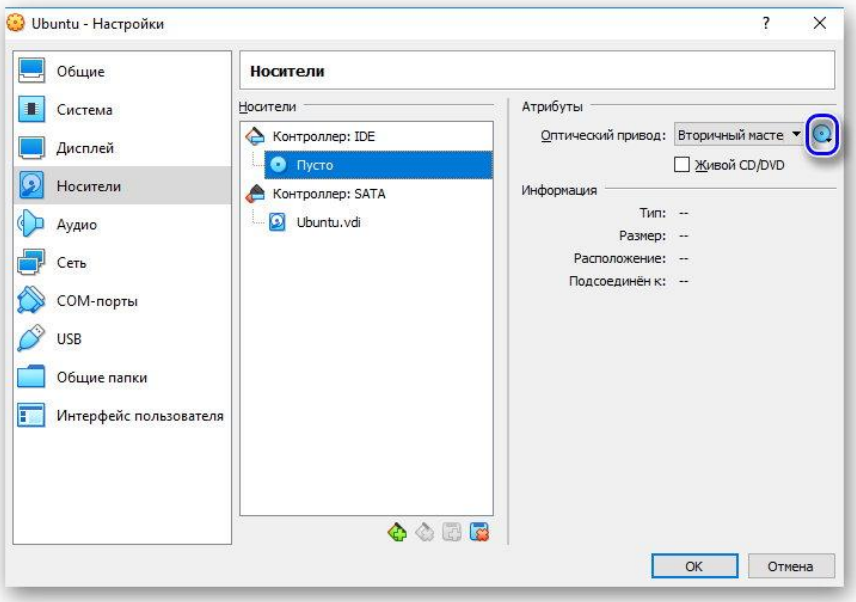
3. В пункте 2 вы выставили «Создать новый виртуальный диск». Сейчас потребуется указать его размер. Тип форматирования и остальные параметры зависят от выбранного семейства ОС.



4. Машина создана, но подготовка ещё не завершена. Выбираем «Настройки».



5. В графе «Носители» выбираем «Оптический привод» и нажимаем «Добавить образ». Таким образом будет смонтирован виртуальный диск из образа, который качался ранее.

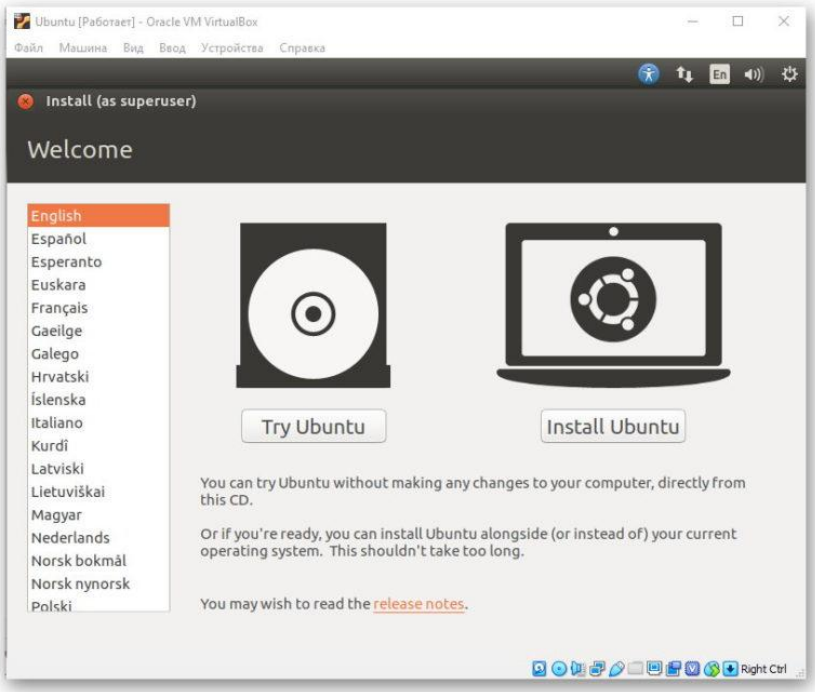


6. Указываем путь к образу Ubuntu.

Теперь подготовка завершена. Остаётся только запустить виртуальную машину, нажав «Запустить».

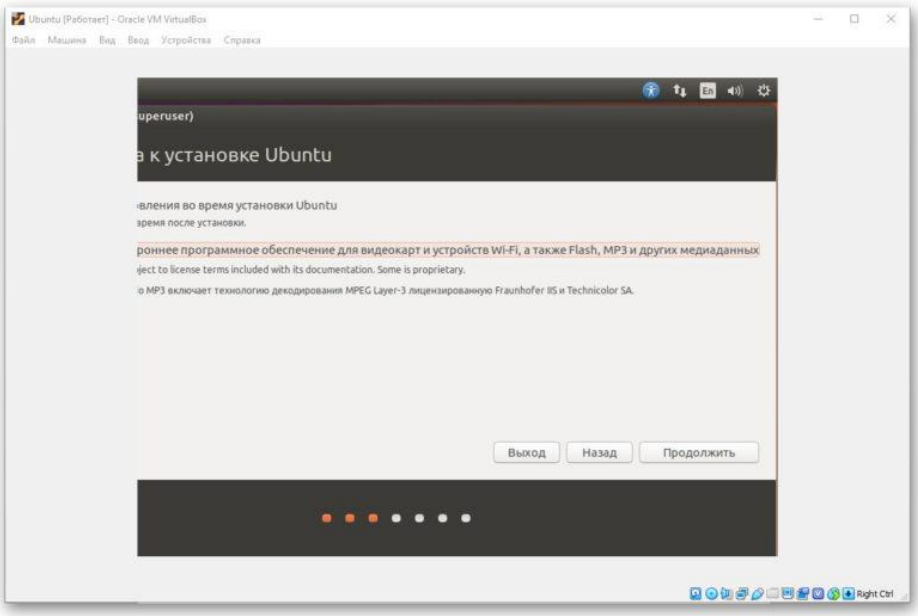
Установка системы:

1. Выбрать язык, который будет использоваться. Данный выбор делается не только для процесса установки, ног и самой ОС. Кстати, вместо установки можно выбрать «Предпросмотр» и ознакомиться с функционалом ОС.

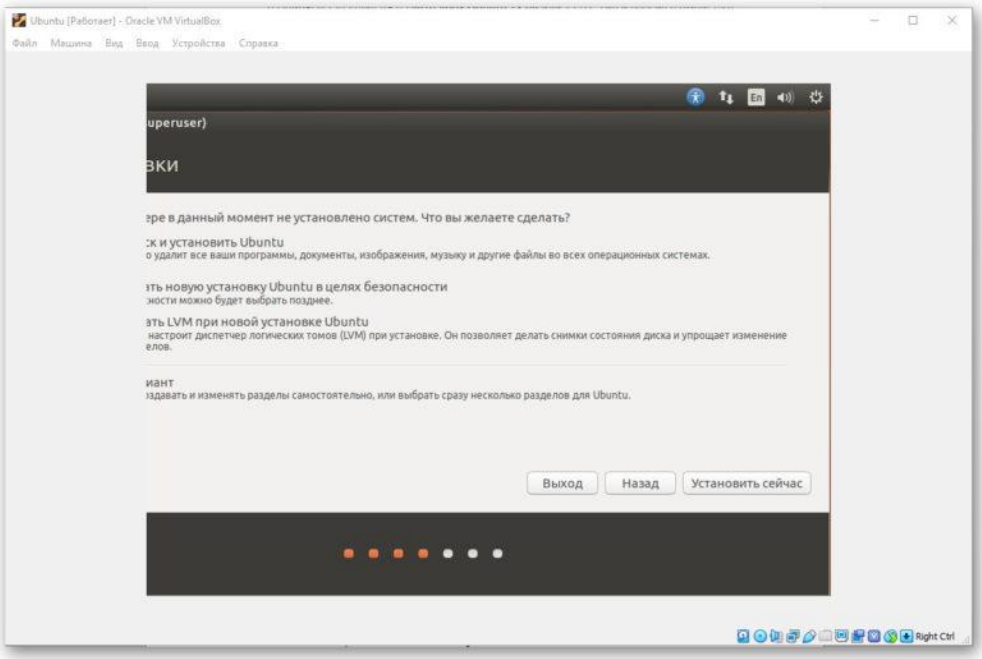


2. Нажимаем «Установить Ubuntu».

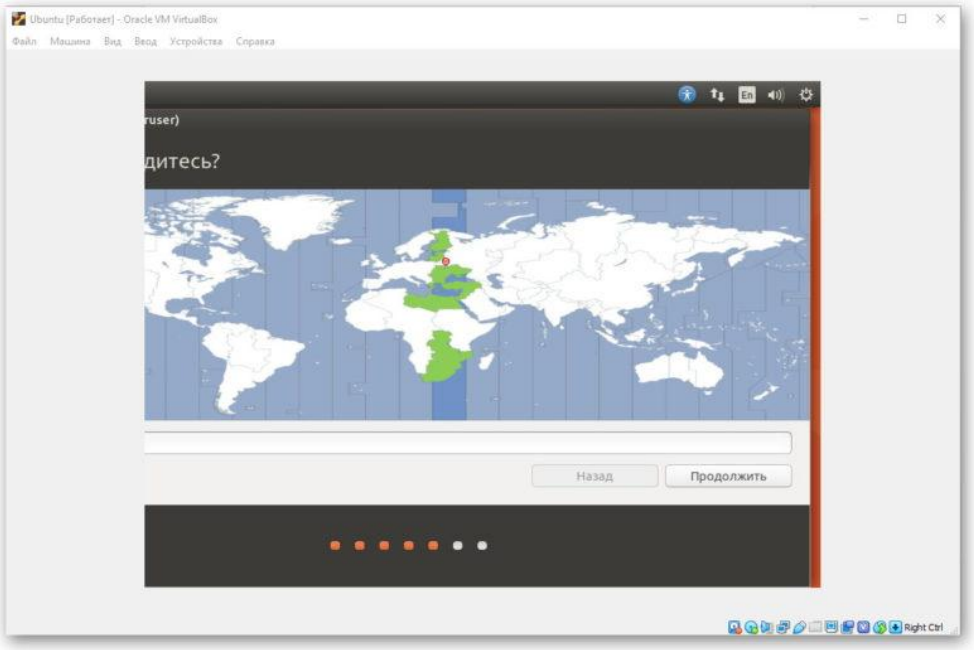
3. Отмечаете необходимые галочки (например, обновления для ОС) и нажимайте «Продолжить». Если кнопку продолжить не видно, то можно потянуть мышью само окно. Увы, VirtualBox подхватывает размер экрана и передаёт его виртуальной машине. На куске в 800x600 будет отображаться родное расширение.



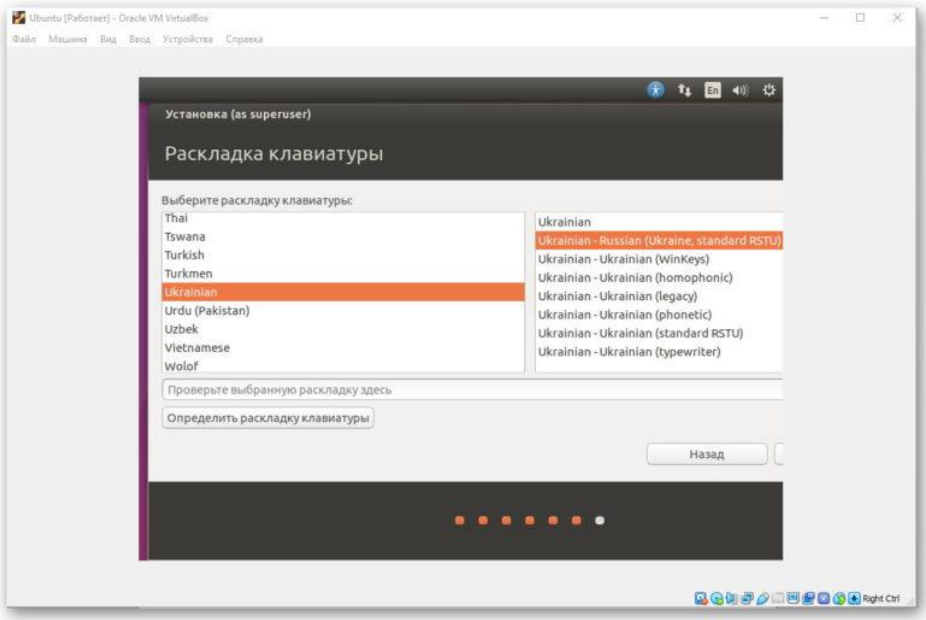
4. Поскольку используется виртуальный диск, то можно смело пропустить шаг с выбором настройки. Установщик ОС принимает выделенную виртуальную область за реальный диск. В пределах Windows ничего стёрто не будет. Нажимайте «Установить сейчас» и подтвердите изменения.



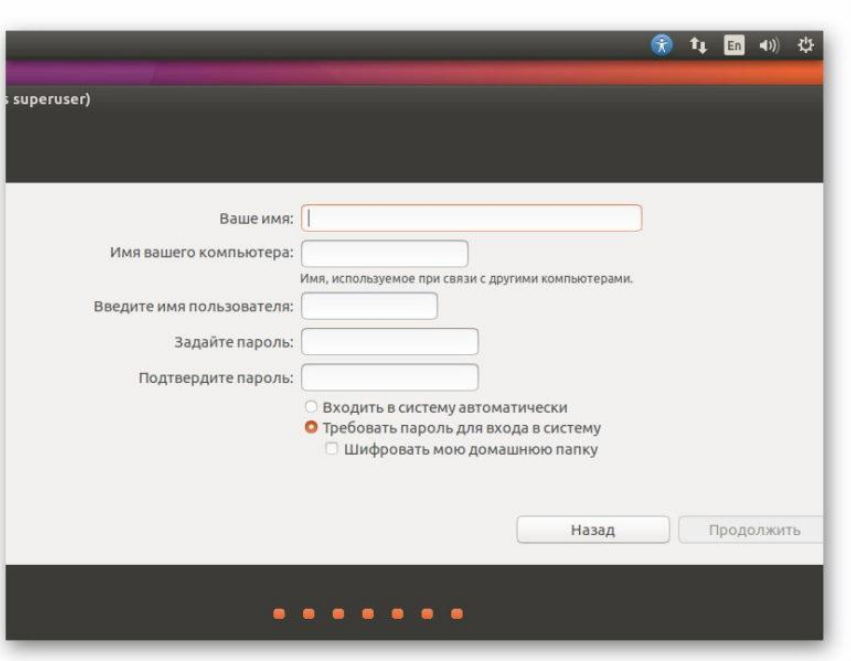
5. Выберите регион. Это потребуется для настроек часового пояса и перехода на летнее/зимнее время.



6. Добавьте основную раскладку клавиатуры.



7. Задайте настройки пользователя. Учётную запись и пароль (будет паролем суперпользователя). Дальше установка не требует от нас никаких действий.



По окончанию установки продемонстрируйте рабочую систему преподавателю.

Шкала оценивания:

Оценка «отлично» за практическую работу выставляется, если работа выполнена полностью правильно и самостоятельно, а аспирант способен уверенно отвечать на вопросы преподавателя о проделанной работе;

Оценка «хорошо» выставляется, если работа выполнена в целом корректна, но имеются неточности в ответах на вопросе преподавателя или некоторые действия выполнены неоптимально;

Оценка «удовлетворительно» выставляется, когда работа не может быть завершена аспирантом самостоятельно, без помощи преподавателя.

Оценка «неудовлетворительно» выставляется, если работа не была выполнена.

Практическое занятие 6. Организация баз данных и знаний

Тематика: Основные понятия структурной части реляционной модели данных.

Текст задания:

1. Создать базу данных «Фирма», представленную отношениями:
   - Филиалы (Филиал, Страна_Филиала),
   - Заказы (Филиал, Заказчик, № Заказа),
   - Товары (№ заказа, Товар, Количество, Цена),
   - Клиенты (Заказчик, Страна_Заказчика).

Обозначить ключевые поля, поля связи, тип связи.

2. Создать базу данных «Банк», представленную отношениями:
   - Клиент (ФИО клиента, № филиала, № счета, Остаток, Кредит),
   - Филиал (№ Филиала, Район)

Обозначить ключевые поля, поля связи, тип связи.

3. Создать базу данных «Программные комплексы», представленную отношениями:
   - Файлы (Файл, Сотрудник),
   - ПК_Файлы (Код_ПК, Файл),

- Программные комплексы (Код_ПК, Название ПК),
- Сотрудники (Сотрудник, Отдел)

Обозначить ключевые поля, поля связи, тип связи.
4. Открыть БД Банк. Составить запрос: Клиентов, которые имеют счета с остатком, превышающим 10 000 руб.
5. Открыть БД Банк. Составить запрос: Клиентов, которые имеют по одному счету
6. Открыть БД Банк. Составить запрос: Выбрать клиентов, имеющих на счете до 200руб.
7. Открыть БД Банк. Составить запрос: Выбрать клиентов, имеющих на счете 0 руб.
8. Открыть БД Банк. Составить запрос: Выбрать клиентов из Салавата и Ишимбая
9. Открыть БД Банк. Составить запрос: Выбрать клиентов, у которых сумма кредита больше суммы остатка
10. Открыть БД Банк. Составить запрос: Клиентов, у которых сумма кредита не превышает сумму остатка более чем в два раза
11. Открыть БД Банк. Составить запрос: Клиентов, которые имеют на счету остаток ноль.
12. Открыть БД Банк. Составить запрос: Клиентов, которые имеют несколько счетов
13. Открыть БД Банк. Составить запрос: Филиалы с общей суммой выданного кредита
14. Открыть БД Банк. Составить запрос: Клиентов, у которых сумма кредита не превышает 50% от суммы остатка
15. Открыть БД Банк. Составить запрос: Клиентов, обслуживающихся в филиале №3
16. Открыть БД Банк. Составить запрос: Клиентов с общей суммой полученного кредита
17. Открыть БД «Сотрудники». Составить запрос: «Должности, занимаемые сотрудниками». Поле «ФИО» в запросе отсортировать по возрастанию
18. Открыть БД «Сотрудники». Составить запрос: «Работающие, свыше 7 лет», поле - стаж - не выводить на экран
19. Открыть БД «Сотрудники». Составить запрос: «Сложность работ» (сложность выполняемых работ – по возрастанию)
20. Открыть БД «Сотрудники». Составить запрос: «Заработная плата сотрудников». Поле «Зарплата» отсортировать по убыванию. Вывести в запросе поля: ФИО, Должность, Зарплата. В запросе не выводить поля, из которых формируется заработная плата
21. Открыть БД «Сотрудники». Составить запрос: «Работающие в должности…» (запрос с параметром). Вывести в запросе поля: ФИО, Должность.
22. Открыть БД «Сотрудники». Составить запрос: «Сотрудники – мужчины 1978 и 1979 г. рождения». Вывести в запросе поля: ФИО, Должность. Поле «дата рождения» - не выводить.
23. Открыть БД «Сотрудники». Составить запрос: «Средняя заработная плата»
24. Открыть БД «Сотрудники». Составить запрос: «Минимальная заработная плата»

Шкала оценивания:

Оценка «отлично» за практическую работу выставляется, если было выполнено не меньше, чем 85% всех представленных заданий.

Оценка «хорошо» за практическую работу выставляется, если было выполнено не меньше, чем 71% всех представленных заданий.

Оценка «удовлетворительно» за практическую работу выставляется, если было выполнено не меньше, чем 61% всех представленных заданий.

Оценка «неудовлетворительно» за практическую работу выставляется, если было выполнено менее 61% всех представленных заданий.

Практическое занятие 7. Искусственные нейронные сети

Текст задания №1: Вам дан датасет с предложениями на разных языках Европы: по 15000 предложений на каждом из 20 языков из списка: bg,cs,da,de,el,es,et,fi,fr,hu,it,lt,lv,nl,pl,pt,ro,sk,sl,sv.

Цель: научиться распознавать по слову его язык.

Ссылка на данные: `https://disk.yandex.ru/d/yTryI8ySINOjEA`

В папке 20 файлов вида train_language на разных языках по 15000 строк.

Файл test — 12000 пронумерованных слов, у которых нужно определить язык.

Кодировка всех файлов — Юникод. На языке python можно открывать файлы как `open('filename', 'r', encoding='utf-8')`.

Текст задания №2: На острове живут n человек. Каждый из них либо рыцарь, либо лжец. Все рыцари всегда говорят правду, все лжецы всегда лгут. Каждый человек сделал по одному утверждению. i-тый островитянин ($1 \le i \le n$) сказал: "Среди всех людей на острове не более k_i рыцарей". По заданным утверждениям островитян найдите любое возможное количество рыцарей на острове или определите, что такая ситуация невозможна.

Текст задания №3: Петя и Вася сгенерировали граф с n вершинами следующим образом. Петя случайным образом разбил вершины на две группы и пометил их метками 1 и 2. Каждую вершину он помечал единицей с вероятностью 0.5 независимо от остальных, а остальные пометил двойками. После разбиения на группы Петей Вася начал проводить ребра: любые две вершины из разных групп Вася соединял с вероятностью 0.9 независимо от остальных, а две различные вершины из одной группы он соединял с вероятностью 0.1 тоже независимо от остальных. После этого метки вершин стерлись, и ни Петя, ни Вася не помнят их. Помогите им восстановить метки вершин по графу. При этом важно лишь разбиение вершин на группы, то есть можно все вершины с меткой 1 пометить двойками, а все вершины с меткой 2 пометить единицами.

Текст задания №4: Для получения данных о поверхности морского дна команда учёных отправила в море исследовательского робота. К сожалению, робот разрядился, находясь в море, но в последний момент успел послать свои координаты (x,y). Теперь каждую минуту его сносит течением на расстояние 1 в случайную сторону. А именно, происходит одно из 4 событий, каждое с вероятностью 0.25: координата x увеличивается на 1, координата y увеличивается на 1, координата x уменьшается на 1, координата y уменьшается на 1. На его поиски был отправлен летающий дроид, который за одну минуту может ровно k раз изменить одну из своих координат на 1. Дроид начинает свои поиски в точке (0,0). Если дроид оказывается от робота на расстоянии не более чем 2 по обеим координатам, то он его замечает и тем самым выполняет свою миссию. Формально, если в некоторую минуту координаты робота (x1,y1), а координаты дроида (x2,y2), то дроид замечает робота, если $\max(|x1-x2|,|y1-y2|) \le 2$. Считается, что каждую минуту сначала дроид выполняет свои k команд и только после этого робота сносит течением. Напишите программу, которая позволит дроиду разыскать разрядившегося робота за 1000 минут. К сожалению, любая программа при некоторых x и y с некоторой вероятностью не найдет робота. Поэтому вашей программе в каждом наборе данных будут даны 100 тестов, из которых она должна правильно решить как минимум 80. В каждом из тестов траектория робота будет фиксированной, но выбранной независимо от других из описанного в условии распределения.

Текст задания №5: У Пети есть n квадратных трехчленов вида $a_2x^2+a_1x+a_0$. Вася хочет отгадать хотя бы один из них. Он может делать запрос про точку x, и Петя скажет значение произвольного из своих n трехчленов в этой точке, причем не скажет, какого именно. Напишите программу, которая поможет Васе отгадать хотя бы один трехчлен, задав не более чем 100 вопросов.

Шкала оценивания:

Оценка «отлично» за практическую работу выставляется, если было выполнено не меньше, чем 85% всех представленных заданий.

Оценка «хорошо» за практическую работу выставляется, если было выполнено не меньше, чем 71% всех представленных заданий.

Оценка «удовлетворительно» за практическую работу выставляется, если было выполнено не меньше, чем 61% всех представленных заданий.

Оценка «неудовлетворительно» за практическую работу выставляется, если было выполнено менее 61% всех представленных заданий.

Практическое занятие 8. Программная инженерия

Тематика: Технологии создания ПО.

Текст задания:

Графический пользовательский интерфейс GUI (Graphical User Interface) – это разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений. В отличие от интерфейса командной строки (типа старого DOS системы Windows), в GUI пользователь имеет свободный доступ ко всем элементам интерфейса экрана и может ими управлять с помощью клавиатуры или мыши.

Библиотека tkinter (от англ. Tk interface) – это встроенная графическая библиотека на основе средств Tk, широко распространенная в мире GNU/Linux и других UNIX-подобных систем, которая портирована (переведена), в частности, на Microsoft Windows и Apple Mac OS). Tk (от англ. Toolkit – «набор инструментов», «инструментарий») – кроссплатформенная библиотека базовых элементов графического интерфейса, распространяемая с открытыми исходными текстами.

Библиотека tkinter входит в стандартный дистрибутив языка Python (до версии Python 3.0 именовалась Tkinter). В tkinter стандартизированные компоненты графического интерфейса, с которыми взаимодействует пользователь, называются виджетами (widget, от англ. window gadget).

Создание окна

Для подключения модуля tkinter используется форма

```
import tkinter from *
```

позволяющая использовать все имена модуля без указания префикса tkinter.

Tk() является базовым классом любого tkinter приложения. При создании объекта этого класса запускается интерпретатор tcl/tk и создается виджетами наивысшего уровня. Обычно им является основное окно приложения. tkinter является событийно-ориентированной библиотекой. В приложениях такого типа имеется главный цикл обработки событий. В tkinter такой цикл запускается методом mainloop(). Для явного выхода из интерпретатора и завершения цикла обработки событий используется метод quit().

Таким образом минимальное приложение на Tkinter будет таким:

```
import tkinter from *
root = Tk()
root.mainloop()
```

Создание окна, рамки и кнопок

В качестве первого примера создания пользовательского приложения с использоваением графической библиотеки tkinter рассмотрим программу gui1.py, в которой создается пользовательское окно, а в нем – рамка с тремя кнопками:

```
from tkinter import *
class My_frame (Frame):
    ''' Рамка с тремя кнопками '''
    def __init__ (self,master):
        super ().__init__(master)
        self.grid()
        self.create_widgets()
    def create_widgets (self):
        ''' Создает три кнопки '''
        self.bt1=Button(self, text='Кнопка №1', fg='blue',
                        font='12' )
        self.bt1.grid()
        self.bt2=Button(self)
        self.bt2.grid()
        self.bt2['text']= 'Кнопка №2'
        self.bt2['fg']= 'blue'
        self.bt2['font']= '12'
        self.bt3=Button(self)
        self.bt3.grid()
        self.bt3.config(text='Кнопка №3', fg='blue',
                        font='12' )

root=Tk() #Создание базового окна
root.title( 'Рамка с кнопками' )
root.geometry( '230x95+800+200' )
app=My_frame(root)
root.mainloop()
```

В программе gui1.py подключение модуля tkinter осуществляется с использованием формы from tkinter import *. При этом имена всех классов и методов модуля помещаются в пространство имен программы и обращение к ним происходит без добавления префикса tkinter.

На базе класса Frame модуля tkinter создается пользовательский класс My_frame, который описывает рамку, содержащую три кнопки. При создании конструктора этого класса используется встроенная функция super(), которая предназначена для наследования методов родительского класса при их перегрузке. Таким образом, к методам конструктора класса Frame() объекта master добавляются методы: grid() – для размещения рамки, и create_widgets() – для создания кнопок в рамке.

Кнопки в виде объектов bt1, bt2 и bt3 создаются с использованием класса Button модуля tkinter. Хотя кнопки создаются в этом примере одинаковые, способы задания параметров кнопки: text – указывающего название кнопки, fg и font – задающих соответственно цвет (blue) и размеры (12 пикселей) символов, используются разные:

- для кнопки №1 – непосредственно при ее создании;
- для кнопки №2 – задаются с помощью ключей словаря экземпляра класса Button;
- для кнопки №3 – с помощью метода config().

После описания класса в программе gui1.py:

- создается пользовательское окно, обычно именуемое root;

- указываются имя этого окна – 'Рамка с кнопками' и его размеры и расположение на экране:
  - длина – 230 пикселей;
  - ширина – 85 пикселей;
  - смещение по горизонтали – 800 пикселей;
  - смещение по вертикали – 200 пикселей;
- в этом окне создается рамка, свойства которой описаны в классе My_frame();
- запускается цикл обработки событий (хотя в этом примере события не обрабатываются).

Обработка событий

В качестве примера обработки событий, т.е. действий пользователя над виджетами, приведена программа gui2.py, в которой также, как и в программе gui1.py, создается пользовательский класс My_frame2, описывающий рамку, которая, однако, имеет следующие отличия:

- содержит только одну кнопку
- для этой кнопки кроме параметра text, указывающего ее начальное название, задаются также следующие параметры:
  - command – указывает метод – обработчик события для кнопки (self.count);
  - width – задает длину кнопки (30);
  - bg – задает цвет фона (#efffef');
  - fg – задает цвет символов;(blue)
  - font – задает размер символов (6mm);
- содержит метод count(), который по каждому щелчку по кнопке увеличивает значение переменной bt_clicks на единицу и изменяет в названии кнопки число выполненных по ней щелчков.

```
from tkinter import *
class My_frame2 (Frame):
    '''Рамка с кнопкой, подсчитывающей события '''
    def __init__ (self,master):
        super ().__init__(master)
        self.grid()
        self.bt_clicks=0
        self.create_widgets()
    def create_widgets (self):
        ''' Создает кнопку '''
        self.bt1=Button(self,text= 'Число щелчков: 0' ,
                        command=self.count,
                        width= '20' ,
                        height= '2' ,
                        bg= '#efffef' ,
                        fg= 'blue' ,
                        font= 'arial 14' )
        self.bt1.grid()
    def count (self):
        self.bt_clicks+=1
        self.bt1['text'] = 'Число щелчков: ' \
                           + str (self.bt_clicks)
```

```
root=Tk()
root.title( 'Обработка событий' )
root.geometry( '240x60+800+300' )
app2=My_frame2(root)
root.mainloop()
```

Работа с текстом

```
from tkinter import *
class My_frame3 (Frame):
    ''' Рамка для работы с текстом '''
    def __init__ (self,master):
        super ().__init__(master)
        self.grid()
        self.create_widgets()
    def create_widgets(self):
        ''' Создает 2 метки, текстовое поле,
            текстовую область и кнопку '''
        #метка-инструкция
        self.ins_lb=Label(self,
                          text='Как прожить до 100 лет?',
                          fg= 'blue' , font= 'arial 14' )
        self.ins_lb.grid(row=0, column=0,
                         columnspan=2,sticky=W)
        #метка-пароль
        self.ps_lb=Label(self, text= 'Пароль',
                         fg='blue' , font='arial 10' )
        self.ps_lb.grid(row=1, column=0, sticky=W)
        #текстовое поле для ввода пароля
        self.ps_ent=Entry(self, bg='#f0f0ff',
                          font='arial 10' )
        self.ps_ent.grid(row=1, column=1, sticky=E)
        #создание текстовой области, куда будет помещен ответ
        self.txt=Text(self, width=30, height=4, wrap=WORD,
                      bg='#f0f0ff' , font='arial 10' )
        self.txt.grid(row=2, column=0, columnspan=2)
        #кнопка отправки сообщения
        self.bt=Button(self, text='Узнать секрет', fg='blue',
                       bd='5' , font='arial 12',
                       command=self.reveal)
        self.bt.grid(row=3, column=0, sticky=W)
    def reveal(self):
        contents=self.ps_ent.get()
        if (contents== 'secret'):
            message='Чтобы прожить до 100 лет, надо ' \
            'сначала прожить 99 лет и 11 месяцев, а потом ' \
            'вести себя очень-очень осторожно.'
        else:
            message= 'Вы ввели неправильный пароль.'
        self.secret_txt.delete(0.0,END)
```

```
            self.secret_txt.insert(0.0,message)

root=Tk()
root.title( 'Работа с текстом' )
root.geometry( ''250x160+850+300' )
app3=My_frame3(root)
root.mainloop()
```

В программе gui3.py описан класс My_frame3, в котором с помощью метода create_widgets() создаются следующие виджеты:

- неизменяемые сообщения (метки) на базе класса Label:
  - 'Чтобы узнать секрет долголетия введите пароль' (объект ins_lb);
  - 'Пароль' (объект ps_lb);
- текстовое поле для ввода пароля на базе класса Entry (объект ps_ent);
- кнопка с сообщением (надписью) 'Узнать секрет' на базе класса Button (объект submit_bt);
- текстовая область для помещения ответа на базе класса Text (объект secret_txt), которая имеет следующие параметры:
  - ширина (width) – 35 символов;
  - высота (height) – 5 рядков;
  - перенос по границе слева (WORD))
- Помимо метода create_widgets() в классе My_frame3 также описан обработчик щелчка по кнопке 'Узнать секрет' – метод reveal(), который:
  - получает с помощью метода get() из текстового поля ps_ent введенный пользователем пароль;
  - сравнивает его с паролем 'secret' и формирует сообщение, соответствующее результату сравнения: либо приводится "секрет долголения", либо сообщается о том, что пароль введен неправильно;
  - очищает текстовую область ответа (secret_txt);
  - помещает туда сформированное сообщение.

Отметим использование в этой программе метода grid(), который осуществляет размещение (упаковку) графических элементов на рамке. В отличие от программ gui1.py и gui2.py, где метод grid() использовался без параметров и, следовательно, виджеты располагались по усматрению интерпретатора, в программе gui3.py явным образом указано расположение элементов. При использовании метода grid() поле рамки изображается в виде решетки, состоящей из рядков и колонок. Для указания местоположения виджета использованы следующие параметры:

- row – указывает номер рядка;
- column – указывает номер колонки;
- columnspan – указывает размер ячейки в колонках (по умолчанию – 1);
- sticky – указывает выравнивание данных относительно границ ячейки:
  - N – по верхней границе;
  - E – по правой границе;
  - S – по нижней границе;
  - W – по левой границе.

Использование флажков

```
from tkinter import *
class My_frame4 (Frame):
```

```
    ''' Использование флажков '''
    def __init__ (self,master):
        super (My_frame4, self).__init__(master)
        self.grid()
        self.create_widgets()
    def create_widgets (self):
        ''' Создает три флажка '''
        #метка-описание
        Label(self, text='Укажите любимые телепередачи',
              font='arial 13' , fg='blue'
              ).grid(row=0, column=0,sticky=W)
        #метка-инструкция
        Label(self, text='Выберете все, что нравится',
              font='arial 11' , fg='blue'
              ).grid(row=1, column=0,sticky=W)
        #флажок 'Футбол'
        self.likes_football=BooleanVar()
        Checkbutton(self, text='Футбол',
                    variable=self.likes_football,
                    command=self.update_text
                    ).grid(row=2, column=0,sticky=W)
        #флажок 'История'
        self.likes_history=BooleanVar()
        Checkbutton(self, text='История',
                    variable=self.likes_history,
                    command=self.update_text
                    ).grid(row=3, column=0,sticky=W)
        #флажок 'Природа'
        self.likes_nature=BooleanVar()
        Checkbutton(self, text='Природа',
                    variable=self.likes_nature,
                    command=self.update_text
                    ).grid(row=4, column=0,sticky=W)
        #текстовая область с результатом
        self.results_txt=Text(self, fg='blue',
                              font='arial 11', width=40,
                              height=5, wrap=WORD)
        self.results_txt.grid(row=5, column=0, columnspan=3)
    def update_text (self):
        likes= ''
        if self.likes_football.get():
            likes += 'Вам нравится футбол\n'
        if self.likes_history.get():
            likes += 'Вас привлекает история\n'
        if self.likes_nature.get():
            likes += 'Вам небезразлична природа\n'
        self.results_txt.delete(0.0,END)
        self.results_txt.insert(0.0,likes)

root=Tk()
```

```
root.title( 'Использование флажков' )
root.geometry( '270x185+800+200' )
app4=My_frame4(root)
root.mainloop()
```

В отличие от программ gui1.py, gui2.py и gui3.py в программе gui4.py при создании объектов меток (класс Label) используется более короткая форма – без указания имен объектов. Это допускается делать в тех случаях, когда имена этих объектов в дальнейшем не используются в описании класса.

Другой особенностью программы gui4.py является использование флажков – графических элементов класса Checkbutton, при помощи которых осуществляется выбор (от одной до трех) телепередач:

- о футболе;
- об истории;
- о природе.

При этом для каждого флажка:

- указывается переменная класса BooleanVar, соответственно:
  - likes_football;
  - likes_history;
  - likes_nature.
- создается с использованием короткой формы (т.е. без указания имени) объект класса Checkbutton и задаются следующие значения его параметров:
  - для text – наименование флажка (соответственно, "Футбол", "История" и "Природа");
  - для command – наименование функции (update_text), которая обрабатывает события, связанные с изменением состояния флажка (т.е. выбран флажок или нет);
  - для variable, отражающего текущее состояние флажка, – имя переменной, с которой этот параметр связывается(соответственно likes_football, likes_history или likes_nature).

Обработчик update_text() выполняет следующие действия:

- последовательно анализирует состояние переменных likes_football, likes_history и likes_nature и в зависимости от значения переменной (True или False) добавляет или не добавляет к переменной likes значение, соответственно, 'Вам нравится футбол', 'Вас привлекает история' или 'Вам небезразлична природа';
- очищает текстовое поле results_txt;
- выводит в текстовое поле results_txt значение переменной likes.

При выполнении программы gui4.pyw на экране появляется окно со следующими виджетами:

- меткой-описанием "Укажите любимые телепередачи";
- меткой-инструкцией "Выберете все, что нравится";
- флажками:
  - "Футбол";
  - "История";
  - "Природа";
- текстовой областью для вывода результатов выбора.

Использование переключателей

```
from tkinter import *
```

```
class My_frame5 (Frame):
    ''' Использование переключателей '''
    def __init__ (self,master):
        super (My_frame5, self).__init__(master)
        self.grid()
        self.create_widgets()
    def create_widgets (self):
        ''' Создает переключатель '''
        #метка-инструкция
        Label(self, text='Укажите любимую телепередачу',
              fg= 'blue' , font= 'arial 13'
              ).grid(row=0, column=0,sticky=W)
        self.favorite=StringVar()
        self.favorite.set( None )
        #положение переключателя 'Футбол'
        Radiobutton(self, text='Футбол',
                    variable=self.favorite,
                    value= 'о футболе',
                    command=self.update_text
                    ).grid(row=1, column=0,sticky=W)
        #положение переключателя 'История'
        Radiobutton(self, text= 'История',
                    variable=self.favorite,
                    value= 'об истории',
                    command=self.update_text
                    ).grid(row=2, column=0,sticky=W)
        #положение переключателя 'Природа'
        Radiobutton(self, text= 'Природа',
                    variable=self.favorite,
                    value= 'о природе',
                    command=self.update_text
                    ).grid(row=3, column=0,sticky=W)
        #текстовая область для результата
        self.results_txt=Text(self, fg='blue',
                              font='arial 10', width=40,
                              height=5, wrap=WORD)
        self.results_txt.grid(row=4, column=0, columnspan=3)
    def update_text (self):
        message= "Ваша любимая телепередача - "
        message += self.favorite.get()
        self.results_txt.delete(0.0, END)
        self.results_txt.insert(0.0, message)

root=Tk()
root.title( 'Использование переключателя' )
root.geometry( '310x120+800+200' )
app=My_frame5(root)
root.mainloop()
```

В отличие от программы gui4.py, в которой может осуществляться выбор нескольких телепередач, в программе gui5.py можно выбрать только одну телепередачу. Поэтому вместо трех флажков класса Checkbutton используется один переключатель с тремя положениями, наименования которых соответствуют наименованиям флажков.

При этом выполняются следующие действия:

- указывается атрибут self.favorite класса StringVar для хранения наименования выбранного положения переключателя (при его создании указывается значение None);
- для каждого положения переключателя создается объект класса Radiobutton и указываются значения его параметров:
    - для text – наименование положения переключателя (соответственно, "Футбол", "История" и "Природа");
    - для command – наименование функции (update_text), которая обрабатывает события, связанные с изменением состояния переключателя;
    - для variable, отражающего текущее состояние переключателя, – self.favorite (имя атрибута, с которыи этот параметр связывается;
    - для value – значение выбранного положения переключателя, соответственно, 'о футболе', 'об истории' или 'о природе'.

Обработчик update_text() выполняет следующие действия:

- задает начальное значение переменной message ("Ваша любимая телепередача - ");
- используя атрибут self.favorite и метод get() получает значение выбранного положения переключателя и добавляет его к значению переменной message;
- очищает текстовое поле results_txt;
- выводит в текстовое поле results_txt значение переменной message.

При выполнении программы IE: gui5.pyw на экране появляется окно со следующими виджетами:

- меткой-инструкцией "Укажите любимую телепередачу";
- переключателем, имеющем три состояния:
    - "Футбол";
    - "История";
    - "Природа";
- текстовой областью для вывода результата выбора.

Индивидуальные задания

Разработать программу на языке Python, в которой на базе модуля tkinter разработан графический интерфейс, имеющий несколько окон:

- окно с кнопкой (объект Button), надпись на которой меняется по каждому щелчку и содержит следующую информацию – номер щелчка и число, которое определяется согласно колонке "Число" табл. №1:
    - 1 – очередное простое число (начиная с 1);
    - 2 – очередное число Фибоначчи (начиная с 1);
    - 3 – очередное простое число вида 2 n - 1 (начиная с n = 1);
- окно, содержащее:
    - текстовое поле (объект Text), куда выводится начало некоторого небольшого сообщения (рассказа и т.п.);
    - поле ввода пароля (объект Entry);
    - кнопку для продолжения вывода сообщения (объект Button);
    - второе текстовое поле (объект Text), куда выводится неожиданное (смешное) завершение сообщения (при правильном вводе пароля) или сообщение о неправильном вводе пароле;

  - метки (объект Label) – при необходимости;
- окно, содержащее:
  - пять флажков (объекты Checkbutton), задающие согласно колонке "Флажки" табл. №1:
    - 1 – числа 1, 2, 4, 8 и 16;
    - 2 – 5 букв кириллицы (на усмотрение студента);
    - 2 – 5 английских букв (на усмотрение студента);
  - текстовое поле, куда помещается результат:
    - для чисел – сумма заданных чисел;
    - для букв – слово, состоящее из выбранных букв;
- окно, содержащее:
  - переключатели (объекты Radiobutton), задающие согласно колонке "Переключатели" табл. №1:
    - 1 – изменяемые типы;
    - 2 – неизменяемые типы;
    - 3 – контейнеры;
    - 4 – последовательности;
  - текстовое поле, куда помещается результат – пример литерала для выбранного класса.

Таблица 1 – Перечень индивидуальных заданий

Номер п/п	**Число**	**Флажки**	**Переключатели**
1	1	3	1
2	2	1	2
3	3	2	3
4	1	3	4
5	2	1	1
6	3	2	2
7	1	3	3
8	2	1	4
9	3	2	1
10	1	3	2
11	2	1	3
12	3	2	4
13	1	3	1
14	2	1	2
15	3	2	3
16	1	3	4
17	2	1	1
18	3	2	2
19	1	3	3
20	2	1	4

Шкала оценивания:

Оценка «отлично» за практическую работу выставляется, если работа выполнена полностью правильно и самостоятельно, а аспирант способен уверенно отвечать на вопросы преподавателя о проделанной работе;

Оценка «хорошо» выставляется, если работа выполнена в целом корректна, но имеются неточности в ответах на вопросе преподавателя или некоторые действия выполнены неоптимально;

Оценка «удовлетворительно» выставляется, когда работа не может быть завершена аспирантом самостоятельно, без помощи преподавателя.

Оценка «неудовлетворительно» выставляется, если работа не была выполнена.

Практическое занятие 9. Заключительное занятие. Подведение и анализ промежуточных результатов освоения дисциплины

При подготовке материалов для заключительного практического занятия, преподаватель учитывает недостаточную степень развития навыков аспирантов, определенную в ходе практических занятий и при оценке самостоятельной работы. На практическом занятии обсуждаются итоги проведения занятий по дисциплине, проверяется полнота и качество собранного аспирантом портфолио материалов и результатов выполнения заданий по дисциплине. Обсуждаются вопросы для подготовки к сдаче кандидатского экзамена по специальности 2.3.5. Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей. Оцениваются индивидуальные достижения, полученные аспирантов в ходе внеурочной работы (сертификаты олимпиад, конкурсов, грантов, написание тезисов/статей в сборники конференций различного уровня).

3. Методические указания по выполнению других видов самостоятельной работы по дисциплине

ПРОГРАММА ДЛЯ ПОДГОТОВКИ К СДАЧЕ КАНДИДАТСКОГО ЭКЗАМЕНА ПО СПЕЦИАЛЬНОСТИ

1. Математические основы программирования

Понятие алгоритма и его уточнения: машины Тьюринга, нормальные алгоритмы Маркова, рекурсивные функции. Эквивалентность данных формальных моделей алгоритмов. Понятие об алгоритмической неразрешимости. Примеры алгоритмически неразрешимых проблем.

Понятие сложности алгоритмов. Классы P и NP. Полиномиальная сводимость задач. Теорема Кука об NP-полноте задачи выполнимости булевой формулы. Примеры NP-полных задач, подходы к их решению. Точные и приближенные комбинаторные алгоритмы.

Примеры эффективных (полиномиальных) алгоритмов: быстрые алгоритмы поиска и сортировки; полиномиальные алгоритмы для задач на графах и сетях (поиск в глубину и ширину, о минимальном остове, о кратчайшем пути, о назначениях).

Автоматы. Эксперименты с автоматами. Алгебры регулярных выражений. Теорема Клини о регулярных языках.

Алгебра логики. Булевы функции, канонические формы задания булевых функций. Понятие полной системы. Критерий полноты Поста. Минимизация булевых функций в классах нормальных форм.

Исчисление предикатов первого порядка. Понятие интерпретации. Выполнимость и общезначимость формулы первого порядка. Понятие модели. Теорема о полноте исчисления предикатов первого порядка.

Отношения и функции. Отношение эквивалентности и разбиения. Фактор множества. Отношения частичного порядка. Теоретико-множественное и алгебраическое определения решетки, их эквивалентность. Свойства решеток. Булевы решетки. Полные решетки.

Формальные языки и способы их описания. Классификация формальных грамматик. Их использование в лексическом и синтаксическом анализе.

λ-исчисление, правила редукции, единственность нормальной формы и правила ее достижения, представление рекурсивных функций.

Основы комбинаторного анализа. Метод производящих функций, метод включений и исключений. Примеры применения.

Коды с исправлением ошибок. Алфавитное кодирование. Методы сжатия информации.

Основы криптографии. Задачи обеспечения конфиденциальности и целостности информации. Теоретико-информационный и теоретико-сложностный подходы к определению криптографической стойкости. Американский стандарт шифрования DES и российский стандарт шифрования данных ГОСТ 28147-89. Системы шифрования с открытым ключом (RSA). Цифровая подпись. Методы генерации и распределения ключей.

2. Вычислительные машины, системы и сети

Архитектура современных компьютеров. Организации памяти и архитектура процессора современных вычислительных машин. Страничная и сегментная организация виртуальной памяти. Кэш-память. Командный и арифметический конвейеры, параллельное выполнение независимых команд, векторные команды. Специализированные процессоры. Машины, обеспечивающие выполнение вычислений, управляемых потоком данных. Организация ввода-вывода, каналы и процессоры ввода-вывода, устройства сопряжения с объектами.

Классификация вычислительных систем (ВС) по способу организации параллельной обработки. Многопроцессорные и многомашинные комплексы. Вычислительные кластеры. Проблемно-ориентированные параллельные структуры: матричные ВС, систолические структуры, нейросети.

Назначение, архитектура и принципы построения информационно – вычислительных сетей (ИВС). Локальные и глобальные ИВС, технические и программные средства объединения различных сетей.

Методы и средства передачи данных в ИВС, протоколы передачи данных.

Особенности архитектуры локальных сетей (Ethernet, Token Ring, FDDI).

Сеть Internet, доменная организация, семейство протоколов TCP/IP. Информационно-вычислительные сети и распределенная обработка информации.

3. Языки и системы программирования. Технология разработки программного обеспечения

Языки программирования. Процедурные языки программирования (Фортран, Си), Функциональные языки программирования (Лисп), логическое программирование (Пролог), объектно-ориентированные языки программирования (Ява).

Процедурные языки программирования. Основные управляющие конструкции, структура программы. Работа с данными: переменные и константы, типы данных (булевский, целочисленные, плавающие, символьные, типы диапазона и перечисления, указатели), структуры данных (массивы и записи). Процедуры (функции): вызов процедур, передача параметров (по ссылке, по значению, по результату), локализация переменных, побочные эффекты. Обработка исключительных ситуаций. Библиотеки процедур и их использование.

Объектно-ориентированное программирование. Классы и объекты, наследование, интерфейсы. Понятие об объектном окружении. Рефлексия. Библиотеки классов. Средства обработки объектов (контейнеры и итераторы).

Распределенное программирование. Процессы и их синхронизация. Семафоры, мониторы Хоара. Объектно-ориентированное распределенное программирование. CORBA. Параллельное программирование над общей памятью. Нити. Стандартный интерфейс Open MP. Распараллеливание последовательных программ. Параллельное программирование над распределенной памятью. Парадигмы SPMD и MIMD. Стандартный интерфейс MPI.

Основы построения трансляторов. Структура оптимизирующего транслятора. Промежуточные представления программы: последовательность символов, последовательность лексем, синтаксическое дерево, абстрактное синтаксическое дерево. Уровни промежуточного представления: высокий, средний, низкий. Формы промежуточного представления.

Анализ исходной программы в компиляторе. Автоматные (регулярные) грамматики и сканирование, контекстно свободные грамматики и синтаксический анализ, организация таблицы символов программы, имеющей блочную структуру, хеш-функции. Нисходящие (LL(1)-грамматики) и восходящие (LR(1)-грамматики) методы синтаксического анализа. Атрибутные грамматики и семантические программы, построение абстрактного синтаксического дерева. Автоматическое построение лексических и синтаксических анализаторов по формальным описаниям грамматик. Системы lex и yacc. Система Gentle.

Оптимизация программ при их компиляции. Оптимизация базовых блоков, чистка циклов. Анализ графов потока управления и потока данных. Отношение доминирования и его свойства, построение границы области доминирования вершины, выделение сильно связанных компонент графа. Построение графа зависимостей. Перевод программы в SSA-представление и обратно. Глобальная и межпроцедурная оптимизация.

Генерация объектного кода в компиляторах. Перенастраиваемые (retargetable) компиляторы, gcc (набор компиляторов Gnu). Переработка термов (term rewriting). Применение оптимизационных эвристик (целочисленное программирование, динамическое программирование) для автоматической генерации генераторов объектного кода (системы BEG, Iburg и др.).

Машинно-ориентированные языки, язык ассемблера. Представление машинных команд и констант. Команды транслятору. Их типы, принципы реализации. Макросредства, макровызовы, языки макроопределений, условная макрогенерация, принципы реализации.

Системы программирования (СП), типовые компоненты СП: языки, трансляторы, редакторы связей, отладчики, текстовые редакторы. Модульное программирование. Типы модулей. Связывание модулей по управлению и данным.

Пакеты прикладных программ (ППП). Системная часть и наполнение. Языки общения с ППП. Машинная графика. Средства поддержки машинной графики. Графические пакеты.

Технология разработки и сопровождения программ. Жизненный цикл программы. Этапы разработки, степень и пути их автоматизации. Обратная инженерия. Декомпозиционные и сборочные технологии, механизмы наследования, инкапсуляции, задания типов. Модули, взаимодействие между модулями, иерархические структуры программ.

Отладка, тестирование, верификация и оценивание сложности программ. Генерация тестов. Системы генерации тестов. Срезы программ (slice, chop) и их применение при отладке программ и для генерации тестов.

Методы спецификации программ. Методы проверки спецификации. Схемное, структурное, визуальное программирование. Разработка пользовательского интерфейса, стандарт CUA, мультимедийные среды интерфейсного взаимодействия.

4. Операционные системы

Режимы функционирования вычислительных систем, структура и функции операционных систем. Основные блоки и модули. Основные средства аппаратной поддержки функций операционных систем (ОС): система прерываний, защита памяти, механизмы преобразования адресов в системах виртуальной памяти, управление каналами и периферийными устройствами.

Виды процессов и управления ими в современных ОС. Представление процессов, их контексты, иерархии порождения, состояния и взаимодействие. Многозадачный (многопрограммный) режим работы. Команды управления процессами. Средства взаимодействия процессов. Модель клиент-сервер и ее реализация в современных ОС.

Параллельные процессы, схемы порождения и управления. Организация взаимодействия между параллельными и асинхронными процессами: обмен сообщениями, организация почтовых ящиков. Критические участки, примитивы взаимоисключения процессов, семафоры Дейкстры и их расширения. Проблема тупиков при асинхронном выполнении процессов, алгоритмы обнаружения и предотвращения тупиков.

Операционные средства управления процессами при их реализации на параллельных и распределенных вычислительных системах и сетях: стандарты и программные средства PVM, MPI, OpenMP, POSIX.

Одноуровневые и многоуровневые дисциплины циклического обслуживания процессов на центральном процессоре, выбор кванта.

Управление доступом к данным. Файловая система, организация, распределение дисковой памяти. Управление обменом данными между дисковой и оперативной памятью. Рабочее множество страниц (сегментов) программы, алгоритмы его определения.

Управление внешними устройствами.

Оптимизация многозадачной работы компьютеров. Операционные системы Windows, Unix, Linux. Особенности организации, предоставляемые услуги пользовательского взаимодействия.

Операционные средства управления сетями. Эталонная модель взаимодействия открытых систем ISO/OSI. Маршрутизация и управление потоками данных в сети. Локальные и глобальные сети. Сетевые ОС, модель клиент — сервер, средства управления сетями в ОС UNIX, Windows NT. Семейство протоколов TCP/IP, структура и типы IP-адресов, доменная адресация в Internet. Транспортные протоколы TCP, UDP .

Удаленный доступ к ресурсам сети. Организация электронной почты, телеконференций. Протоколы передачи файлов FTP и HTTP, язык разметки гипертекста HTML, разработка WEB-страниц, WWW-серверы.

5. Методы хранения данных и доступа к ним. Организация баз данных и знаний

Концепция типа данных. Абстрактные типы данных. Объекты (основные свойства и отличительные признаки).

Основные структуры данных, алгоритмы обработки и поиска. Сравнительная характеристика методов хранения и поиска данных.

Основные понятия реляционной и объектной моделей данных.

Теоретические основы реляционной модели данных (РДМ). Реляционная алгебра, реляционное исчисление. Функциональные зависимости и нормализация отношений.

CASE-средства и их использование при проектировании базы данных (БД).

Организация и проектирование физического уровня БД. Методы индексирования.

Обобщенная архитектура, состав и функции системы управления базой данных (СУБД). Характеристика современных технологий БД. Примеры соответствующих СУБД.

Основные принципы управления транзакциями, журнализацией и восстановлением.

Язык баз данных SQL. Средства определения и изменения схемы БД, определения ограничений целостности. Контроль доступа. Средства манипулирования данными.

Стандарты языков SQL. Интерактивный, встроенный, динамический SQL.

Основные понятия технологии клиент—сервер. Характеристика SQL-сервера и клиента. Сетевое взаимодействие клиента и сервера.

Информационно-поисковые системы. Классификация. Методы реализации и ускорения поиска.

Методы представления знаний: процедурные представления, логические представления, семантические сети, фреймы, системы продукций. Интегрированные методы представления знаний. Языки представления знаний. Базы знаний.

Экспертные системы (ЭС). Области применения ЭС. Архитектура ЭС. Механизмы вывода, подсистемы объяснения, общения, приобретения знаний ЭС. Жизненный цикл экспертной системы. Примеры конкретных ЭС.

6. Защита данных и программных систем

Аппаратные и программные методы защиты данных и программ. Защита данных и программ с помощью шифрования.

Защита от несанкционированного доступа в ОС. Система безопасности и разграничения доступа к ресурсам ОС.

Защита от несанкционированного копирования. Методы простановки некопируемых меток, настройка устанавливаемой программы на конкретный компьютер, настройка на конфигурацию оборудования.

Защита от разрушающих программных воздействий. Вредоносные программы и их классификация. Загрузочные и файловые вирусы, программы-закладки. Методы обнаружения и удаления вирусов, восстановления программного обеспечения.

Защита информации в вычислительных сетях.

По представленным темам программы предполагается выполнение аспирантом конспектов или повторение материала с использованием конспектов лекций.

4. Примеры контрольных заданий (оценочных средств), используемых при текущем контроле успеваемости и промежуточной аттестации по дисциплине

Демоверсия итогового тестирования:

1. Как называется свойство алгоритма, заключающееся в том, что он разрабатывается в общем виде (т. е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными)?

а) дискретность;

б) результативность;

в) массовость (правильный ответ);

г) понятность;

2. Как описывается алгоритм быстрой сортировки (quick-sort)?

а) для нового неупорядоченного элемента в правой части массива итеративно выбирается место среди уже упорядоченных ключей;

б) массив делится рекурсивно на две части, элементы массива переставляются так, чтобы в левой части оказались элементы, которые не превосходят по величине элементов в правой части (правильный ответ);

в) итеративно выбирается место среди оставшихся неупорядоченных ключей, найденный минимум или максимум вынимается из текущего множества в ответ;

г) исходная последовательность A делится на две части A_1 и A_2, которые рекурсивно сортируются;

3. Пусть на вход алгоритма быстрой сортировки поступает N различных ключей. Каким будет математическое ожидание времени его работы?

а) $O(N^2)$;

б) O(N * log N) (правильный ответ);

в) O(N);

г) O(log N);

4. Что умел распознавать персептрон Ф.Розенблата ?

а) буквы алфавита (правильный ответ);

б) рукописные символы;

в) кубики и пирамиды;

г) объекты военного назначения;

5. Тест Тьюринга имеет следующие важные особенности:

а) дает объективное понятие об интеллекте, т.е. реакции заведомо разумного существа на определенный набор вопросов и предотвращает дебаты об “истинности” его природы (правильный ответ);

б) препятствует заведению в тупик безответными вопросами типа “должна ли машина осознавать свои действия?” (правильный ответ);

в) показывает, что компьютеры могут делать лишь то, что в них заложено разработчиками;

г) исключает предвзятость в пользу живых существ (правильный ответ);

6. Какие из перечисленных свойств относятся к сетям с выделенным сервером?

а) отсутствие центрального места хранения разделяемых файлов;

б) не требуют специальной должности администратора сети;

в) обладают сильной централизованной защитой (правильный ответ);

г) освобождают пользователей от задачи управления разделяемыми ресурсами (правильный ответ);

д) способность обслуживать одновременно большое количество сетевых соединений (правильный ответ);

7. Какие из перечисленных конструкций не являются конструкциями функциональной модели предметной области базы данных?

а) модель потока данных;

б) бизнес-модель процессов;

в) диаграмма сущность-связь (правильный ответ);

г) модель жизненного цикла сущности;

8. Модель жизненного цикла сущности предназначена для:

а) описания процессов перемещения данных в предметной области базы данных;

б) описания процессов и функций системы в предметной области базы данных;

в) описания изменения состояний сущности и переходов между ними (правильный ответ).

Автор(ы)-составитель(и): *Привалов А.Н.*, профессор кафедры математики, информатики и методики обучения, доктор технических наук, профессор