НАМАНГАНСКИЙ ИНЖЕНЕРНО-СТРОИТЕЛЬНЫЙ ИНСТИТУТ

Абдуллаева Озода Сафибуллаевна

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ТЕХНИЧЕСКИХ СИСТЕМАХ

МИНИСТЕРСТВО ВЫСШЕГО И СРЕДНЕГО СПЕЦИАЛЬНОГО ОБРАЗОВАНИЯ РЕСПУБЛИКИ УЗБЕКИСТАН

НАМАНГАНСКИЙ ИНЖЕНЕРНО-СТРОИТЕЛЬНЫЙ ИНСТИТУТ

Абдуллаева Озода Сафибуллаевна

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Учебник для бакалавров по специальности

60710600 – Электроэнергия (Электроснабжение)
60711000 – Альтернативные источники энергии (по видам)

2022 год

УДК: 69.002.8

Аннотация: Данный учебник предназначен для студентов 1-го курса по следующим направлениям образования: 60710600 – Электроэнергия (Электроснабжение), 60711000 – Альтернативные источники энергии (по видам). Данный учебник также могут использовать студенты по следующим сферам образования: 710 000 – Инженерные работы, 720 000 – Производство и обработка, 640 000 – Безопасность жизнедеятельности, изучающих дисциплину «Информационные технологии в технических системах».

Учебник разработан в полном соответствии с учебной программой дисциплины «Информационные технологии в технических системах».

Основная цель учебника оказать помощь обучающимся в приобретении знаний и выработке рефлексивных умений по дисциплине «Информационные технологии в технических системах». В нём приводятся теоретические и практические основы информационных технологий.

Особое внимание посвящено актуальных проблемам основ предмета информационные технологии, основным направлениям использования информационных технологий в технических системах, моделирования информационных процессов, а также вопросам безопасности информации и способам защиты информации, алгоритмизации и программирования информационных процессов, компьютерной графики, трехмерного моделирования.

Данный учебник может также представлять интерес для широкого круга специалистов и преподавателей.

Ключевые слова: информационные технологии, технические системы, моделирование, информация, алгоритмизация, программирование, информационные процессы, компьютерная графика, трехмерное моделирование, технологии.

Рекомендовано к изданию решением Совета Наманганского инженерно-строительного института от 29.10.2021 г. выпиской протокола №3

Ответственный редактор:
доцент кафедры «Информационных технологий» НамИТИ,
кандидат технических наук,
К.Д.Исманова

Рецензенты:
профессор кафедры «Аудиовизуализация» ТУИТ,
доктор технических наук (DSc),
С.С.Бекназарова

доцент кафедры «Информационные технологии в технических системах» НамИСИ, PhD
А.И.Исомиддинов

ВВЕДЕНИЕ

Характерной чертой нашего времени являются интенсивно развивающиеся процессы информатизации практически во всех сферах человеческой деятельности. Они привели к формированию новой информационной инфраструктуры, которая связана с новым типом общественных отношений, с новой реальностью, с новыми информационными технологиями различных видов деятельности.

В данном учебнике рассматриваются основные теоретические и практические аспекты дисциплины «Информационные технологии в технических системах». Целью обучения данной дисциплины является освоение обучающими технологического подхода к информационной деятельности как способа её теоретического осмысления и практического внедрения информационных технологий в технических системах, так и в общественной жизни.

Основными задачами при изучении дисциплины являются формирование следующих профессиональных компетенций:

- способность к проектированию базовых и прикладных информационных технологий;
- способность разрабатывать средства реализации информационных технологий (методические, информационные, математические, технические и программные.

Учебник разработан в полном соответствии с учебной программой дисциплины «Информационные технологии в технических системах».

Структура издания ориентирована на системное изложение учебного материала 1 семестра обучения данного курса. В первой главе «Основные направления использования информационных технологий в технических системах» излагается информация об основных направлениях использования современных компьютерных технологий в технических системах, современных системах автоматизированного проектирования и их применение в технических областях, экспертных системах и их программном

обеспечении. Во второй главе под названием «Моделирование информационных процессов» представлены сведения о классификации видов моделирования технических систем, об основах математического моделирования и набор специальных программ для статистической обработки данных, основах графического моделирования использовании графических возможностей AutoCAD в процессе проектирования, особенностях имитационного моделирования в технических системах. В третьей главе «Сетевые возможности в технических системах. Информационная безопасность» рассмотрены основные понятия сетевых технологий и облачных сервисов, гипертекстовых и мультимедийных информационных технологий, криптографических методов защиты данных. В четвёртой главе «Современные технологии программирования» изложены сведения об алгоритмизации и программировании процессов в технических системах, языках программирования PYTHON. В конце каждого параграфа представлены вопросы для самоконтроля. В издании имеется список использованной литературы, на которую в тексте имеются ссылки.

ГЛАВА I. ОСНОВНЫЕ НАПРАВЛЕНИЯ ИСПОЛЬЗОВАНИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В ТЕХНИЧЕСКИХ СИСТЕМАХ

1.1. Основные направления использования современных компьютерных технологий в технических системах

Основные модули

- Роль информационных технологий в развитии общества.
- Информационная культура и информатизация общества.
- Информационные продукции.
- Характеристика информации и данных.
- Информационные технологии.
- Значение технических средств во внедрении компьютерных технологий.

Роль информационных технологий в развитии общества чрезвычайно велика. С ней связано начало революции в области накопления, передачи и обработки информации. Эта революция, следующая за революциями в овладении веществом и энергией, затрагивает и коренным образом преобразует не только сферу материального производства, но и интеллектуальную, духовную сферы жизни.

Информатизация общества – организованный социально-экономический и научно-технический процесс создания оптимальных условий для удовлетворения информационных потребностей и реализации прав граждан, органов государственной власти, органов местного самоуправления организаций, общественных объединений на основе формирования и использования информационных ресурсов.

Цель информатизации – улучшение качества жизни людей за счет увеличения производительности и облегчения условий их труда.

Информатизация – это сложный социальный процесс, связанный со значительными изменениями в образе жизни населения. Он требует серьёзных усилий на многих направлениях, включая ликвидацию компьютерной неграмотности, формирование культуры использования новых информационных технологий и др.

Информационная технология – это совокупность методов и устройств, используемых людьми для обработки информации.

Поскольку, как сказано выше, информатика – это научное направление, занимающееся изучением законов, методов и способов накапливания, обработки и передачи информации с помощью ЭВМ и других технологических средств, компьютеры, являясь средством изучения предмета информатики (информации) составляют ее неотъемлемую часть. Информатизация – это реализация комплекса мер, направленных на обеспечение полного и своевременного использования достоверных знаний во всех общественно – значимых видах человеческой деятельности. Трудно переоценить роль ЭВМ в процессе информатизации, поскольку последняя на современном этапе развития науки и техники осуществляется исключительно посредством ЭВМ. Поэтому, необходимо изучить устройство ЭВМ и принцип его работы.

Основные устройства, периферийные устройства ПК

ПК состоит из следующих основных элементов: процессор, монитор, клавиатура, и дополнительные устройства. Клавиатура служит для ввода символов в компьютер, следовательно, является устройством, служащим для ввода информации пользователем в компьютер. Монитор, или дисплей дает возможность отображения информации, т.е. служит для вывода на экран текстовой и графической информации и работает, соответственно, в двух режимах текстовом и графическом. В текстовом режиме экран условно разделен на столбцы и строки. В графическом режиме, кроме текстовой информации на экран выводится и график. В этом режиме экран состоит из точек (пиксели).

Основной (системный) блок – основное устройство ЭВМ, состоит из устройств, в котором расположены микропроцессор, оперативная память, жесткий диск, контроллеры.

Микропроцессор – выполняет функции управления компьютером и все вычислительные действия. Микропроцессор или микрочип – целая электрическая фабрика, состоящая из сотен тысяч микроскопических электронных схем, выгравированных на поверхности крошечного кремниевого кристалла. Работой микропроцессора управляют электро-импульсы, которые открывают и закрывают его тысячи и миллионы раз в секунду. Каждое открытие или закрытие представляет собой единицу информации, закодированную в виде 0 или 1.

Оперативная память (ОП) хранит работающие в компьютере программы и информацию. Информация переносится с постоянной (внешней) памяти в оперативную, полученные результаты при необходимости записывается на жесткий диск.

Жесткий диск используется для постоянного хранения используемых программ и информации. Это программы операционной системы (ОС), редакторы, системы программирования, прикладные программы, информация и др.

Контроллеры – (электрические схемы) управляют работой устройств, из которых состоит компьютер.

Периферийные устройства ЭВМ:

Принтер – предназначен для вывода информации на бумагу. Обычно принтеры могут выводить не только текстовую информацию, но также рисунки и графики. Одни принтеры позволяют печатать только в одном цвете (черном), другие могут выводить также и цветные изображения.

Существуют несколько видов принтеров:

- Матричные принтеры. Сейчас эти принтеры вытеснены струйными и лазерными, так как обеспечивают значительно худшее качество печати, сильно шумят при работе и малопригодны для цветной печати.

- Лазерные принтеры обеспечивают наилучшее (близко к типографическому) качество черно-белой печати, а цветные лазерные принтеры – также и очень высокое качество цветной печати. Лазерные принтеры обеспечивают самую высокую среди всех принтеров скорость печати и не требуют специальной бумаги.

- Струйные принтеры являлись одним из распространенных типов принтеров.

Мышь – манипулятор, использующийся для ввода данных. Мышь значительно облегчает диалог пользователя с компьютером и поэтому сейчас является неотъемлемой частью ПК.

Модем – это устройство для обмена информацией с другими компьютерами через телефонную сеть. Для всех пользователей, желающих использовать глобальные электронные сети типа Internet, работать с электронной почтой, получать извне офиса доступ к локальной сети своей фирмы, посылать и получать факсы с помощью компьютера необходим модем.

Сканер – вводит информацию с бумаги в виде изображения в компьютер.

Плоттер – устройство, выводящее на бумагу графику.

Сетевой адаптер – дает возможность подсоединения компьютеров к местной сети.

Средства мультимедиа. «Мультимедиа» означает возможность работы с информацией в различных видах, а не только в цифровом виде. Прежде всего, здесь имеются в виду звуковая и видеоинформация. К средствам мультимедиа относятся: звуковая карта, колонки, дисковод для компакт-дисков и некоторое программное обеспечение.

Характеристика информационных технологий в технических системах

Информационная продукция – документы, информационные массивы, базы данных и информационные услуги, являющиеся результатом функционирования информационных систем.

Термин «**информация»** происходит от латинского слова «**informatio"**, что означает **сведения, разъяснения, изложение**.

Информация – это обозначение содержания, полученного из внешнего мира в процессе нашего приспособления к нему и приспособления к нему наших чувств.

Информация может существовать в виде:

- текстов, рисунков, чертежей, фотографий;
- световых или звуковых сигналов;
- радиоволн;
- электрических и нервных импульсов;
- магнитных записей;
- жестов и мимики;
- запахов и вкусовых ощущений;
- хромосом, посредством которых передаются по наследству признаки и свойства организмов и т.д.

Понятие **«информационная технология»** базируется на понятии «технология». Наиболее широкое по содержанию его толкование дал польский философ и писатель Станислав Лем, который определил технологии как «...обусловленные состоянием знаний и общественной эффективностью способы достижения целей, поставленных обществом...».

А наиболее распространенным является определение, зафиксированное в различных энциклопедиях и словарях: «ТЕХНОЛОГИЯ (от греч. ***techne*** – искусство, мастерство, умение и ...логия) – совокупность методов обработки, изготовления, изменения состояния, свойств, формы сырья, материала или полуфабриката, осуществляемых в процессе производства продукции...».

Последнее определение, несомненно, уже и конкретнее того, которое сформулировано С. Лемом, поскольку ограничивает его применение сферой промышленного производства, но оно позволяет вычленить основные его составляющие (рис. 1.1.1):

- объект технологии, т.е. то, на что направлены действия, осуществляемые в рамках технологии (сырье, материалы, полуфабрикаты);

- цель технологии, т. е. конечный результат действий, осуществляемых в рамках технологии (обработка, изготовление, изменение состояния, свойств, формы);

- средства технологии и методы их применения, т.е. способы осуществления действий над объектом технологии для достижения цели технологии.

Поскольку в соответствии с определением С.Лема технологии не ограничиваются сферой промышленного производства, а определяются потребностями общества во всем их многообразии, то различные области человеческой деятельности требуют и различных технологий.

Различия технологий проявляются в том, на что направлена деятельность людей в той или иной сфере, т.е. в объектах технологий. Для промышленного производства, как уже указывалось, это сырье, Материалы, полуфабрикаты – все, что составляет материально-вещественные ресурсы производства.

Технология		
Объект	Цель	Средства и методы

Рис. 1.1.1. Составляющие понятия «технология»

Если в качестве объекта деятельности, а следовательно, и соответствующих способов ее осуществления выступают энергетические ресурсы (например, электрическая энергия), то мы получаем энергетические технологии (производство, передача, преобразование, распределение, потребление энергии).

Финансовые ресурсы как объект деятельности порождают финансовые технологии (банковские и бухгалтерские технологии, технологии работы на

рынке ценных бумаг, технологии финансового и экономического анализа и т.п.).

Информация как общественный ресурс тоже является объектом деятельности и, следовательно, связана с соответствующими технологиями – информационными технологиями.

Опираясь на рассмотренное содержание понятия «технология», можно сформулировать следующее определение понятия «информационная технология»:

Информационная технология – это совокупность средств и методов их применения для целенаправленного изменения свойств информации, определяемого содержанием решаемой задачи или проблемы

Объекты информационных технологий. В сформулированном определении понятия «информационная технология» в качестве ее объекта выступает информация. В современной научной литературе существует множество подходов к определению содержания понятия «информация».

Для наших целей достаточно указать на практическое совпадение содержания таких понятий, как «информация», «сведения» «сообщение», «данные», которые в словарях и энциклопедиях определяются друг через друга. Будем в дальнейшем опираться на достаточно однозначное понимание содержания этих понятий как сведений о чем-либо.

Эти сведения или информация как объект информационных технологий характеризуются формой восприятия или представления и содержательной интерпретацией, а также материальным носителем (рис. 1.1.2).

Объекты информационных технологий		
Форма представления и восприятия	Содержательная интерпретация	Материальный носитель

Рис. 1.1.2. Характеристика объекта информационных технологий

Форма восприятия и представления информации определяет основной способ конечного их использования в той или иной сфере деятельности и предполагает один из следующих вариантов (рис. 1.1.3):

- текстовая информация;
- аудиоинформация (звуковая);
- видеоинформация (визуальная).

Форма восприятия и представления информации		
Текстовая информация	Аудио-информация (звуковая)	Видео-информация (визуальная)

Рис. 1.1.3. Формы восприятия и представления информации

Текстовая информация – это различные виды письменной речи или представления данных с помощью систем специальных знаков (математические и химические формулы, тексты программ и т.п.).

Аудиоинформация – это устная речь, музыка, звуки естественного или искусственного происхождения, системы звуковых сигналов различного назначения.

Видеоинформация – это различного вида образы, воспринимаемые органами зрения (рисунки, схемы, карты, фильмы и т.п.).

Содержательная интерпретация определяет восприятие конкретной информации той или иной формы восприятия и представления в рамках конкретного вида деятельности или решаемой задачи.

Так, текст некоторого документа на английском языке понятен и может быть использован специалистом, знающим английский язык, но не имеет практического смысла для человека, не владеющего указанным языком. Одна и та же математическая формула описывает различные сущности в зависимости от интерпретации операндов, ее составляющих. Одни и те же звуковые сигналы, подаваемые с помощью горна в различных армиях мира, воспринимаются по-разному. Этих примеров достаточно для того, чтобы

показать необходимость такой характеристики информации, как ее содержательная интерпретация.

Носитель информации – это материальное воплощение информации той или иной формы восприятия и представления.

В принципе, в качестве носителя информации может выступать любой материальный объект (в том числе и физическое поле той или иной природы), определенные состояния или свойства которого могут рассматриваться как представление информации. Рассмотрим примеры.

Носителями ***текстовой информации*** в разное время человеческой истории выступали такие материальные объекты, как поверхность каменных пещер, выделанные шкуры животных, изготовленные из тростника папирусные свитки, берестяная кора, глиняные и деревянные дощечки, ткани и, наконец, наиболее распространенный в этом отношении носитель – бумага. Все эти носители имели то свойство, что в течение определенного времени изменяли свои физические свойства в диапазоне, позволяющем сохранять изображение текста.

Носители ***аудиоинформации*** не так разнообразны. Это прежде всего естественная среда, передающая звуковые волны, а также различного рода искусственные материальные объекты, определенные свойства которых позволяют фиксировать, хранить и воспроизводить звуковые колебания (восковые валики, виниловые диски, намагниченные проволока и пленка, магнитные и оптические диски). Естественно, следует упомянуть и электромагнитные поля, позволяющие воспринимать, передавать и воспроизводить звуковые колебания (радио, телефон, телеграф и т.п.).

Носители ***видеоинформации*** естественным образом включают в себя все перечисленные выше носители текстовой информации. Кроме того, они включают в себя различного рода фотоматериалы, голографические пластины и прочие материалы, позволяющие фиксировать, хранить и воспроизводить зрительные образы. К носителям видеоинформации следует

отнести электромагнитные поля, позволяющие воспринимать, передавать и воспроизводит звуковые колебания (телевидение).

К особым видам носителей информации относят так называемые «электронные». Это не вполне точное название (поскольку в большинстве случаев речь идет о магнитных и оптических носителях) объединяет все виды носителей, которые хранят данные в виде некоторых объектов (файлов, дисковых томов и т.п.), интерпретация которых с помощью программ, выполняемых компьютером, воспроизводит ту или иную форму информации на соответствующих устройствах.

Результаты информационных технологий

Целью, или результатом, информационной технологии является целенаправленное изменение свойств информации, определяемое содержанием решаемой задачи или проблемы.

Такие изменения осуществляются с помощью различного рода информационных преобразований.

Каждое такое преобразование характеризуется содержанием, направлением и объемом (рис. 1.1.4).

Информационное преобразование		
Содержание	Направление	Объем

Рис.1.1.4. Характеристики информационного преобразования

Содержание информационного преобразования определяется конкретным набором изменяемых свойств информации, и с этой точки зрения выделяют следующие информационные преобразования (рис. 1.1.5):

- сбор информации;
- накопление информации;
- регистрацию информации;
- передачу информации;

- копирование информации;
- упорядочение информации;
- хранение информации;
- поиск информации;
- представление информации;
- выдачу информации;
- защиту информации.

Сбор информации представляет собой процесс получения сведений из различных источников о состоянии тех явлений и объектов, свойства которых являются существенными для решения конкретных задач.

Накопление информации – это процесс аккумулирования собранных сведений в каком-либо накопителе в том случае, когда нет возможности немедленного их использования.

Регистрация информации – это процесс фиксирования собранных (или иных) сведений на том или ином материальном носителе.

Передача информации – это процесс изменения пространственных координат сведений, т.е. их перемещение из одного места в другое.

Копирование информации – это процесс дублирования сведений для одновременного их использования в нескольких местах.

Упорядочение информации – это процесс размещения сведений в соответствии с определенными отношениями между ними.

Хранение информации – это процесс изменения временных координат сведений, т.е. их содержание в хранилище (архиве) с целью последующего использования. Хранится только упорядоченная информация.

Поиск информации – это процесс выборки сведений из хранимой информации по тому или иному запросу. Запросы, как правило, учитывают упорядоченность хранимой информации.

Представление информации – это процесс приведения сведений из формы получения (при передаче) или хранения (при поиске) в форму, удобную для последующего использования при решении конкретных задач.

Выдача информации – это процесс передачи сведений в необходимой форме представления для решения конкретных задач.

Защита информации – это процесс обеспечения сохранности сведений как таковых, а также процесс ограничения доступа к ним

При решении конкретных задач для каждого вида информационного преобразования определяются его направление и объев. Направление характеризует конкретную реализацию преобразования (например, степень упорядоченности в соответствующем преобразовании), а объем – его количественные характеристик (например, количество сведений, передаваемых на хранение).

Средства и методы информационных технологий

Каждое информационное преобразование в зависимости от его направления и объема, а также возможностей конкретной реализации может осуществляться различными методами и средствам.

Средства и методы информационных технологий включают в себя (рис. 1.1.5):

- комплекс технических средств;
- средства управления техническим комплексом;
- организационно-методическое обеспечение.

Средства и методы информационных технологий		
Комплекс технических средств	Средства управления техническим комплексом	Организационно-методическое обеспечение

Рис. 1.1.5. Структура средств и методов информационных

Комплекс технических средств – это совокупность инструментов, приспособлений, машин, механизмов и автоматических устройств, с

помощью которых осуществляется собственно информационное преобразование.

Средства управления техническим комплексом позволяют персоналу осуществлять целенаправленное использование технических средств для реализации информационного преобразования.

Организационно-методическое обеспечение увязывает реализацию всех действий технических средств и персонала в единый монологический процесс в соответствии с назначением конкретного информационного преобразования и включает в себя:

- нормативно-методические материалы по подготовке и оформлению различных документов в рамках решения конкретной задачи;
- инструктивные и нормативные материалы по эксплуатации технических средств, в том числе по технике безопасности работы и по условиям поддержания нормальной работоспособности оборудования;
- инструктивные и нормативно-методические материалы по организации работы персонала в рамках конкретной информационной технологии.

Если основу комплекса технических средств составляют средства компьютерной техники, то речь идет о **компьютерных информационных технологиях**.

Архитектурой компьютера называется его описание на некотором общем уровне, включающее описание пользовательских возможностей программирования, системы команд, системы адресации, организации памяти и т.д. Архитектура определяет принципы действия, информационные связи и взаимное соединение основных логических узлов компьютера: процессора, оперативного ЗУ, внешних ЗУ и периферийных устройств.

Технические средства – совокупность средств человеческой деятельности создаваемых и используемых для осуществления процессов производства и обслуживания непроизводственных потребностей общества.

Основное назначение техники:

- облегчение и повышение уровня эффективности трудовых усилий человека;

- расширение его возможностей в процессе трудовой деятельности;

- освобождение (полное или частичное) человека от работы в условиях, опасных для здоровья.

Состав технических средств весьма разнообразен, но можно предложить следующую их классификацию (рис. 1.1.6), учитывающую описанное назначение техники:

- приспособления и инструменты;

- машины и механизмы;

- автоматические устройства.

Технические средства		
Приспособления и инструменты	Машины и механизмы	Автоматические устройства

Рис. 1.1.6. Классификация технических средств

В процессе общественного развития технические средства последовательно приобретали новые возможности, расширяя сферы своего применения.

Первоначально они представляли собой различные **приспособления и инструменты**, с помощью которых облегчалось выполнение трудовых операций на основе использования мускульной силы человеческого организма без применения внешних источников энергии.

Качественно иной, более высокий уровень развития технических средств представляют собой **машины и механизмы** – механические устройства, выполняющие полезную работу на основе использования внешних (по отношению к человеческому организму) источников энергии. При своей энергетической независимости машины и механизмы существенно зависят от

человека, осуществляющего управление ими. Использование машин и механизмов и той или иной сфере деятельности называется механизацией.

Следующий уровень развития технических средств представлен **автоматами** – устройствами, самостоятельно, под управлением некоторой программы, выполняющими ряд заданных операций. Их отличие от машин и механизмов состоит в том, что наряду с энергетической независимостью они обладают определенной автономностью поведения в рамках заданной программы. Использование автоматов (автоматических устройств) в той или иной сфере деятельности называется автоматизацией.

Определение состава и классификацию технических средств информационных технологий можно производить на основе приведенных общих положений о средствах и орудиях трудовой деятельности с учетом специфики предметов труда, которыми в данном случае выступают информационные объекты – данные на материально-вещественных носителях.

Во многом общие представления о средствах и орудиях трудовой деятельности сложились исходя из преобладающего энергосилового характера выполняемых операций над материальными объектами, составляющими множество предметов труда в процессе производства. Они практически без изменения могут быть применены к тем техническим средствам офисных технологий, объектами действия которых являются собственно материально-вещественные носители данных, но не сами эти данные. С учетом этого в составе технических средств достаточно просто выделить группы, относящиеся к приспособлениям и инструментам, машинам и механизмам, автоматическим устройствам.

При рассмотрении в качестве предметов трудовой деятельности собственно данных необходимо уточнить критерии отнесения тех или иных технических средств к определенной группе, поскольку речь идет уже не об энергосиловых, а об информационных преобразованиях, не о физическом, а об умственном труде.

Умственную деятельность можно определить как совокупность преобразований информации, совместно выполняемых различны ми органами человеческого организма и включающих в себя:

- восприятие данных различной формы представления (через органы чувств);
- их содержательную (семантическую) обработку в процессе мозговой деятельности;
- оперативное и долговременное хранение, реализуемое соответствующими биохимическими процессами;
- выдачу результатов посредством их представления в той или иной форме (с помощью голосовых связок, мимики, жестов, создания зрительных образов с использованием подручных средств).

Все указанные преобразования информационных объектов можно свести к трем группам:

- изменение формы представления информации (запись текста под диктовку, зачитывание вслух бумажного документа, переписывание документа и т. п.);
- изменение материального носителя данных (часто сопровождает изменение формы представления данных);
- изменение содержания (семантики) данных (реферирование документа, формирование управленческого решения и т. п.).

Достаточно очевидно, что основу умственной деятельности составляет изменение содержания данных (а зачастую и их создание), в то время как изменение их носителя и формы представления играет подчиненную, обслуживающую роль. Поэтому решение вопроса о развитии и группировке технических средств обеспечения умственного труда следует начинать именно с содержательной обработки данных. Исторически такие средства начали развиваться и применяться применительно к счетной работе.

Выполнение вычислений предполагает:

- восприятие и фиксацию исходных чисел;

- выполнение действий над ними (арифметических операций) с кратковременным (оперативным) хранением промежуточных результатов;
- отображение (представление) итоговых значений.

Разработанные для выполнения этой работы технические средства могут быть сгруппированы в зависимости от того, какие операции на них возлагаются:

1) счеты, счетные палочки, логарифмические линейки, арифмометры – относятся к инструментам и приспособлениям;
2) настольные счетные машины, счетно-перфорационная техника – относятся к машинам и механизмам;
3) компьютерная техника - относится к автоматическим устройствам.

Таким образом, применительно к техническим средствам информационных технологий с учетом изложенных соображений можно применить традиционную классификацию, предполагающую выделение приспособлений и инструментов, механизированных (механических) и автоматизированных (автоматических) устройств.

Указанная группировка технических средств является обобщенной, отражая лишь те их особенности, которые связаны со степенью их применения в тех или иных технологиях с точки зрения замены живого труда.

Более содержательной является функциональная группировка (рис. 1.1.7), отражающая целевое предназначение технических средств. В этом отношении можно выделить:

- средства организационной техники;
- средства коммуникационной техники;
- средства вычислительной (компьютерной) техники.

Технические средства информационных технологий		
Организационная техника	Коммуникационная техника	Компью-терная техника

Рис. 1.1.7. Функциональная структура технических средств

Организационная техника включает в себя различные и разнообразные средства облегчения и обеспечения офисного и инженерно-технического труда от канцелярской «мелочи» (скрепки, кнопки, ластики и т.п.) до сложнейших комплексов копировального и проекционного оборудования.

Коммуникационная техника включает в себя различные средства передачи информации (телефоны, радиосвязь, факсимильная вязь и т.д.).

Компьютерная техника включает в себя различные виды автоматических средств выполнения разнообразной обработки информации.

Вопросы для самоконтроля

1. Какие современные информационные технологии известны вам?
2. Средства и методы информационных технологий?
3. Каков принцип работы компьютера?
4. Архитектура современного компьютера?
5. Как вы понимаете термин информатизация общества?
6. Какова роль информационных технологий в развитии общества?
7. Что такое информационная культура?
8. Классификация технических средств?

1.2. Современные системы автоматизированного проектирования и их применение в технических областях

Основные модули

- Информационные системы, их значение, применение и задачи.
- Строение информационных систем и их характеристика.
- Основные процессы информационных систем.
- Составные части обеспечивающие информационные системы: техническое, программное, математическое, информационное и правовое обеспечение;
- САПР.

Система – это объективное единство закономерно связанных друг с другом предметов, явлений, сведений, а также знаний о природе, обществе и др.

Каждый объект, чтобы его можно было считать системой, должен обладать четырьмя основными свойствами:

1) целостностью и делимостью;

2) наличием устойчивых связей;

3) организацией;

4) эмерджентностью (эффект синергии).

Система – это целостное образование, однако в её составе могут быть выделены целостные объекты или элементы. Для системы первичным является признак целостности, т.е. она рассматривается как единое целое, состоящее из взаимодействующих частей, часто разнокачественных, но совместимых.

Свойство организации характеризуется наличием определённой организации, снижающей энтропию (степень неопределённости) системы по сравнению с энтропией системоформирующих факторов, определяющих возможность создания системы.

Эмерджентность предполагает наличие у системы таких свойств или качеств, которые не присущи ни одному из её элементов в отдельности. Хотя свойства системы и зависят от свойств составных её элементов, но не определяются ими полностью. Отсюда следует, что система не сводится к простой совокупности элементов и, декомпозируя систему на части, а также изучая каждую из них в отдельности, нельзя познать все свойства системы.

Одни и те же элементы в зависимости от принципа, используемого для объединения их в систему, могут образовывать различные по свойствам системы. Поэтому характеристики системы определяются не только и не столько свойствами составляющих её элементов, сколько характеристиками связей между ними. Наличие взаимосвязей и взаимодействий между элементами определяет особое свойство сложных систем – организационную сложность. Добавление элементов в систему не только вводит новые связи, но и изменяет характеристики многих или всех прежних взаимосвязей, приводит к исключению некоторых из них или появлению новых.

Информационная система (ИС)– взаимосвязанная совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели.

Структуру информационной системы составляет совокупность отдельных ее частей, называемых подсистемами.

Подсистема – это часть системы, выделенная по какому-либо признаку.

Общую структуру информационной системы можно рассматривать как совокупность подсистем независимо от сферы применения. В этом случае говорят о структурном признаке классификации, а подсистемы называют обеспечивающими. Таким образом, структура любой информационной системы может быть представлена совокупностью обеспечивающих подсистем.

Среди обеспечивающих подсистем обычно выделяют информационное, техническое, математическое, программное, организационное и правовое обеспечение.

Информационное обеспечение

Назначение подсистемы информационного обеспечения состоит в современном формировании и выдаче достоверной информации для принятия управленческих решений.

Информационное обеспечение – совокупность единой системы классификации и кодирования информации, унифицированных систем документации, схем информационных потоков, циркулирующих в организации, а также методология построения баз данных.

Техническое обеспечение

Техническое обеспечение – комплекс технических средств, предназначенных для работы информационной системы, а также соответствующая документация на эти средства и технологические процессы

Комплекс технических средств составляют:

- компьютеры любых моделей;
- устройства сбора, накопления, обработки, передачи и вывода информации;
- устройства передачи данных и линий связи;
- оргтехника и устройства автоматического съема информации;
- эксплуатационные материалы и др.

Документацией оформляются предварительный выбор технических средств, организация их эксплуатации, технологический процесс обработки данных, технологическое оснащение. Документацию можно условно разделить на три группы:

- общесистемную, включающую государственные и отраслевые стандарты по техническому обеспечению;
- специализированную, содержащую комплекс методик по всем этапам разработки технического обеспечения;
- нормативно-справочную, используемую при выполнении расчетов по техническому обеспечению.

К настоящему времени сложились две основные формы организации технического обеспечения (формы использования технических средств): централизованная и частично или полностью децентрализованная.

Централизованное техническое обеспечение базируется на использовании в информационной системе больших ЭВМ и вычислительных центров.

Децентрализация технических средств предполагает реализацию функциональных подсистем на персональных компьютерах непосредственно на рабочих местах.

Математическое и программное обеспечение

Математическое и программное обеспечение – совокупность математических методов, моделей, алгоритмов и программ для реализации целей и задач информационной системы, а также нормального функционирования комплекса технических средств.

К средствам математического обеспечения относятся:

- средства моделирования процессов управления;
- типовые задачи управления;
- методы математического программирования, математической статистики, теории массового обслуживания и др.

В состав программного обеспечения входят общесистемные и специальные программные продукты, а также техническая документация.

К общесистемному программному обеспечению относятся комплексы программ, ориентированных на пользователей и предназначенных для решения типовых задач обработки информации. Они служат для расширения функциональных возможностей компьютеров, контроля и управления процессом обработки данных.

Специальное программное обеспечение представляет собой совокупность программ, разработанных при создании конкретной информационной системы. В его состав входят пакеты прикладных программ

(ППП), реализующие разработанные модели разной степени адекватности, отражающие функционирование реального объекта.

Техническая документация на разработку программных средств должна содержать описание задач, задание на алгоритмизацию, экономико-математическую модель задачи, контрольные примеры.

Организационное обеспечение

Организационное обеспечение – совокупность методов и средств, регламентирующих взаимодействие работников с техническими средствами и между собой в процессе разработки и эксплуатации информационной системы.

Организационное обеспечение реализует следующие функции:

· анализ существующей системы управления организацией, где будет использоваться ИС, и выявление задач, подлежащих автоматизации;

· подготовку задач к решению на компьютере, включая техническое задание на проектирование ИС и технико-экономическое обоснование ее эффективности;

· разработку управленческих решений по составу и структуре организации, методологии решения задач, направленных на повышение эффективности системы управления.

Правовое обеспечение

Правовое обеспечение – совокупность правовых норм, определяющих создание, юридический статус и функционирование информационных систем, регламентирующих порядок получения, преобразования и использования информации.

Главной целью правового обеспечения является укрепление законности.

В состав правового обеспечения входят законы, указы, постановления государственных органов власти, приказы, инструкции и другие нормативные документы министерств, ведомств, организаций, местных органов власти. В правовом обеспечении можно выделить общую часть,

регулирующую функционирование любой информационной системы, и локальную часть, регулирующую функционирование конкретной системы.

Правовое обеспечение этапов разработки информационной системы включает нормативные акты, связанные с договорными отношениями разработчика и заказчика и правовым регулированием отклонений от договора.

Правовое обеспечение этапов функционирования информационной системы включает:

- статус информационной системы;
- права, обязанности и ответственность персонала;
- правовые положения отдельных видов процесса управления;
- порядок создания и использования информации и др.

В настоящее время существует ряд задач, которые не могут быть решены традиционными методами теории автоматического управления. Это характерно для больших и сложных объектов и систем, алгоритмы работы которых не могут быть формализованы или работают в неопределенных ситуациях. Как правило, управляют такими объектами и системами люди (человек-оператор), эксперты по принятию решений в данной области. Такой тип систем управления называют системами управления на основе знаний или интеллектуальными системами управления.

Автоматизированная информационная система представляет собой совокупность информации, экономико-математических методов и моделей, технических, программных, технологических средств и специалистов, предназначенную для обработки информации и принятия управленческих решений.

Интеллектуальная система (ИС) – автоматизированная система, основанная на знаниях, или комплекс программных, лингвистических и логико-математических средств для реализации основной задачи – осуществления поддержки деятельности человека и поиска информации в режиме продвинутого диалога на естественном языке. Кроме того,

информационно-вычислительными системами с интеллектуальной поддержкой для решения сложных задач называют те системы, в которых логическая обработка информации превалирует над вычислительной. Таким образом, любая информационная система, решающая интеллектуальную задачу или использующая методы искусственного интеллекта, относится к интеллектуальным.

Системы автоматизированного проектирования (САПР) – комплекс программных и аппаратных средств, предназначенных для автоматизации процесса проектирования человеком технических изделий или продуктов интеллектуальной деятельности.

Проектирование новых изделий – основная задача изобретателей конструкторов, протекает в несколько этапов, таких, как нормирование замысла, поиск физических принципов, обеспечивающих реализацию замыслов и требуемые значения конструкции, поиск конструктивных решений, их расчет и обоснование, создание опытного образца, разработка технологий промышленного изготовления. Если формирование замысла и поиск физических принципов пока остаются чисто творческими, не поддающимися автоматизации этапами, то при конструировании и расчетах с успехом могут быть применены САПР (рис. 1.2.1).

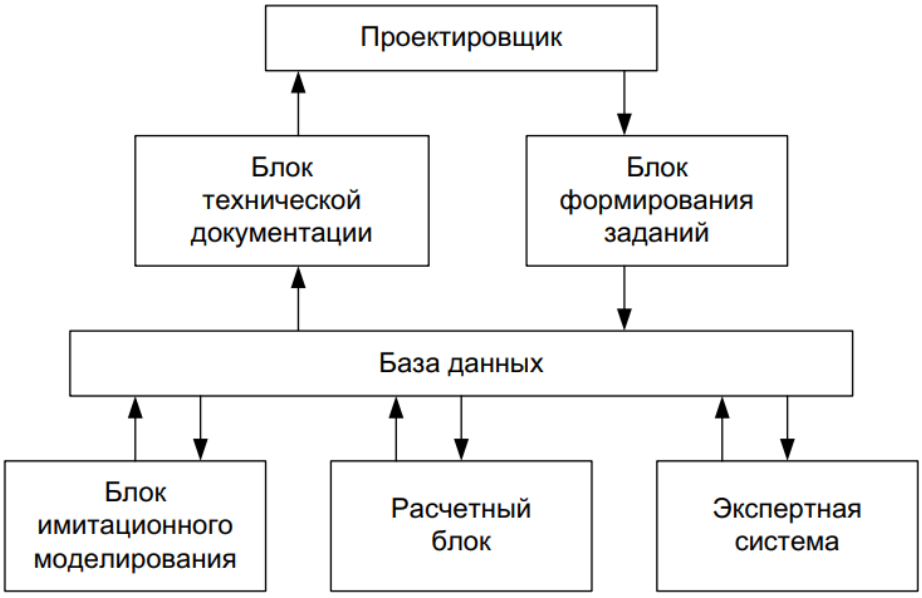


Рис.1.2.1. Типовая схема САПР

База данных, блок имитационного моделирования, расчетный блок и экспертная система выполняют функции, аналогичные функциям соответствующих блоков АСНИ. Вместо блока связи с измерительной аппаратурой в САПР имеется *блок формирования заданий*. Проектировщик вводит в блок техническое задание на проектирование, в нем указаны цели, которые необходимо достичь при проектировании, и все ограничения, которые нельзя нарушить.

Блок подготовки технической документации облегчает создание технической документации для последующего изготовления изделия.

Аппаратное обеспечение САПР составляет ЭВМ с набором устройств, необходимых для ввода и вывода графической информации (графопостроитель, световое перо, графический планшет и др.).

В настоящее время САПР является неотъемлемым атрибутом крупных конструкторских бюро и проектных организаций, работающих в различных предметных областях. Это важная сфера приложения идей и методов информатики. САПР широко применяется в архитектуре, электротехнике, электронике, машиностроении, авиакосмической технике и др.

Целью функционирования САПР является проектирование.

Проектирование – это процесс переработки информации, приводящий в конечном счете к получению полного представления о проектируемом объекте и способах его изготовления.

Проектирование – это один из наиболее сложных видов интеллектуальной работы, выполняемой человеком. Проектирование сложных объектов выполняется творческим коллективом, поэтому он становится более сложным и трудно поддающимся формализации. Для автоматизации такого процесса необходимо четко знать, что в действительности он собой представляет и как выполняется разработчиками.

Объектом автоматизации проектирования являются работы, действия человека, которые он выполняет в процессе проектирования. А то, что проектируют, называют ***объектом проектирования***.

Разделяют САПР изделия и САПР технологических процессов.

Объектом автоматизации проектирования является вся совокупность действий проектировщиков, разрабатывающих изделие или технологический процесс, или то и другое, и оформляющих результаты разработок в виде конструкторской, технологической и эксплуатационной документаций.

Разделив весь процесс проектирования на этапы и операции, можно описать их с помощью определенных математических методов и определить инструментальные средства для их автоматизации. Затем необходимо рассмотреть выделенные проектные операции и средства автоматизации в комплексе и найти способы сопряжения их в единую систему, отвечающую поставленным целям.

В целом для всех этапов проектирования изделий и технологии их изготовления можно выделить следующие основные виды типовых операций обработки информации:

- поиск и выбор из всевозможных источников нужной информации;
- анализ выбранной информации;
- выполнение расчетов;
- принятие проектных решений;
- оформление проектных решений в виде, удобном для дальнейшего использования (на последующих стадиях проектирования, при изготовлении или эксплуатации изделия).

Сущность функционирования современных САПР составляет автоматизация перечисленных операций обработки информации и процессов управления использованием информации на всех стадиях проектирования.

Характерной особенностью САПР является:

- возможность *комплексного* решения общей задачи проектирования, установления тесной связи между частными задачами;

- *интерактивный режим* проектирования, при котором осуществляется непрерывный процесс *диалога* "человек-машина";

- возможность *имитационного моделирования* в условиях работы, близких к реальным, что дает возможность предвидеть реакцию проектируемого объекта на самые различные возмущения, позволяет конструктору "видеть" плоды своего труда в действии без макетирования;

- значительное усложнение программного и информационного обеспечения проектирования. Количественное, объемное увеличение, идеологическое усложнение, связанное с необходимостью создания языков общения проектировщика и ЭВМ, развитых банков данных, программ информационного обмена между составными частями системы, программ проектирования.

При создании САПР руководствуются следующими семью общесистемными, чрезвычайно важными на этапе разработки, принципами:

- *включения* – состоит в том, что требования к созданию, функционированию и развитию САПР определяются со стороны более сложной системы, включающей в себя САПР в качестве подсистемы.
- *системного единства* – предусматривает обеспечение целостности САПР за счет связи между ее подсистемами и функционирования подсистемы управления САПР.
- *комплексности* – требует связности проектирования отдельных элементов и всего объекта в целом на всех стадиях проектирования.
- *информационного единства* – предопределяет информационную согласованность отдельных подсистем и компонентов САПР.
- *совместимости* – состоит в том, что языки, коды, информационные и технические характеристики структурных связей между подсистемами и компонентами САПР должны быть согласованы так, чтобы обеспечить совместное функционирование всех подсистем и сохранить *открытую структуру* САПР в целом.

- *инвариантности* – предусматривает, что подсистемы и компоненты САПР должны быть по возможности универсальными или типовыми, т.е. инвариантными к проектируемым объекта*м* и отраслевой специфике.
- *развития* – требует, чтобы в САПР предусматривалось наращивание и совершенствование компонентов и связей между ними.

Вопросы для самоконтроля

1. Дайте понятие «информационная система».
2. Какими свойствами обладают системы?
3. Процессы в информационных системах.
4. Основные правила функциональной декомпозиции систем.
5. Основные составляющие информационных систем/
6. Дайте определение САПР.
7. Что является целью функционирования САПР?

1.3. Экспертные системы и их программное обеспечение. Технология использования и создания экспертных систем.

Основные модули

- Экспертные системы.
- Искусственный интеллект.
- Типы экспертных систем.
- Виды знаний.
- Области применения экспертных систем.

Наибольший прогресс среди компьютерных информационных систем отмечен в области разработки экспертных систем (ЭС), основанных на использовании элементов искусственного интеллекта. Экспертные системы дают возможность менеджеру или специалисту получать консультации

экспертов по любым проблемам, на основе которых этими системами накоплены знания.

Под ***искусственным интеллектом*** (ИИ) обычно понимают способности компьютерных систем к таким действиям, которые назывались бы интеллектуальными, если бы исходили от человека. Чаще всего здесь имеются в виду способности, связанные с человеческим мышлением. Работы в области искусственного интеллекта не ограничиваются экспертными системами. Они также включают в себя создание роботов, систем, моделирующих нервную систему человека, его слух, зрение, обоняние, способность к обучению.

Решение специальных задач требует специальных знаний. Главная идея использования технологии экспертных систем заключается в том, чтобы получить от эксперта его знания и, загрузив их в память компьютера, использовать всякий раз, когда в этом возникнет необходимость. Являясь одним из основных приложений искусственного интеллекта, экспертные системы представляют собой компьютерные программы, трансформирующие опыт экспертов в какой-либо области знаний в форму эвристических правил. На практике ЭС используются прежде всего как системы-советчики в тех ситуациях, где специалист сомневается в выборе правильного решения. Экспертные знания, хранящиеся в памяти системы, более глубокие и полные, чем соответствующие знания пользователя.

ЭС находят распространение при решении задач с принятием решений в условиях неопределенности (неполноты) для распознавания образов, в прогнозировании, диагностике, планировании, управлении, конструировании и т.д.

Типичная экспертная система состоит из решателя (интерпретатора), БД (базы данных), БЗ (базы знаний), компонентов приобретения знаний, объяснительного и диалогового компонентов.

БД предназначена для хранения исходных и промежуточных данных, используемых для решения задач, фактографических данных.

Решатель, используя исходные данные из БД и знания из БЗ, обеспечивает решение задач для конкретных ситуаций.

Компонент приобретения знаний автоматизирует процесс наполнения БЗ.

Объяснительный компонент объясняет, как система получила решение задачи (или почему не получила) и какие знания она при этом использовала. Диалоговый компонент обеспечивает диалог между экспертной системой и пользователем в процессе решения задачи и приобретения знаний.

Экспертные системы создаются для решения разного рода задач профессиональной деятельности человека, и в зависимости от этого выполняют разные функции.

Типы экспертных систем

Можно назвать несколько *типов современных экспертных систем*.

1. Экспертные системы первого поколения. Предназначены для решения хорошо структурированных задач, требующих небольшого объема эмпирических знаний. Сюда относятся классификационные задачи и задачи выбора из имеющегося набора вариантов.
2. Оболочки ЭС. Имеют механизм ввода-вывода, но БЗ пустая. Требуется настройка на конкретную предметную область. Знания приобретаются в процессе функционирования ЭС, способной к самообучению.
3. Гибридные ЭС. Предназначены для решения различных задач с использованием БЗ. Это задачи с использованием методов системного анализа, исследования операций, математической статистики, обработки информации. Пользователь имеет доступ к объективизированным знаниям, содержащимся в БЗ и пакетах прикладных программ.
4. Сетевые ЭС. Между собой связаны несколько экспертных систем. Результаты решения одной из них являются исходными данными для другой системы. Эффективны при распределенной обработке информации.

При разработке экспертных систем должны участвовать: эксперт той предметной области, задачи которой будет решать система; инженер по знаниям – специалист по разработкам систем; программист – специалист по разработке инструментальных средств. Эксперт определяет знания, то есть описывает предметную область в виде совокупности данных и правил, обеспечивает полноту и правильность введенных в экспертную систему знаний. Данные определяют объекты, их характеристики и значения. Правила указывают на способы манипулирования данными.

Инженер по знаниям помогает эксперту: выявить и структурировать знания, необходимые для функционирования экспертной системы; осуществить выбор инструментальных средств, которые наиболее эффективны для решения задач в данной предметной области; указать способы представления знаний. Программист разрабатывает инструментальную среду, включающую все компоненты экспертной системы, производит ее сопряжение с другими существующими системами.

Виды знаний

1. *Понятийные знания.* Это набор понятий, которыми пользуется ЛПР, работающий в некоторой области интеллектуальной, управляющей деятельности, а также свойства и взаимосвязи этих понятий. Эта категория знаний в основном вырабатывается в сфере фундаментальных наук.

2. *Конструктивные знания* (близкие к понятийным знаниям). Это знания о структуре и взаимодействии частей различных объектов. Они в основном составляют содержание технических, прикладных наук. К примеру, если взять программирование, то понятийное знание – знание о структуре операторов, данных, языка программирования. Конструктивное знание – это знание об устройстве конкретных программ, о типичных алгоритмах.

3. *Процедурные знания.* К ним относятся методические правила решения различных задач, с которыми ЛПР уже сталкивался и их решать. В производственной сфере аналогом процедурных знаний являются технологические знания различных производственных процессов.

Процедурные знания – это опыт интеллектуальной, управляющей деятельности ЛПР в определенной предметной области.

4. *Фактографические знания*. Они включают в себя количественные и качественные характеристики конкретных объектов, явлений и их элементов. Их накопление ведется в виде таблиц, справочников, файлов, баз данных (БД).

Способы формализованного представления знаний в базе знаний (БЗ). Формализованное представление знаний в информационных технологиях управления в виде интеллектуальных систем является первичным. Рассмотрим распространенные способы их формализованного представления:

1. **Представление знаний продукционными правилами.** Продукционные правила представляют знания в форме ЕСЛИ - ТО. Системы, использующие представления знаний продукционными правилами, называются продукционными. Это самый наглядный и простой способ. В таких системах представления знаний имеются средства, позволяющие использовать в данных и правилах нечеткую информацию с определенной вероятностью, называемой фактором уверенности.

2. **Логика предикатов (раздел математической логики).** Константы и переменные определяют отдельные объекты и обозначаются буквами или набором букв (U, V, W, X, Y). Последовательность из n констант или переменных (n – конечно, n > 1) называется функцией. Атомарным предикатом называется последовательность из n сущностей и понятий, описанных константами, переменными или функциями.

Предикат принимает одно из двух значений: истина или ложь. Предикат, в котором все переменные, константы и функции связаны между собой, называется предложением. Предложения используются для представления знаний. Логика предикатов обеспечивает высокий уровень модульности знаний (представляет их как единое целое в определенной предметной области) и позволяет выяснить, имеются или отсутствуют

противоречия между новыми и уже существующими знаниями. Но чрезмерный уровень формализации представления знаний, трудность их прочтения снижают эффективность обработки. Кроме этого, в логике предикатов все отношения описываются предикатами, что не позволяет при компьютерной обработке полностью отразить свойства структуры данных. Для программирования используется язык логического типа ПРОЛОГ.

3. Модель доски объявлений. Модель представляется как совокупность отдельных проблем, каждая из которых составляет отдельное множество знаний. Все множества модели используются согласованно как единое целое и управляются через общую рабочую область памяти, называемую доской объявлений. Отдельное множество знаний называется источником знаний (ИЗ), и каждый ИЗ строится как продукционная система.

4. Семантические сети. Знания можно рассматривать как отношения между понятиями и сущностями, являющимися конкретными объектами реального мира. Понятия и отношения можно представить в виде семантической сети, состоящей из вершин и дуг. В вершинах располагаются понятия, а направленные связи между вершинами соответствуют различного рода отношениям между этими понятиями. Семантические сети могут быть выполнены обучаемыми и растущими, что означает возможность автоматического добавления в сеть новых узлов по мере появления в опыте ее использования новых понятий, а также увеличение весовых коэффициентов, соответствующих дугам. В процессе ее обучения между существующими узлами также могут устанавливаться дополнительные связи.

5. Фреймовые системы. Фреймы рассматриваются как структура описания отдельной сущности или понятия. Они могут быть в виде их совокупностей, представляемых как отдельное множество знаний, относящихся к одному объекту. Каждый фрейм состоит из множества элементов, называемых слотами, которые в свою очередь представляются определенной структурой данных. Каждый фрейм и слот имеют имя,

единственное во всей фреймовой системе. В значение слота содержит конкретную информацию.

Фреймы не связаны в сеть. Управление большим числом источников знаний выполняется самим пользователем путем вызова нужных процедур (в других способах это выполняет сама система). Для поиска нужного объекта задаются значения слотов. Если данные удовлетворяют условиям всех слотов, то объект считается найденным.

Области применения экспертных систем

ЭС в задачах интерпретации, как правило, используют информацию от датчиков для описания ситуации. В качестве примера приведем интерпретацию показаний измерительных приборов на химическом заводе для определения состояния процесса. Интерпретирующие системы имеют дело не с четкими символьными представлениями проблемной ситуации, а непосредственно с реальными данными. Они сталкиваются с затруднениями, которых нет у систем других типов, потому что им приходится обрабатывать информацию зашумленную, недостаточную, неполную, ненадежную или ошибочную. Им необходимы специальные методы регистрации характеристик непрерывных потоков данных, сигналов или изображений и методы их символьного представления.

Интерпретирующие ЭС могут обработать разнообразные виды данных. Например, системы анализа сцен и распознавания речи, используя естественную информацию, - в одном случае визуальные образы, в другом – звуковые сигналы, - анализируют их характеристики и понимают их смысл. Интерпретация в области химии использует данные дифракции рентгеновских лучей, спектрального анализа или ядерно-магнитного резонанса для вывода химической структуры веществ. Интерпретирующая система в геологии использует каротажное зондирование – измерение проводимости горных пород в буровых скважинах и вокруг них, - чтобы определить подповерхностные геологические структуры. Медицинские интерпретирующие системы используют показания следящих систем

(например, значения пульса, кровяного давления), чтобы установить диагноз или тяжесть заболевания. Наконец, в военном деле интерпретирующие системы используют данные от радаров, радиосвязи и сонарных устройств, чтобы оценить ситуацию и идентифицировать цели.

ЭС в задачах прогнозирования определяют вероятные последствия заданных ситуаций. Примерами служат прогноз ущерба урожаю от некоторого вида вредных насекомых, оценивание спроса на нефть на мировом рынке в зависимости от складывающейся геополитической ситуации и прогнозирование места возникновения следующего вооруженного конфликта на основании данных разведки. Системы прогнозирования иногда используют имитационное моделирование, т.е. программы, которые отражают причинно-следственные взаимосвязи в реальном мире, чтобы сгенерировать ситуации или сценарии, которые могут возникнуть при тех или иных входных данных. Эти возможные ситуации вместе со знаниями о процессах, порождающих эти ситуации, образуют предпосылки для прогноза.

ЭС в задачах диагностики используют описания ситуаций, характеристики поведения или знания о конструкции компонент, чтобы установить вероятные причины неправильного функционирования диагностируемой системы. Примерами служат: определение причин заболевания по симптомам, наблюдаемым у пациентов; локализация неисправностей в электронных схемах и определение неисправных компонент в системе охлаждения ядерных реакторов. Диагностические системы часто являются консультантами, которые не только ставят диагноз, но также помогают в отладке. Они могут взаимодействовать с *пользо*вателем, чтобы оказать помощь при поиске неисправностей, а затем предложить порядок действий по их устранению. Медицина представляется вполне естественной областью для диагностирования, и действительно, в медицинской области было разработано больше диагностических систем, чем в любой другой отдельно взятой предметной области.

ЭС, применяемые в области проектирования, разрабатывают конфигурации объектов с учетом набора ограничений, присущих проблеме. Учитывая то, что проектирование столь тесно связано с планированием, многие проектирующие системы содержат механизмы разработки и уточнения планов для достижения желаемого проекта. Наиболее часто встречающиеся области применения планирующих ЭС - химия, электроника и военное дело.

ЭС, которые используются для решения задач наблюдения, сравнивают действительное поведение с ожидаемым поведением системы. Примерами могут служить слежение за показаниями измерительных приборов в ядерных реакторах с целью обнаружения аварийных ситуаций или оценку данных мониторинга больных, помещенных в блоки интенсивной терапии. Наблюдающие ЭС подыскивают наблюдаемое поведение, которое подтверждает их ожидания относительно нормального поведения или их предположения о возможных отклонениях. Наблюдающие ЭС по самой своей природе должны работать в режиме реального времени.

ЭС в задачах отладки находят рецепты для исправления неправильного поведения устройств. Примерами могут служить настройка компьютерной системы с целью преодолеть некоторый вид затруднений в ее работе; выбор типа обслуживания, необходимого для устранения неисправностей в телефонном кабеле; выбор ремонтной операции для исправления известной неисправности в насосе.

ЭС в задачах ремонта аппаратуры следуют плану, который предписывает некоторые рецепты восстановления. Примером является настройка масс-спектрометра, т.е. установка ручек регулировки прибора в положение, обеспечивающее достижение оптимальной чувствительности, совместимой с правильным отношением величин пиков и их формы. Пока что было разработано очень мало ремонтных ЭС отчасти потому, что необходимость фактического выполнения ремонтных процедур на объектах реального мира дополнительно усложняет задачу. Ремонтным системам

также необходимы диагностирующие, отлаживающие и планирующие процедуры для производства ремонта.

ЭС в области обучения подвергают диагностике, "отладке" и исправлению ("ремонту") поведение обучаемого. В качестве примеров приведем обучение студентов отысканию неисправностей в электрических цепях, обучение военных моряков обращению с двигателем на корабле и обучение студентов-медиков выбору антимикробной терапии. Обучающие системы создают модель того, что обучающийся знает и как он эти знания применяет к решению проблемы. Системы диагностируют и указывают обучающемуся его ошибки, анализируя модель и строя планы исправлений указанных ошибок. Они исправляют поведение обучающихся, выполняя эти планы с помощью непосредственных указаний обучающимся.

ЭС в задачах управления адаптивно руководят поведением системы в целом. Примерами служат управление производством и распределением компьютерных систем или контроль за состоянием больных при интенсивной терапии. Управляющие ЭС должны включать наблюдающие компоненты, чтобы отслеживать поведение объекта на протяжении времени, но они могут нуждаться также и в других компонентах для выполнения любых или всех из уже рассмотренных типов задач: интерпретации, прогнозирования, диагностики, проектирования, планирования, отладки, ремонта и обучения. Типичная комбинация задач состоит из наблюдения, диагностики, отладки, планирования и прогноза.

Вопросы для самоконтроля

1. Экспертные системы: назначение, характеристики и основные
2. компоненты.
3. Типы экспертных систем
4. База знаний: назначение и основные виды.

ГЛАВА II. МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

2.1. Классификация видов моделирования технических систем

Основные модули

- Понятие модели и моделирования
- Классификационные признаки моделирования
- Эффективность моделирования систем
- Виды моделирования
- Примеры видов моделирования

Модель – материальный объект, система математических зависимостей или программа, имитирующая структуру или функционирование исследуемого объекта.

Моделирование – представление различных характеристик поведения физической или абстрактной системы с помощью другой системы.

В основе моделирования лежит теория подобия, которая утверждает, что абсолютное подобие может иметь место лишь при замене одного объекта другим точно таким же. При моделировании абсолютное подобие не имеет места и стремятся к тому, чтобы модель достаточно хорошо отображала исследуемую сторону функционирования объекта.

В качестве одного из первых признаков классификации видов моделирования можно выбрать степень полноты модели и разделить модели в соответствии с этим признаком на: полные, неполные, приближенные.

В основе полного моделирования лежит полное подобие, которое проявляется как во времени, так и в пространстве.

Для неполного моделирования характерно неполное подобие модели изучаемому объекту.

В основе приближенного моделирования лежит приближенное подобие, при котором некоторые стороны функционирования реального объекта не моделируются совсем. Классификационные признаки моделирования систем приведены на рис. 2.1.1.

Рис. 2.1.1. Классификационные признаки моделирования систем

В зависимости от характера изучаемых процессов в системе все виды моделирования могут быть разделены:

- детерминированные;
- стохастические;

- статические и динамические;
- дискретные;
- непрерывные;
- дискретно-непрерывные.

Детерминированное моделирование отображает детерминированные процессы, т.е. процессы, в которых предполагается отсутствие всяких случайных воздействий.

Стохастическое моделирование отображает вероятностные процессы и события. В этом случае анализируется ряд реализаций случайного процесса, и оцениваются средние характеристики, т. е. набор однородных реализаций.

Статическое моделирование служит для описания поведения объекта в какой-либо момент времени, а динамическое моделирование отражает поведение объекта во времени.

Дискретное моделирование служит для описания процессов, которые предполагаются дискретными, соответственно непрерывное моделирование позволяет отразить непрерывные процессы в системах, а дискретно-непрерывное моделирование используется для случаев, когда хотят выделить наличие как дискретных, так и непрерывных процессов.

В зависимости от формы представления объекта можно выделить мысленное и реальное моделирование.

Мысленное моделирование часто является единственным способом моделирования объектов, которые либо практически нереализуемы в заданном интервале времени, либо существуют вне условий, возможных для их физического создания. Например, на базе мысленного моделирования могут быть проанализированы многие ситуации микромира, которые не поддаются физическому эксперименту.

Пример *Дискретное моделирование*

- *Мысленное моделирование может быть реализовано в виде:*
- *наглядного;*
- *символического;*

• *математического.*

При наглядном моделировании на базе представлений человека о реальных объектах создаются различные наглядные модели, отображающие явления и процессы, протекающие в объекте.

1) *В основу гипотетического моделирования* исследователем закладывается некоторая гипотеза о закономерностях протекания процесса в реальном объекте, которая отражает уровень знаний исследователя об объекте и базируется на причинно-следственных связях между входом и выходом изучаемого объекта. Гипотетическое моделирование используется, когда знаний об объекте недостаточно для построения формальных моделей.

2) *Аналоговое моделирование* основывается на применении аналогий различных уровней. Наивысшим уровнем является полная аналогия, имеющая место только для достаточно простых объектов. С усложнением объекта используют аналогии последующих уровней, когда аналоговая модель отображает несколько либо только одну сторону функционирования объекта.

3) Существенное место при мысленном наглядном моделировании занимает *макетирование*. Мысленный макет может применяться в случаях, когда протекающие в реальном объекте процессы не поддаются физическому моделированию, либо может предшествовать проведению других видов моделирования.

В основе построения мысленных макетов также лежат аналогии, однако обычно базирующиеся на причинно-следственных связях между явлениями и процессами в объекте. Если ввести условное обозначение отдельных понятий, т. е. *знаки*, а также определенные операции между этими знаками, то можно реализовать *знаковое моделирование* и с помощью знаков отображать набор понятий — составлять отдельные цепочки из слов и предложений. Используя операции объединения, пересечения и дополнения

теории множеств, можно в отдельных символах дать описание какого-то реального объекта.

В основе языкового моделирования лежит некоторый тезаурус. Последний образуется из набора входящих понятий, причем этот набор должен быть фиксированным. Следует отметить, что между тезаурусом и обычным словарем имеются принципиальные различия.

• *Тезаурус* - словарь, который очищен от неоднозначности, т. е. в нем каждому слову может соответствовать лишь единственное понятие, хотя в обычном словаре одному слову могут соответствовать несколько понятий.

• *Символическое моделирование* представляет собой искусственный процесс создания логического объекта, который замещает реальный и выражает основные свойства его отношений с помощью определенной системы знаков или символов.

• *Математическое моделирование*. Для исследования характеристик процесса функционирования любой системы S математическими методами, включая и машинные, должна быть проведена формализация этого процесса, т. е. построена математическая модель.

Под математическим моделированием будем понимать процесс установления соответствия данному реальному объекту некоторого математического объекта, называемого математической моделью, и исследование этой модели, позволяющее получать характеристики рассматриваемого реального объекта. Вид математической модели зависит как от природы реального объекта, так и задач исследования объекта и требуемой достоверности и точности решения этой задачи. Любая математическая модель, как и всякая другая, описывает реальный объект лишь с некоторой степенью приближения к действительности. Математическое моделирование для исследования характеристик процесса функционирования систем можно разделить на:

- аналитическое;
- имитационное;

- комбинированное.

Для аналитического моделирования характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений или логических условий. Аналитическая модель может быть исследована следующими методами:

а) аналитическим, когда стремятся получить в общем виде явные зависимости для искомых характеристик;

б) численным, когда, не умея решать уравнений в общем виде, стремятся получить числовые результаты при конкретных начальных данных;

в) качественным, когда, не имея решения в явном виде, можно найти некоторые свойства решения.

При имитационном моделировании реализующий модель алгоритм воспроизводит процесс функционирования системы во времени, причем имитируются элементарные явления, составляющие процесс, с сохранением их логической структуры и последовательности протекания во времени, что позволяет по исходным данным получить сведения о состояниях процесса в определенные моменты времени, дающие возможность оценить характеристики системы.

Основным преимуществом имитационного моделирования по сравнению с аналитическим является возможность решения более сложных задач. Имитационные модели позволяют достаточно просто учитывать такие факторы, как наличие дискретных и непрерывных элементов, нелинейные характеристики элементов системы, многочисленные случайные воздействия и др., которые часто создают трудности при аналитических исследованиях. В настоящее время имитационное моделирование — наиболее эффективный метод исследования больших систем, а часто и единственный практически доступный метод получения информации о поведении системы, особенно на этапе ее проектирования.

Комбинированное (аналитико-имитационное) моделирование при анализе и синтезе систем позволяет объединить достоинства аналитического

и имитационного моделирования. При построении комбинированных моделей проводится предварительная декомпозиция процесса функционирования объекта на составляющие подпроцессы =, и для тех из них, где это возможно, используются аналитические модели, а для остальных подпроцессов строятся имитационные модели. Такой комбинированный подход позволяет охватить качественно новые классы систем, которые не могут быть исследованы с использованием только аналитического и имитационного моделирования в отдельности.

При реальном моделировании используется возможность исследования различных характеристик либо на реальном объекте целиком, либо на его части. Такие исследования могут проводиться как на объектах, работающих в нормальных режимах, так и при организации специальных режимов для оценки интересующих исследователя характеристик. Реальное моделирование является наиболее адекватным, но при этом его возможности с учетом особенностей реальных объектов ограничены.

Натурным моделированием называют проведение исследования на реальном объекте с последующей обработкой результатов эксперимента на основе теории подобия.

С развитием техники и проникновением в глубь процессов, протекающих в реальных системах, возрастает техническая оснащенность современного научного эксперимента. Он характеризуется широким использованием средств автоматизации проведения, применением весьма разнообразных средств обработки информации, возможностью вмешательства человека в процесс проведения эксперимента, и в соответствии с этим появилось новое научное направление – *автоматизация научных экспериментом.*

Отличие эксперимента от реального протекания процесса заключается в том, что в нем могут появиться отдельные критические ситуации и определяться границы устойчивости процесса.

Другим видом реального моделирования является *физическое,* отличающееся от натурного тем, что исследование проводится на установках, которые сохраняют природу явлений и обладают физическим подобием. В процессе физического моделирования задаются некоторые характеристики внешней среды и исследуется поведение либо реального объекта, либо его модели при заданных или создаваемых искусственно воздействиях внешней среды. Физическое моделирование может протекать в реальном и нереальном (псевдореальном) масштабах времени, а также может рассматриваться без учета времени. В последнем случае изучению подлежат так называемые «замороженные» процессы, которые фиксируются в некоторый момент времени. Наибольшие сложность и интерес с точки зрения верности получаемых результатов представляет физическое моделирование в реальном масштабе времени.

С точки зрения *математического описания* объекта и в зависимости от его характера модели можно разделить на:

• аналоговые (непрерывные);

• цифровые (дискретные);

• аналого-цифровые (комбинированные).

Под аналоговой моделью понимается модель, которая описывается уравнениями, связывающими непрерывные величины.

• *Под цифровой понимают* модель, которая описывается уравнениями, связывающими дискретные величины, представленные в цифровом виде.

• *Под аналого-цифровой понимается модель*, которая может быть описана уравнениями, связывающими непрерывные и дискретные величины.

Особое место в моделировании занимает *кибернетическое моделирование*, в котором отсутствует непосредственное подобие физических процессов, происходящих в моделях, реальным процессам. В этом случае стремятся отобразить лишь некоторую функцию и рассматривают реальный объект как «черный ящик», имеющий ряд входов и выходов, и моделируют некоторые связи между выходами и входами.

Чаще всего при использовании кибернетических моделей проводят анализ поведенческой стороны объекта при различных воздействиях внешней среды.

Таким образом, в основе кибернетических моделей лежит отражение некоторых информационных процессов управления, что позволяет оценить поведение реального объекта. Для построения имитационной модели в этом случае необходимо выделить исследуемую функцию реального объекта, попытаться формализовать эту функцию в виде некоторых операторов связи между входом и выходом и воспроизвести на имитационной модели данную функцию, причем на базе совершенно иных математических соотношений и, естественно, иной физической реализации процесса.

Эффективность моделирования систем

Обеспечение требуемых показателей качества функционирования больших систем, связанное с необходимостью изучения протекания стохастических процессов в исследуемых и проектируемых системах, позволяет проводить комплекс теоретических и экспериментальных исследований, взаимно дополняющих друг друга.

Эффективность экспериментальных исследований сложных систем оказывается крайне низкой, поскольку проведение натурных экспериментов с реальной системой либо требует больших материальных затрат и значительного времени, либо вообще практически невозможно. Эффективность теоретических исследований с практической точки зрения в полной мере проявляется лишь тогда, когда их результаты с требуемой степенью точности и достоверности могут быть представлены в виде аналитических соотношений или моделирующих алгоритмов, пригодных для получения соответствующих характеристик процесса функционирования исследуемых систем.

Обычно модель строится по иерархическому принципу, когда последовательно анализируются отдельные стороны функционирования

объекта и при перемещении центра внимания исследователя рассмотренные ранее подсистемы переходят во внешнюю среду. Иерархическая структура моделей может раскрывать и ту последовательность, в которой изучается реальный объект, а именно последовательность перехода от структурного (топологического) уровня к функциональному (алгоритмическому) и от функционального к параметрическому.

Результат моделирования в значительной степени зависит от адекватности исходной концептуальной (описательной) модели, от полученной степени подобия описания реального объекта, числа реализаций модели и многих других факторов. В ряде случаев сложность объекта не позволяет не только построить математическую модель объекта, но и дать достаточно близкое кибернетическое описание, и перспективным здесь является выделение наиболее трудно поддающейся математическому описанию части объекта и включение этой реальной части физического объекта в имитационную модель. Тогда модель реализуется, с одной стороны, на базе средств вычислительной техники, а с другой – имеется реальная часть объекта. Это значительно расширяет возможности и повышает достоверность результатов моделирования.

Имитационная система реализуется на ЭВМ и позволяет исследовать имитационную модель, задаваемую в виде определенной совокупности отдельных блочных моделей и связей между ними в их взаимодействии в пространстве и времени при реализации какого-либо процесса.

Можно выделить три основные группы блоков:

- *блоки, характеризующие моделируемый процесс функционирования системы;*
- *блоки, отображающие внешнюю среду и ее воздействие на реализуемый процесс;*

- *блоки, играющие служебную вспомогательную роль, обеспечивая взаимодействие первых двух, а также выполняющие дополнительные функции по получению и обработке результатов моделирования.*

Вопросы для самоконтроля

1. Понятие моделирования
2. Классификационные признаки моделирования
3. Эффективность моделирования систем
4. Виды моделирования

2.2. Основы математического моделирования. Набор специальных программ для статистической обработки данных (MATHCAD).

Основные модули

- Математическое моделирование
- Классификация математических моделей
- Этапы, цели и средства компьютерного математического моделирования
- Набор специальных программ для статистической обработки данных (MATHCAD).

Математическое моделирование - метод исследования процессов и явлений на их математических моделях.

Изучение компьютерного математического моделирования открывает широкие возможности для осознания связи информатики с математикой и другими науками – естественными и социальными. Компьютерное математическое моделирование в разных своих проявлениях использует практически весь аппарат современной математики.

Математическое моделирование не всегда требует компьютерной поддержки. Каждый специалист, профессионально занимающийся

математическим моделированием, делает все возможное для аналитического исследования модели. Аналитические решения (т.е. представленные формулами, выражающими результаты исследования через исходные данные) обычно удобнее и информативнее численных. Возможности аналитических методов решения сложных математических задач, однако, очень ограничены и, как правило, эти методы гораздо сложнее численных. В компьютерном моделировании доминируют численные методы, реализуемые на компьютерах. Однако понятия "аналитическое решение" и "компьютерное решение" отнюдь не противостоят друг другу, так как:

а) все чаще компьютеры при математическом моделировании используются не только для численных расчетов, но и для аналитических преобразований:

б) результат аналитического исследования математической модели часто выражен столь сложной формулой, что при взгляде на нее не складывается восприятия описываемого ей процесса. Эту формулу нужно представить графически, проиллюстрировать в динамике, иногда даже озвучить, т.е. проделать то, что называется "визуализацией абстракций". При этом компьютер - незаменимое техническое средство.

Классификация математических моделей

К классификации математических моделей можно подходить по-разному, положив в основу классификации различные принципы.

Классификация моделей по отраслям наук (математические модели в физике, биологии, социологии и т.д.);

Классификация моделей по применяемому математическому аппарату (модели, основанные на применении обыкновенных дифференциальных уравнений, дифференциальных уравнений в частных производных, стохастических методов, дискретных алгебраических преобразований и т.д.);

Классификация моделей с точки зрения целей моделирования:

- дескриптивные (описательные) модели;
- оптимизационные модели;

- многокритериальные модели;
- игровые модели;
- имитационные модели.

Пример.

1. Моделируя движение кометы, вторгшейся в Солнечную систему, мы описываем (предсказываем) траекторию ее полета, расстояние, на котором она пройдет от Земли и т.д., т.е. ставим чисто описательные цели. У нас нет никаких возможностей повлиять на движение кометы, что-то изменить.
2. Меняя тепловой режим в зернохранилище, мы можем стремиться подобрать такой, чтобы достичь максимальной сохранности зерна, т.е. оптимизируем процесс.
3. Часто приходится оптимизировать процесс по нескольким параметрам сразу, причем цели могут быть весьма противоречивыми. Например, зная цены на продукты и потребность человека в пище, организовать питание больших групп людей (в армии, летнем лагере и др.) как можно полезнее и как можно дешевле.
4. Игровые модели могут иметь отношение не только к детским играм (в том числе и компьютерным), но и к вещам весьма серьезным.
5. Бывает, что модель в большой мере подражает реальному процессу, т.е. имитирует его.

Этапы, цели и средства компьютерного математического моделирования

Здесь мы рассмотрим процесс компьютерного математического моделирования, включающий численный эксперимент с моделью (рис. 2.2.1).

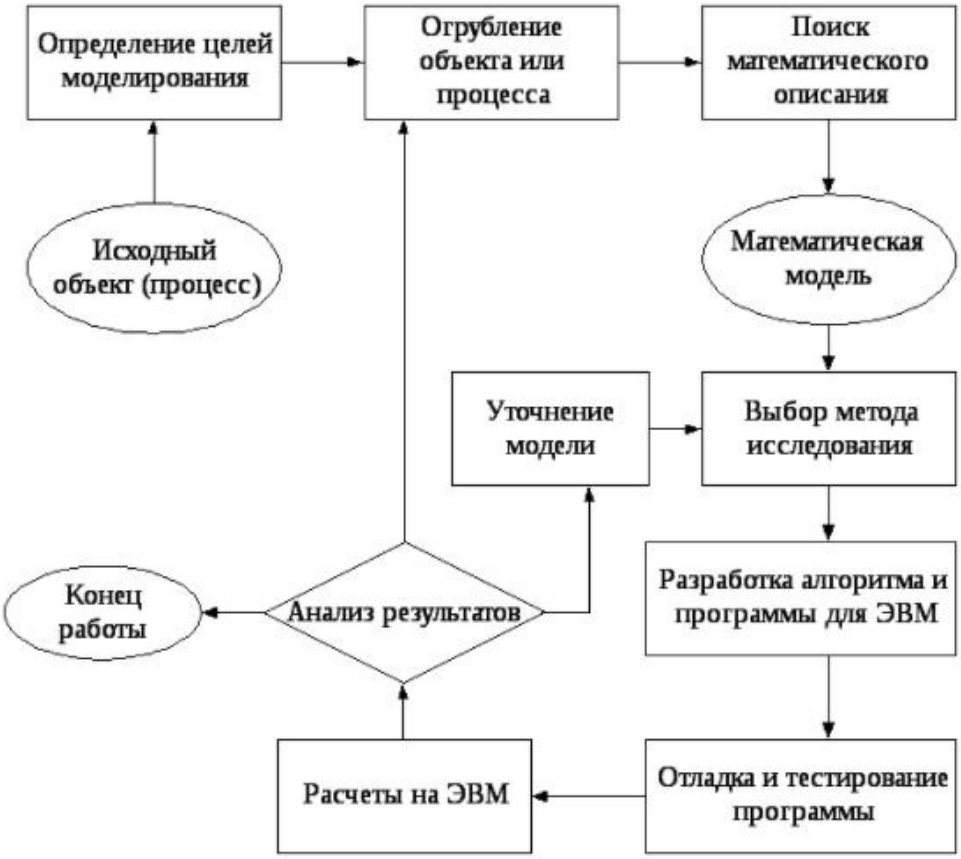


Рис. 2.2.1. Общая схема процесса компьютерного математического моделирования

Первый этап - определение целей моделирования.

Основные из них таковы:

1. модель нужна для того, чтобы понять, как устроен конкретный объект, какова его структура, основные свойства, законы развития и взаимодействия с окружающим миром (понимание);
2. модель нужна для того, чтобы научиться управлять объектом (или процессом) и определить наилучшие способы управления при заданных целях и критериях (управление);
3. модель нужна для того, чтобы прогнозировать прямые и косвенные последствия реализации заданных способов и форм воздействия на объект (прогнозирование).

Выработка концепции управления объектом - другая возможная цель моделирования. Какой режим полета самолета выбрать для того, чтобы полет

был вполне безопасным и экономически наиболее выгодным? Как составить график выполнения сотен видов работ на строительстве большого объекта, чтобы оно закончилось в максимально короткий срок? Множество таких проблем систематически возникает перед экономистами, конструкторами, учеными.

Наконец, *прогнозирование последствий* тех или иных воздействий на объект может быть как относительно простым делом в несложных физических системах, так и чрезвычайно сложным – на грани выполнимости – в системах биолого-экономических, социальных. Если относительно легко ответить на вопрос об изменении режима распространения тепла в тонком стержне при изменениях в составляющем его сплаве, то несравненно труднее проследить (предсказать) экологические и климатические последствия строительства крупной ГЭС или социальные последствия изменений налогового законодательства. Возможно, и здесь методы математического моделирования будут оказывать в будущем более значительную помощь.

Второй этап – разделение входных и выходных параметров.

Составим список величин, от которых зависит поведение объекта или ход процесса, а также тех величин, которые желательно получить в результате моделирования. Обозначим первые (входные) величины через x_1, x_2, ..., x_n; вторые (выходные) через $y_1, y_2, ..., y_k$.

Символически поведение объекта или процесса можно представить в виде: $y_j = Fj(x_1, x_2, ..., x_n)$ $(j = 1,2, ... , k)$,

где F – те действия, которые следует произвести над входными параметрами, чтобы получить результаты. Входные параметры, могут быть известны "точно", т.е. поддаваться (в принципе) измерению однозначно и с любой степенью точности – тогда они являются детерминированными величинами. Так, в классической механике, сколь сложной ни была бы моделируемая система, входные параметры детерминированы – соответственно, детерминирован, однозначно развивается во времени процесс эволюции такой системы.

Однако в природе и обществе гораздо чаще встречаются процессы иного рода, когда значения входных параметров известны лишь с определенной степенью вероятности, т.е. эти параметры, являются вероятностными (стохастическими), и, соответственно, таким же является процесс эволюции системы (случайный процесс).

"Случайный" – не значит "непредсказуемый"; просто характер исследования, задаваемых вопросов резко меняется (они приобретают вид "С какой вероятностью...", "С каким математическим ожиданием..." и т.п.). Примеров случайных процессов не счесть как в науке, так и в обыденной жизни (силы, действующие на летящий самолет в ветреную погоду, переход улицы при большом потоке транспорта и т.д.).

Для стохастической модели выходные параметры могут быть как величинами вероятностными, так и однозначно определяемыми.

Важнейшим этапом моделирования является разделение входных параметров по степени важности влияния их изменений на выходные. Такой процесс называется ***ранжированием*** (разделением по рангам). Чаще всего невозможно (да и не нужно) учитывать все факторы, которые могут повлиять на значения интересующих нас величин y.

От того, насколько умело выделены важнейшие факторы, зависит успех моделирования, быстрота и эффективность достижения цели. Выделить более важные (или, как говорят, значимые) факторы и отсеять менее важные может лишь специалист в той предметной области, к которой относится модель.

Отбрасывание (по крайней мере при первом подходе) менее значимых факторов огрубляет объект моделирования и способствует пониманию его главных свойств и закономерностей. Умело ранжированная модель должна быть адекватна исходному объекту или процессу в отношении целей моделирования. Обычно определить, адекватна ли модель, можно только в процессе экспериментов с ней, анализа результатов.

Следующий этап – поиск математического описания. На этом этапе необходимо перейти от абстрактной формулировки модели к формулировке, имеющей конкретное математическое наполнение. В этот момент модель предстает перед нами в виде уравнения, системы уравнений, системы неравенств, дифференциального уравнения или системы таких уравнений и т.д.

Когда математическая модель сформулирована, выбирается *метод ее исследования.* Как правило, для решения одной и той же задачи есть несколько конкретных методов, различающихся эффективностью, устойчивостью и т.д. От верного выбора метода часто зависит успех всего процесса.

Разработка алгоритма и составление программы для ЭВМ – это творческий процесс. В настоящее время при компьютерном математическом моделировании часто используются приемы процедурно-ориентированного (структурного) программирования.

При создании имитационной модели можно также воспользоваться возможностями одного из пакетов математической поддержки (MATHEMATICA, MathCad, MathLab и др).

В настоящее время существуют проблемно-ориентированные имитационные языки, в которых объединяются различные альтернативные подходы, и которые самой своей структурой определяют возможную схему действий разработчика модели. Характерным примером такого рода является имитационный язык СЛАМ II (SLAM – Simulating Language for Alternative Modeling имитационный язык для альтернативного моделирования).

После составления программы решаем с ее помощью простейшую *тестовую задачу* (желательно, с заранее известным ответом) с целью устранения грубых ошибок. Это – лишь начало процедуры тестирования, которую трудно описать формально исчерпывающим образом. По существу, тестирование может продолжаться долго и закончиться тогда, когда

пользователь по своим профессиональным признакам сочтет программу верной.

Затем следует собственно численный эксперимент, и выясняется, соответствует ли модель реальному объекту (процессу). Модель адекватна реальному процессу, если некоторые характеристики процесса, полученные на ЭВМ, совпадают с экспериментальными с заданной степенью точности. В случае несоответствия модели реальному процессу возвращаемся к одному из предыдущих этапов.

Набор специальных программ для статистической обработки данных (MATHCAD).

MathCAD – это мощная и простая универсальная среда для решения задач в различных отраслях науки и техники, финансов и экономики, физики и астрономии, строительства и архитектуры, математики и статистики, организации производства и управления. Она располагает широким набором инструментальных, информационных и графических средств. MathCAD – одна из самых популярных математических систем, которая пользуется спросом у экономистов, менеджеров, инженеров, научных работников и всех тех, чья деятельность связана с количественными методами расчета.

Интерфейс этого пакета схож с другими приложениями Windows, что позволяет обучаемому оказаться в привычной среде при загрузке пакета и использовать знания и умения, полученные при работе с другими Windows-приложениями. Большое удобство создает панель **Математика**, с помощью которой можно активизировать другие панели, необходимые при решении задач: **Графики, Матрицы (Матрицы), Калькулятор (Арифметика), Матанализ (Исчисление)** и др. (рис. 2.2.2) – разные названия специализированных панелей вызваны различными версиями пакета.

Начинать работу с пакетом, лучше всего с простейших его функций – *вычисление математических выражений* (при этом не надо давать сложные выражения, чтобы не оттолкнуть пользователя). Для таких целей есть панель Калькулятор. Но в отличие от обычного калькулятора на панели

Калькулятор, кроме основных математических операторов, есть и другие, позволяющие использовать этот пакет для записи и вычисления сложных математических формул.

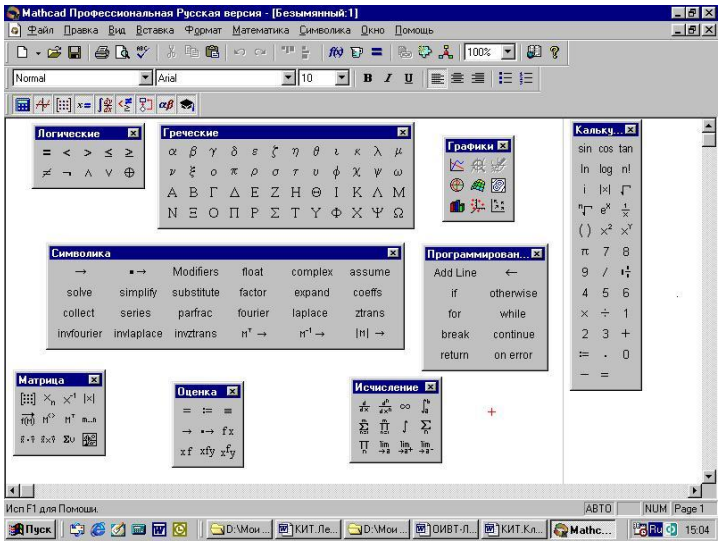
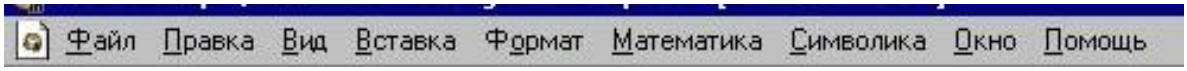


Рис. 2.2.2. Интерфейс пакета MathCAD

Вторая строка окна системы – главное меню. Назначение его команд приведено ниже:



File (Файл) – работа с файлами, сетью интернет и электронной почтой;

Ниспадающее меню содержит команды, стандартные для Windows-приложений.

Edit (Правка) – редактирование документов; Ниспадающее меню также содержит команды, стандартные для Windows-приложений. Большинство из них доступны только в случае, если в документе выделены одна или несколько областей (текст, формула, график и т.д.)

View (Обзор) – изменение средств обзора;

Toolbars (Панели) – позволяет отображать или скрывать панели инструментов **Standart** (Стандартная), **Formatting** (Форматирования), **Math** (Математика).

Status bar (Строка состояния) – включение или отключение отображения строки состояния системы.

Ruler (линейка) – включение-отключение линейки.

Regions (Границы) – lелает видимыми границы областей (текстовых, графических, формул).

Zoom (изменение масштаба).

Refresh (Обновить) [Ctrl+R] – обновление содержимого экрана.

Animate (Анимация) – Команда позволяет создать анимацию.

Playback (Проигрыватель) – Воспроизведение анимации, хранящейся в файле с расширением AVI.

Preferences (Настройки) – Одна из вкладок всплывающего окна **(General)** позволяет задать некоторые параметры работы программы, не влияющие на вычисления, другая вкладка **(Internet)** служит для ввода информации при совместной работе с MathCAD-документами через Internet.

Insert (Вставка) – Команды этого меню позволяют помещать в MathCAD-документ графики, функции, гиперссылки, компоненты и встраивать объекты.

Format (Формат) – изменение формата объектов.

Equation (Уравнение) – Форматирование формул и создание собственных стилей для представления данных.

Result (Результат) – Позволяет задать формат представления результатов вычислений.

Text (Текст) – Форматирование текстового фрагмента (шрифт, размер, начертание).

Paragraf (Абзац) – Изменение формата текущего абзаца (отступы, выравнивание).

Tabs (Табуляция) – Задание позиций маркеров табуляции.

Style (Стиль) – Оформление текстовых абзацев.

Properties (Свойства) – Вкладка **Display** (Отображение) позволяет задать цвет фона для наиболее важных текстовых и графических областей;

вставленный в документ рисунок (Insert -> Picture) позволяет заключить в рамку, вернуть ему первоначальный размер.

Вкладка **Calculation** (Вычисление) – позволяет для выделенной формулы включить и отключить вычисление; в последнем случае в правом верхнем углу области формулы появляется маленький черный прямоугольник, и формула превращается в комментарий.

Graf (График) – Позволяет менять параметры отображения графиков.

Separate regions (Разделить области) – Позволяет раздвигать перекрывающиеся области.

Align regions (Выровнять области) – Выравнивает выделенные области по горизонтали или по вертикали.

Headers/Footers (Колонтитулы) – создание и редактирование колонтитулов.

Repaganite Now (Перенумерация страниц) – Производит разбивку текущего документа на страницы.

Math (Математика) – управление процессом вычислений; в MathCAD существует два режима вычислений: автоматический и ручной. В автоматическом режиме результаты вычислений полностью обновляются при каком-либо изменении в формуле.

Automatic Calculation (Автоматическое вычисление) – позволяет переключать режимы вычислений.

Calculate (Вычислить) – При ручном режиме вычислений позволяет пересчитать видимую часть экрана.

Calculate Worksheet (Просчитать документ) – Пересчет всего документа целиком.

Optimization (Оптимизация) – При помощи этой команды можно заставит MathCAD перед численной оценкой выражения произвести символьные вычисления и при нахождении более компактной формы выражения использовать именно ее. Если выражение удалось оптимизировать, то справа от него появляется маленькая красная звездочка.

Двойной щелчок на ней открывает окно, в котором находится оптимизированный результат.

Options (Параметры) – позволяет задавать параметры вычислений.

Symbolik (Символика) – выбор операций символьного процессора.

Window (Окно) – управление окнами системы;

Help (?) – работа со справочной базой данных о системе;

Mathcad Help (Справка по MathCAD) – содержит три вкладки: **Содержание** – справка упорядочена по темам; **Указатель** – предметный указатель; **Поиск** – находит нужное понятие при вводе его в форму.

Resource Center (Центр ресурсов) – Информационный центр, содержащий обзор вычислительных способностей MathCAD **(Overview and Tutorials)**, быструю справку в виде примеров из различных областей математики **(Quicksheets and Reference tables)**.

Tip of the Day – Всплывающие окна-подсказки с полезными советами (возникают при загрузке системы).

Open Book (Открыть книгу) – позволяет открыть справочник системы MathCAD.

Наборные математические панели инструментов.

Для ввода математических знаков в MathCAD используются удобные перемещаемые наборные панели со знаками. Они служат для вывода заготовок – шаблонов математических знаков (цифр, знаков арифметических операций, матриц, знаков интегралов, производных и т.д.). Для вывода панели **Math** необходимо выполнить команду **View -> Toolbar -> Math**. Наборные панели появляются в окне редактирования документов при активизации соответствующих пиктограмм – первая линия пиктограмм управления системой. Используя общую наборную панель, можно вывести или все панели сразу или только те, что нужны для работы. Для установки с их помощью необходимого шаблона достаточно поместить курсор в желаемое место окна редактирования (красный крестик на цветном дисплее)

и затем активизировать пиктограмму нужного шаблона, установив на него курсор мыши и нажав ее левую клавишу (рис.2.2.3.).

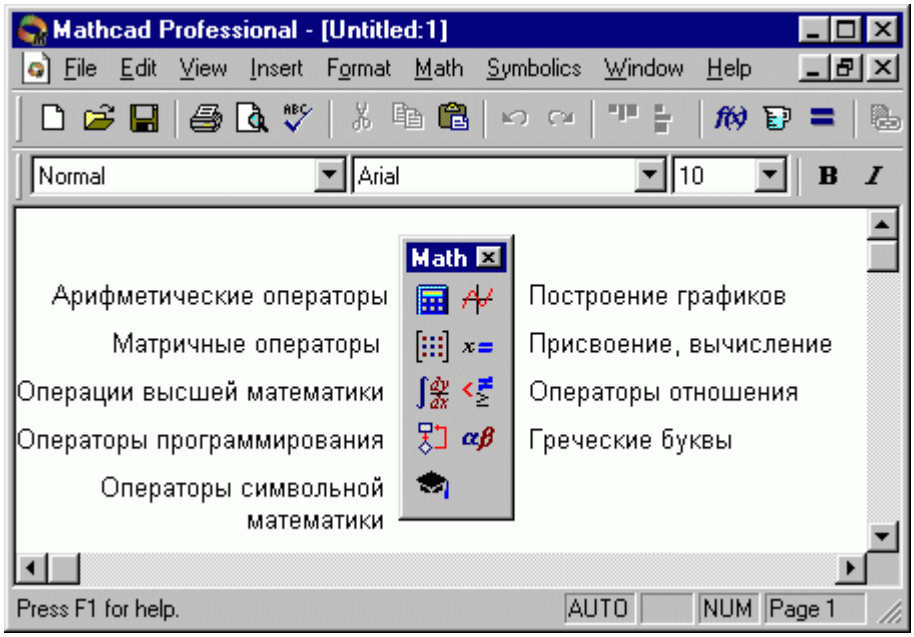


Рис.2.2.3. Математическая панель инструментов

Многие функции и операции, которые вставляются в документ с помощью наборных математических панелей, могут быть помещены в документ с помощью "быстрых клавиш". При этом работа в системе MathCAD становится более продуктивной. Рекомендуем запомнить сочетания клавиш хотя бы для некоторых наиболее часто употребляемых команд.

Math CAD обладает специализированным входным языком программирования очень высокого уровня, ориентированным на математические расчеты. Поэтому, рассматривая входной язык системы как язык программирования, мы можем выделить в нем типичные понятия и объекты. К ним относятся идентификаторы, константы, переменные, массивы и другие типы данных, операторы и функции, управляющие структуры и т.д. Четкое представление об их возможностях и правилах применения (синтаксисе) весьма полезно при решении задач умеренной и высокой сложности.

Следующим шагом в изучении этого пакета может стать *вычисление выражений, содержащих переменные.* Здесь снова очевидное преимущество пакета в его простоте и наглядности, когда выражение набирается в обычной математической нотации (сравните с пакетом «Математика», где используется специальная символика). А результаты можно получать разные в зависимости от значения переменных.

При этом в пакете имеется много встроенных функций, которые можно вставлять в выражения. И кроме обычных функций: логарифмических, тригонометрических, – есть и такие, как: нахождение минимума, максимума, среднего и др.

Есть и функция **if**, которую удобно использовать для вычисления значения разрывной функции на разных отрезках и в точках разрыва (рис. 2).

Например, для вычисления *Y* по формуле

$$Y = \begin{cases} x, & \text{при } x >= 0 \\ x + 10, & \text{при } x < 0 \end{cases}$$

функция if запишется так: if (x >= 0, x, x + 10).

Некоторые из встроенных функций, наиболее часто используемые, имеются на панели Калькулятор. Но, кроме этого, в пакете есть и другие встроенные функции, которые хранятся в библиотеке функций. Их можно вызвать для работы, используя команду Вставка - Функция. Кроме имеющихся встроенных функций, пользователь может создавать свои - *функции пользователя* (рис. 2.2.4.).

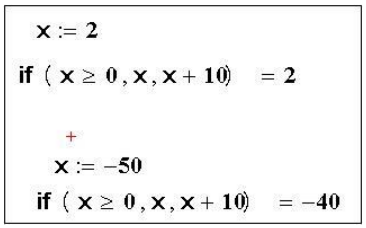


Рис. 2.2.4. Функция if

Общеизвестно, как бывает утомительно *упрощать математические выражения*. В MathCAD имеется соответствующий аппарат, облегчающий

эту работу. Для этого используется команда Символы - Упростить. Этой возможностью можно пользоваться, например, для проверки выполненного вручную преобразования.

Есть в пакете и возможность *развернуть выражение*: команда Символы – Расширить. В пакете есть *средство для разложения на множители*, с помощью которого числа раскладываются на произведение простейших множителей: команда Символы – Фактор. Можно и средствами пакета *приводить подобные члены*: команда Символы – Подобные.

Система MathCAD позволяет *решать уравнения*: команда Символы – Переменная – Вычислить.

Следующий раздел, к которому, на наш взгляд, следует перейти, – это *построение графиков*. Графики позволяют наглядно представить различные ситуации. На графике можно проанализировать, как, например, ведет себя прямая при изменении углового коэффициента и свободного члена. При этом на одной координате плоскости можно сразу представить несколько графиков (рис. 2.2.5), и этот графический образ останется в памяти и позволит не только легко освоить соответствующий материал, но и запомнить графическую интерпретацию.

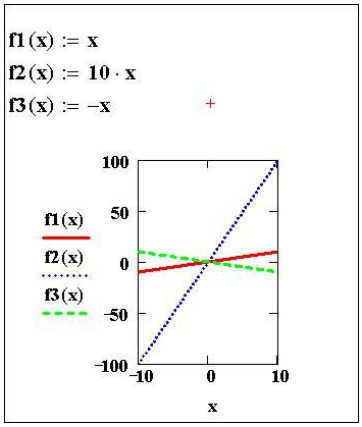


Рис. 2.2.5. Построение графиков

Аналогичный графический анализ можно провести и для тригонометрических (рис.2.2.6) и логарифмических функций, а также для функций второго порядка.

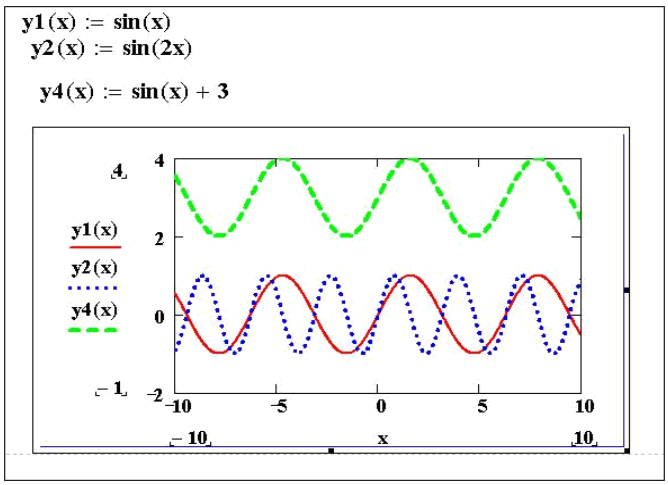


Рис. 2.2.6. Тригонометрические функции

Такие графические образы не только позволяют понять изучаемый материал, но и привлекают к пакету и увлекают им. Кроме того, графическая символика позволяет применить этот аппарат и в прикладных задачах: экономических, инженерных и др.

Графики можно строить как двухмерные, так и трехмерные. Построенные графики можно форматировать, придавая им цвет, тип линии (сплошная, пунктир и т.д.) и другие возможности.

Работа с графикой обычно вызывает большой интерес у обучаемых, и они затем самостоятельно пытаются строить самые различные графики, усваивая самостоятельно новый материал. Более того, обучаемые начнут пытаться использовать графические возможности и при изучении других дисциплин.

После рассмотрения темы «Графики» можно перейти к *линейной алгебре:* работе с векторами и матрицами. Здесь обучаемый видит, как можно легко найти определитель матрицы, транспонированную и обратную матрицы, перемножить векторы и матрицы.

А из темы матрицы – прямая дорога к *решению системы линейных уравнений* путем обратной матрицы, правила Крамера.

Ну и конечно средствами пакета MathCAD можно *находить пределы, производные, интегралы, разлагать функцию в ряд Тейлора* и др.

При изучении пакета MathCAD стоит обратить внимание и на то, что средства пакета позволяет *решить нелинейные уравнения и системы*. Для этого в пакете имеется соответствующий набор встроенных функций, упрощающих этот процесс, например функция **find.**

Примеры решения задач на MathCAD.

***1-пример*. Показательный пример линейного алгоритма.**

Вычислите коэффициент изгиба стержня, работающий при сжатии: $\varphi = \frac{\delta_{kr}}{K R_y}$

Здесь K – коэффициент безопасности ($K=1,4$).

δ_{kr} – значение критического напряжения, R_y – сопротивление.

Для решения задачи уточним входные и выходные параметры.

Входные параметры: δ_{kr} *(sigma), K,* R_y

Выходные параметры: φ *(Fi)* (рис.2.2.7).

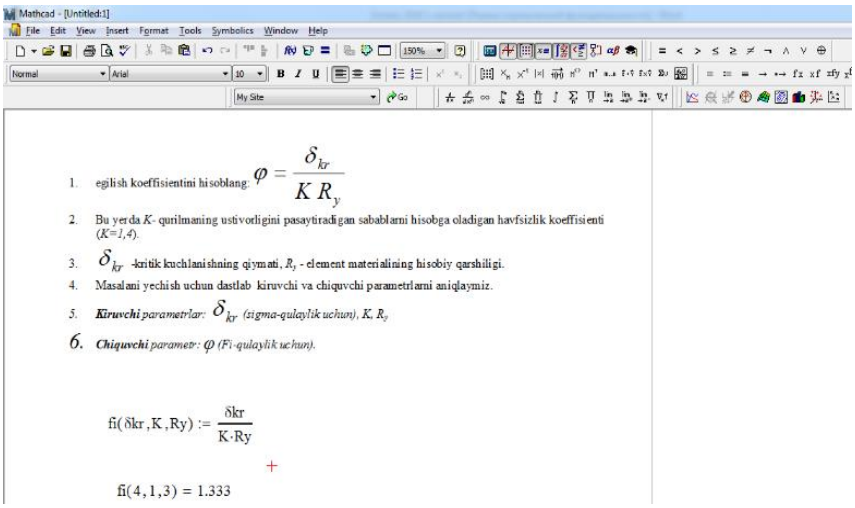


Рис. 2.2.7. Вычисления линейного алгоритма

2-пример: Высота построенного здания в виде цилиндра 45 м, диаметр основания 25 м, стороны покрыты окнами, вычислите сколько окон в м^2 расходуется для здания (рис. 2.2.8.).

Входные параметры: h=45, d=25, Выходные параметры $S_{yon} = ?$

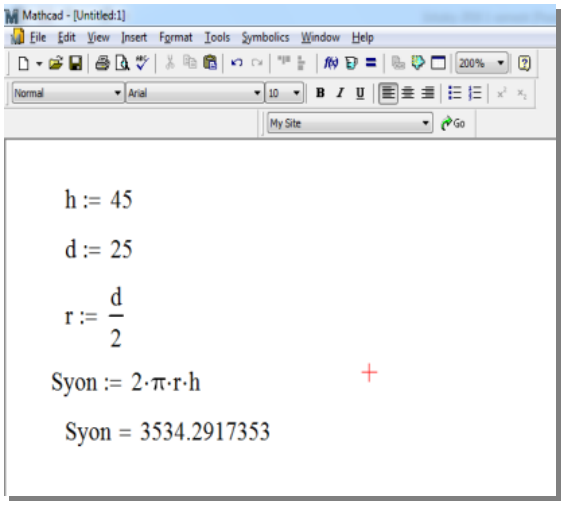


Рис. 2.2.8. Вычисления параметров здания

3-пример: Решение уравнения методом Крамера (рис. 2.2.9)

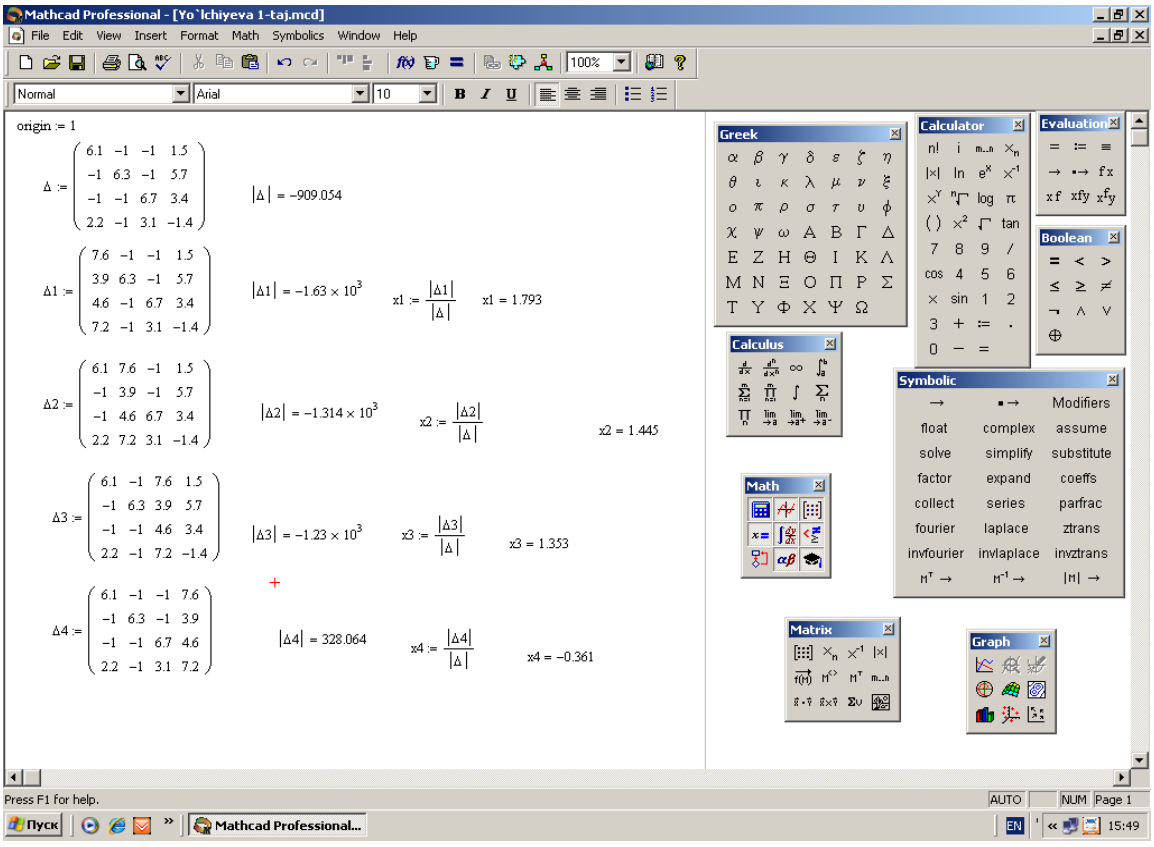


Рис. 2.2.9. Решение уравнения методом Крамера

Вопросы для самоконтроля

1. Математическое моделирование
2. Классификация математических моделей

3. Этапы, цели и средства компьютерного математического моделирования
4. Набор специальных программ для статистической обработки данных (MATLAB, MATHCAD).

2.3. Основы графического моделирования. Использование графических возможностей AutoCAD в процессе проектирования.

Основные модули

- Понятие компьютерной графики
- Виды компьютерной графики
- Графическое моделирование объектов на программе AutoCAD
- Описание окна AutoCAD

Компьютерная графика

Компьютерная графика – это наука, предметом изучения которой является создание, хранение и обработка моделей и их изображений с помощью ЭВМ, т.е. это раздел информатики, который занимается проблемами получения различных изображений (рисунков, чертежей, мультипликации) на компьютере.

Под компьютерной графикой обычно понимают автоматизацию процессов подготовки, преобразования, хранения и воспроизведения графической информации с помощью компьютера. Под графической информацией понимаются модели объектов и их изображения.

Компьютерная графика – это область информатики, занимающаяся проблемами получения различных изображений (рисунков, чертежей, мультипликации) на компьютере. Работа с компьютерной графикой - одно из самых популярных направлений использования персонального компьютера, причем занимаются этой работой не только профессиональные художники и

дизайнеры. На любом предприятии время от времени возникает необходимость в подаче рекламных объявлений в газеты и журналы, в выпуске рекламной листовки или буклета. Иногда предприятия заказывают такую работу специальным дизайнерским бюро или рекламным агентствам, но часто обходятся собственными силами и доступными программными средствами.

Без компьютерной графики не обходится ни одна современная программа. Работа над графикой занимает до 90% рабочего времени программистских коллективов, выпускающих программы массового применения.

Основные трудозатраты в работе редакций и издательств тоже составляют художественные и оформительские работы с графическими программами.

Необходимость широкого использования графических программных средств стала особенно ощутимой в связи с развитием Интернета и, в первую очередь, благодаря службе World Wide Web, связавшей в единую "паутину" миллионы "домашних страниц". У страницы, оформленной без компьютерной графики мало шансов привлечь к себе массовое внимание.

Область применения компьютерной графики не ограничивается одними художественными эффектами. Во всех отраслях науки, техники, медицины, в коммерческой и управленческой деятельности используются построенные с помощью компьютера схемы, графики, диаграммы, предназначенные для наглядного отображения разнообразной информации. Конструкторы, разрабатывая новые модели автомобилей и самолетов, используют трехмерные графические объекты, чтобы представить окончательный вид изделия. Архитекторы создают на экране монитора объемное изображение здания, и это позволяет им увидеть, как оно впишется в ландшафт.

Виды компьютерной графики

Существует три вида компьютерной графики:

- Растровая графика
- Векторная графика
- Фрактальная графика

Растровое изображение, цифровое изображение – это файл данных или структура, представляющая прямоугольную сетку пикселей или точек цветов на компьютерном мониторе, бумаге и других отображающих устройствах и материалах. То есть, *растровая графика* – это формат изображения, который содержит информацию о расположении, количестве и цвете пикселей.

Главным достоинством *растровой графики* является создание (воспроизведение) практически любого рисунка, вне зависимости от сложности, в отличие, например, от векторной, где невозможно точно передать эффект перехода от одного цвета к другому (в теории, конечно, возможно, но файл размером 1 МБ в формате BMP будет иметь размер 200 МБ в векторном формате).

Векторная графика (другое название – **геометрическое моделирование**) – это использование геометрических примитивов, таких как точки, линии, сплайны и многоугольники, для представления изображений в компьютерной графике. Термин используется в противоположность к растровой графике, которая представляет изображения как матрицу пикселей (точек).

Изначально человеческий глаз воспринимает изображение подобно растровому образу. Картинка проецируется на сетчатку, состоящую из отдельных, реагирующих на свет клеток. Далее система глаз-мозг распознаёт в изображении отдельные объекты, геометрические фигуры, которые уже легче обрабатывать и запоминать.

Фрактальная графика основана на математических вычислениях. Базовым элементом фрактальной графики является сама математическая формула, то есть никаких объектов в памяти компьютера не хранится и изображение строится исключительно по уравнениям. Таким способом

строят как простейшие регулярные структуры, так и сложные иллюстрации, имитирующие природные ландшафты и трехмерные объекты.

Графическое моделирование геометрических объектов на AutoCAD

Графическое моделирование включает следующие составляющие:

1. Примитив

Примитивы - это стандартные геометрические элементы, при помощи которых строятся изображения в AutoCAD. Дело в том, что Автокад работает не с изображением как таковым, а с геометрическим описанием объектов, составляющих изображение, что обусловлено задачами САПР. У такого представления есть как преимущества, так и недостатки. С одной стороны, графическое представление изделия с его геометрическим описанием более компактно и позволяет производить различные геометрические преобразования, а также напрямую использовать такое описание в автоматизированных системах технологической подготовки производства. С другой стороны, мы не можем рисовать кривые произвольной формы. Однако геометрического представления достаточно для создания любого технического изображения. Такое представление данных называется векторным представлением в отличие от пиксельного описания картинки как поля цветных или черно-белых точек экрана.

Все примитивы AutoCAD обладают рядом свойств (принадлежность слою, цвет, тип линии, ширина).

2. Система координат

В AutoCAD используется стандартная система декартовых координат. Кроме того пользователю предоставляется возможность определять свои собственные системы координат. Такие введенные пользователем системы координат называют пользовательскими системами координат – ПСК (UCS).

Таким образом, работая с рисунком можно пользоваться разными системами координат. Однако в каждый момент времени пользователь работает только с одной предварительно выбранной системой координат, которая называется текущей. Вся работа с изображением осуществляется в

текущей системе координат. В левом нижнем углу графической зоны AutoCAD постоянно показывает пиктограмму текущей системы координат.

3. Единицы измерения

Расстояния между точками на рисунке измеряются в условных единицах. Конкретный формат представления размеров (дюймы, сантиметры, миллиметры) не имеет значения для AutoCAD. Иначе говоря, при создании объектов в чертеже AutoCAD измеряет все расстояния в относительных единицах. Соответствие между условными единицами AutoCAD и конкретной системой (метрической, дюймовой) устанавливается выбором формата представления.

4. Вид

Когда создается чертеж на AutoCAD, работа ведется с изображением части (или всего) чертежа выводимого на дисплей. Эту часть изображения называют видом. Эту видимую часть чертежа (окно зрения) можно увеличивать (при этом изображение чертежа будет уменьшаться), уменьшать (изображение будет увеличиваться) или перемещать по полю чертежа без изменения масштаба отображения (панорамирование).

5. Слой

AutoCAD даёт возможность распределять выбранные фрагменты чертежа по различным слоям. Количество слоёв не ограничено. Слои можно делать видимыми и невидимыми. С каждым слоем чертежа связывается цвет и тип линий. По мере создания чертежа можно вводить новые слои и менять свойства уже существующих. При создании слоя ему присваивается имя, отличающее его от других слоёв. Подобно системе координат в любом чертеже AutoCAD всегда существует по крайней мере один слой с именем “0”.

6. Чертёж

Чертёж – это файл с информацией, описывающей графический объект. AutoCAD позволяет создавать и редактировать чертеж множеством различных способов. Любое изображение создается при помощи базового

набора примитивов. AutoCAD предоставляет в распоряжение пользователя все вычислительные ресурсы ЭВМ, новые средства создания и редактирования чертежа. Помимо высокой точности построений (задаваемой пользователем), AutoCAD позволяет упростить геометрические построения, определяя геометрические характерные точки объектов, предоставляет информацию о чертеже (координаты, расстояния, площади).

7. Блоки

Примитивы могут быть простыми, сложными и составными. Составные примитивы в AutoCAD называют блоками. Перечислим преимущества использования блоков:

1. Блоки хранятся отдельно от остального чертежа, и один и тот же блок может использоваться в чертеже многократно, что позволяет сократить время создания чертежа, упростить редактирование и т.д.

2. Блок можно вставлять в разрабатываемый чертёж под любым углом и в любом масштабе. Это позволяет создавать изображения из любых символов.

3. Есть возможность создания библиотеки постоянно используемых блоков, что значительно ускоряет процесс проектирования.

4. Блоки отличаются от других примитивов тем, что имеют имя. Поэтому, определив новый блок с тем же именем, что и старый мы заменим блок во всём чертеже (при этом старый блок будет удалён).

8.Атрибут

Атрибут – специальный примитив AutoCAD, предназначенный для работы с блоками. Будучи записан в блок, он служит в качестве текстовой переменной, в которую при вставке блока можно записать некоторую строку. Это позволяет, однажды создав блок, в который входит атрибут, с каждой вставкой блока в чертёж связывать новую текстовую строку. Атрибуты особенно удобны при создании различных схем, в которых один и тот же графический символ отрисовывается несколько раз с разными обозначениями

Описание окна AutoCAD

Заголовок окна располагается в самом верху окна. Он содержит имя файла, с которым идет работа, и используется для изменения положения окна на экране. В правом углу заголовка расположены кнопки свертывания, развертывания и закрытия окна. Свернутое окно отображается как кнопка на панели задач.

Большую часть экрана занимает окно документа (рис.2.3.1.).

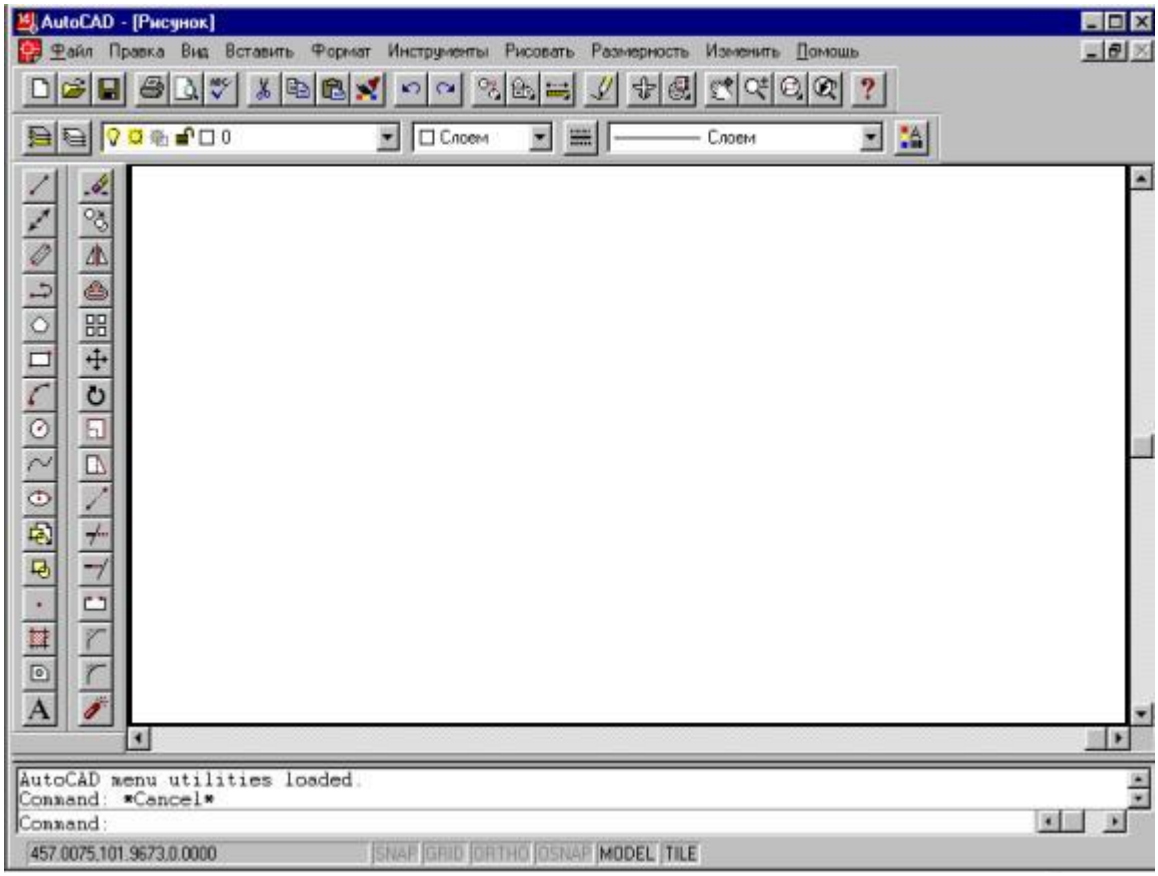


Рис.2.3.1. Окно документа

Главное меню расположено вверху экрана сразу под заголовком окна. Оно содержит названия ниспадающих меню. Если щелкнуть на название меню, на экране появится список команд, предназначенных для доступа к различным функциям AutoCAD.

Строка состояния и командная строка располагаются внизу экрана. Командная строка – это самая главная зона графического экрана. В ней отображается весь диалог с AutoCAD, вне зависимости от того, как

выбирается команда. В строке состояния отображаются сведения о выделенном объекте или команде.

Панель графики обеспечивает доступ инструментам, служащим для создания объектов разных форм, изменения заданных объектов.

При работе в системе AutoCAD также часто используются диалоговые окна (рис.2.3.2.).

Рис. 2.3.2. Главное меню

Как уже говорилось выше главное меню содержит названия ниспадающих меню. Если щелкнуть на название меню, на экране появится список команд, предназначенных для доступа к различным функциям AutoCAD. Многие пункты подменю также являются корнями подменю более низкого уровня.

Пункт главного **меню Файл** открывает ниспадающее меню, опции которого предназначены для непосредственной работы с файлами. Все опции данного меню имеют свои подменю или вызывают диалоговые окна (рис.2.3.3.).

- команда Новый меню Файл служит для создания нового чертежа.
- команда Открыть меню Файл служит для открытия уже созданного чертежа.
- команды Сохранить и Сохранить как меню Файл служат для сохранения созданного чертежа.
- команда Экспорт и Отправить меню Файл служат для передачи данных, находящихся в разных файлах.
- команды Настройки принтера и Печать меню Файл служат для настройки печатающего устройства и передачи чертежей, созданных в системе AutoCAD на печать.
- команда Предварительный просмотр меню Файл служит для просмотра расположения чертежа на бумаге заданного формата.

- команда Выход меню Файл служит для завершения работы в системе AutoCAD.

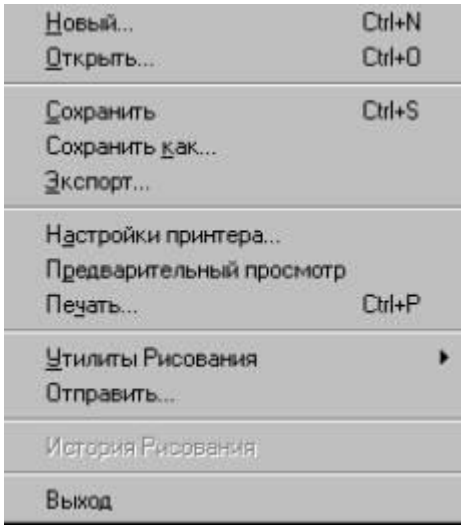


Рис. 2.3.3. Меню Файл

Пункт главного **меню Правка** открывает ниспадающее меню, опции которого предназначены для редактирования создаваемых изображений (рис.2.3.4.).

- команды Копировать, вырезать, вставить, Специальная вставка в меню Правка служат для копирования, удаления и помещения выделенных объектов в буфер обмена Windows.
- команды Отменить и повторить меню Правка служат для отмены или повтора каких-либо команд.

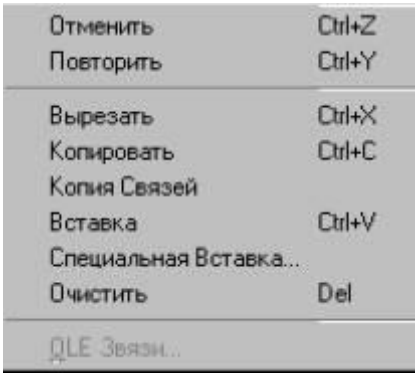


Рис. 2.3.4. Меню Правка

Пункт главного **меню** **Вид** открывает ниспадающее меню, опции которого предназначены для изменения режимов просмотра создаваемых изображений и настройки системы меню и панели инструментов. Команды этого меню помогают изменять размеры изображения, его ориентацию в пространстве. Создавать и просматривать именованные виды чертежа. Многие опции содержат подменю или вызывают диалоговые окна (рис.2.3.5.).

- команда Перерисовать выполняет перерисовку экрана – очищает экран от маркеров и от графического “мусора”, остающегося после выполнения команд редактирования.
- команды Регенерация и регенерировать всё полностью перестраивают изображение после редактирования геометрического описания чертежа. После регенерации всегда автоматически выполняется перерисовка изображения на экране.
- команда Панорама позволяет, не изменяя масштаба отображения чертежа на экране, перемещать его относительно экрана на заданный вектор.
- команды Форточка и Вид с воздуха позволяют управлять размером вида на экране.
- команда Названные Виды открывает диалоговое окно для работы с именованными видами изображения.
- при помощи команд 3D Окно и 3D Динамический Вид AutoCAD предоставляет богатые возможности для работы в трёхмерном пространстве – выбор точки зрения с отработкой проекций, выполнение сечений в трёхмерном пространстве и т.д.
- команда Панель Инструментов предназначена настройки системы меню и панели инструментов.

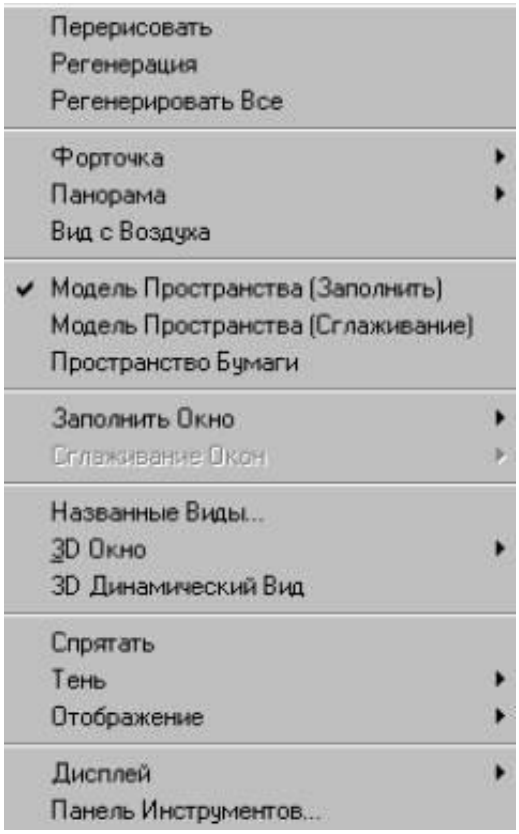


Рис. 2.3.5. Меню Вид

Пункт главного **меню Вставить** открывает ниспадающее меню, опции которого предназначены для вставки в чертеж внешних по отношению к системе AutoCAD объектов (3D Studio, ACIS Solid, метафайлов Windows) и блоков из другого чертежа. Все опции вызывают диалоговые окна для выбора соответствующих объектов (рис.2.3.6.).

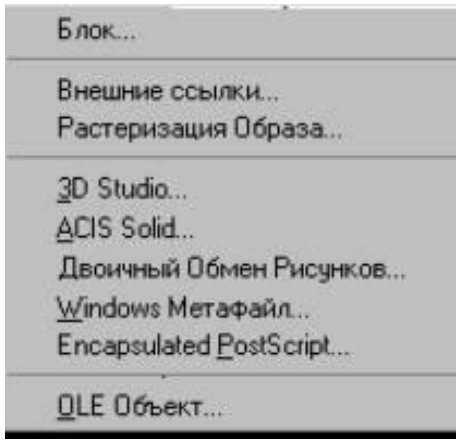


Рис. 2.3.6. Меню Вставить

Пункт главного **меню Формат** открывает ниспадающее меню, опции которого предназначены для задания свойств различным объектам системы AutoCAD (рис.2.3.7.).

- команда Слой меню Формат открывает диалоговое окно для создания и работы с уже созданными слоями чертежа.
- команда Цвет меню Формат открывает диалоговое окно для задания цвета объекта чертежа.
- команда Тип линии меню Формат открывает диалоговое окно для выбора шаблона, по которому обрисовываются линии объекта чертежа.
- команда Стиль текста меню Формат открывает диалоговое окно для задания стиля выводимого на чертеже текста.
- команды Стиль разграничений, Стиль отметок, Стиль мультилиний меню Формат открывают диалоговые окна для задания стиля выводимых на чертеже разграничений, отметок, мультилиний.

Пункт главного **меню Рисовать** открывает ниспадающее меню, опции которого предназначены для создания различных примитивов системы AutoCAD. Команды данного меню дублируются в виде пиктограмм, расположенных на панели графики (таблица 2.3.1.).

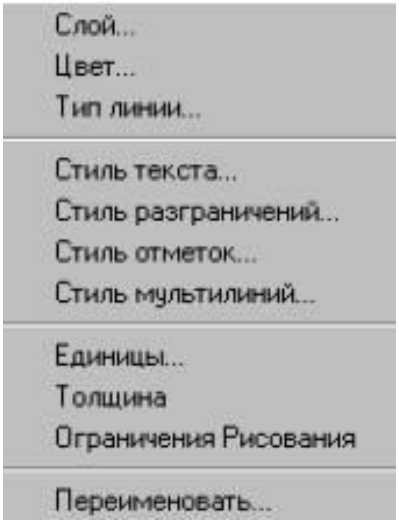


Рис. 2.3.7. Меню Формат

Таблица 2.3.1.

Инструмент	Характеристика изображения
Точка (Point)	Простейший примитив AutoCAD. Точка определяется координатами (X,Y,Z). Точки могут изображаться на экране дисплея различными графическими знаками.
Отрезок (Line)	Часть прямой линии, определяемая двумя крайними точками. Отрезок в AutoCAD имеет нулевую толщину и изображается на экране размером растра.
Фигура (Solid)	Часть плоскости, ограниченная 4(3) отрезками, определяемыми по 4 точкам, 2 из которых могут совпадать. На отрисовку фигуры влияет установка режима закрашивания.
Прямоугольник	Рисование прямоугольников и квадратов, т.е. фигур, ограниченных 4 отрезками, расположенными под углом 90^0.
Полоса (Trace)	Примитив того же типа, что и фигура. Он определен в системе по 4 точкам, однако отрисовывается как отрезок заданной ширины.
Эллипс (Ellips)	Рисование эллипсов и окружностей. Отрисовывается в AutoCAD одной полилинией, сегменты которой являются дугами двенадцатицентрового овала.
Дуга (Arc)	Часть окружности, определяемая центром, радиусом и двумя центральными углами. Дугу в AutoCAD можно построить 10 способами.
Кольцо (Donut)	Объект, создаваемый одной замкнутой полилинией, состоящей из одного имеющего ширину дугового сегмента.
Круг (Circle)	Часть плоскости, ограниченная окружностью. Задается центром и радиусом.

Инструмент	Характеристика изображения
Полилиния (Polyline)	Составная линия, включающая в себя прямолинейные и дуговые сегменты. Её особенность в том, что её сегменты могут иметь не только постоянную, но и переменную ширину.
Многоугольник (Polygon)	Объект, создаваемый одной замкнутой полилинией, состоящей из прямолинейных сегментов равной длины.
3М Грань (3D Face)	Часть плоскости, ограниченная четырьмя отрезками. Основным свойством является непрозрачность граней.
3М Сеть (3D Mesh)	Трехмерный объект, определяемый двухпараметрическим массивом вершин. Трехмерная сеть является развитием трёхмерной грани и может быть расчленена на самостоятельные трёхмерные грани.
Текст (Text)	Примитив, являющийся совокупностью строки текста и специфических для текста свойств: гарнитуры, угла наклона, точки вставки и т.д. Работая с текстом, можно выполнять следующее: выбирать шрифт и задавать его размер и другие атрибуты; задавать интервалы между символами, словами, строками и абзацами.
Граница	Используется для изменения контуров выделенных объектов. Открывает для этого соответствующее диалоговое окно.
Тела	Используется для выбора из соответствующего диалогового окна необходимых геометрических тел.
Блок (Block)	Составной примитив, сформированный из других примитивов и имеющий имя.

Пункт главного **меню Размерность** открывает ниспадающее меню, опции которого предназначены для нанесения размеров на чертежи, задания атрибутов текста размеров, выносных линий, стрелок и т.д. (рис.2.3.8.).

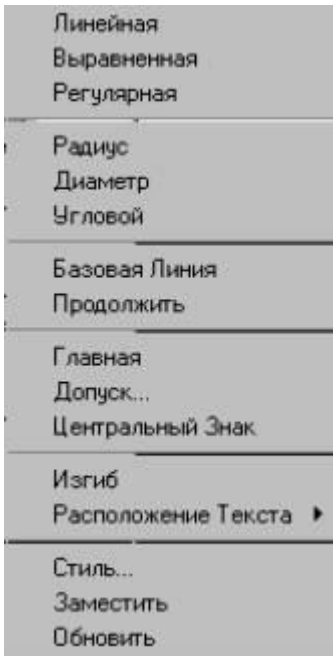


Рис. 2.3.8. Меню Размерность

AutoCAD позволяет наносить различные линейные, угловые размеры, радиусы, диаметры и т.д. в соответствии с нормами международной системой ЕСКД. Допускается привязка к базовой линии. Значения размеров вычисляется самой системой и выводится в заранее заданных единицах измерениях.

Пункт главного **меню Изменить** открывает ниспадающее меню, опции которого предназначены для внесения изменений в создаваемый чертеж, преобразования объектов, изменения стилей, свойств и атрибутов объектов AutoCAD (рис.2.3.9.).

Преобразование положения объекта в программе AutoCAD означает изменение его ориентации или внешнего вида без изменения его исходной формы. Для изменения положения объекта используются команды Двигать и Повернуть.

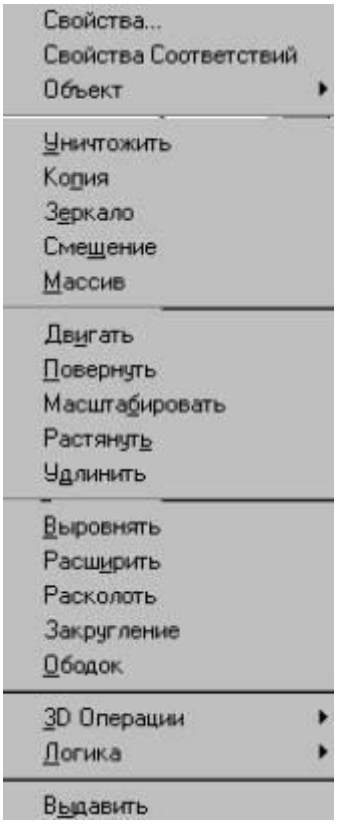


Рис. 2.3.9. Меню Изменить

Преобразование объекта в программе AutoCAD означает изменение его исходной формы. Для этого служат команды Выровнять, расширить, растянуть, удлинить, расколоть, Выдавить.

Программа AutoCAD также позволяет создавать копии объектов и преобразовывать их, оставляя исходные объекты неизменными. Для этого служат команды Копия, Зеркало, Массив.

К объекту или группе объекта можно применять любое количество операций по преобразованию.

- команда Свойства изменяет свойства созданных примитивов – цвет, принадлежность слою, тип линии, высоту и т.д.
- команда Уничтожить удаляет выделенный объект.
- команда Копия копирует выделенные объекты, оставляя оригиналы не тронутыми, и размещает копии в заданном месте или на заданном расстоянии от оригинала.

- команда Зеркало позволяет формировать зеркальные отражения существующих объектов, удаляя или сохраняя при этом оригиналы.
- команда Смещение обеспечивает плоскопараллельный перенос одного или нескольких объектов в указанное место.
- команда Массив позволяет получить несколько копий объектов, группируя их в прямоугольной или круговой матрице.
- команда Повернуть поворачивает заданную группу объектов вокруг заданной точки на заданный угол.
- команда Масштабировать изменяет размеры одного или нескольких объектов. Изображение при этом изменяется относительно заданной базовой точки.
- команда Растянуть обеспечивает перемещение выбранной части изображения, сохраняя при этом связь с остальной частью.
- команда Удлинить удлиняет отрезки, дуги и двумерные полилинии до пересечения с аналогичными примитивами.
- команда Закругление плавно сопрягает отрезки, дуги и окружности дугами разного, в том числе и нулевого, радиуса, а также сопрягает полилинии. При этом “лишние” части примитивов автоматически удаляются.
- команда Ободок проводит линию фаски, удаляя ненужные части примитивов.

Пункт главного **меню Помощь** открывает ниспадающее меню, опции которого предназначены для предоставления пользователю в любой момент времени дополнительной справочной информации:

- о наборе команд AutoCAD.
- о каждой команде в отдельности – контекстная спрвка.
- о геометрических характеристиках объектов – координатах точек, углах.
- о площади, периметре и другой вспомогательной вычисляемой информации.

Примитивы

В отличие от "художественных" графических редакторов, AutoCAD работает не с изображением как таковым, а с геометрическим описанием объектов, составляющих изображение. Так, например, отрезок во внутреннем представлении графического редактора AutoCAD описывается двумя точками, круг описывается центром и радиусом.

Все примитивы AutoCAD обладают рядом свойств (принадлежность слою, цвет, тип линии, ширина). Некоторые из этих свойств (например, цвет) присущи всем примитивам.

Системы координат

Используется традиционная декартова система координат. Можно ввести пользовательские системы координат с помощью команды USC. В определенный момент времени пользователь работает только с одной предварительно выбранной системой координат, которая называется текущей. Вся работа с изображением проводится в текущей системе координат.

Единицы измерения и масштаб

Расстояния между точками на рисунке измеряются в условных единицах. Конкретный формат представления размеров (дюймы, футы, миллиметры и др.) не имеет значения для AutoCAD. Иначе говоря, при создании объектов в чертеже AutoCAD "измеряет" все расстояния в относительных единицах. В AutoCAD нет масштаба в обычном понимании конструктора, конструктор задает все расстояния и координаты в реальных единицах - мы как бы работаем в масштабе 1:1. Масштабирование различных частей изображения в соответствии с желаемым форматом документа может осуществляться в момент компоновки чертежа (команда SCALE (МАСШТАБ)) или при выводе чертежа или его части на плоттер (принтер).

Вид

Когда вы создаете чертеж, то работаете с изображением части чертежа, выводимой на дисплей. Будем называть эту часть изображения видом. Эту видимую часть чертежа (окно зрения) можно увеличивать (при этом изображение чертежа будет уменьшаться), уменьшать (изображение будет увеличиваться) или перемещать по полю чертежа без изменения масштаба отображения (панорамирование). Изменение вида осуществляется командой ZOOM.

Слой

AutoCAD дает возможность распределять выбранные фрагменты чертежа по различным слоям. Работая за кульманом, конструктор имеет дело с одним листом бумаги и располагает изображения объектов на нем и только на нем. В среде AutoCAD располагать изображение можно как бы на нескольких совмещенных в пространстве носителях (это можно сравнить с наложенными друг на друга прозрачными кальками). Например, чертеж может содержать на одном слое построение призмы со всеми вспомогательными линиями, на другом - сам чертеж в окончательном исполнении, а на третьем - ход лучей в призме. Количество слоёв не ограничивается. Слои можно делать видимыми и невидимыми. С каждым слоем чертежа связывается цвет и тип линий. По мере создания чертежа вы можете вводить новые слои, менять свойства существующих. Подобно системе координат, в любом чертеже AutoCAD всегда существует по крайней мере один слой с именем "0".

Чертёж

Чертёж - это файл, содержащий некую графическую и вспомогательную информацию, полностью описывающую графический объект. AutoCAD предоставляет в распоряжение конструктора все вычислительные ресурсы микроЭВМ и новые средства создания и редактирования чертежа.

Вместе с каждым рисунком AutoCAD содержит так называемые ***системные переменные***, в которые заносится определенная информация: о текущих установках рисования (т. е. установках слоя, цвета, типа линий и т. п.), о последнем выполненном действии (т. е. имя последней команды, последняя точка, последний радиус и т. п.), о настройках некоторых команд (т. е. длины фаски, радиус сопряжения и т. п.) и многое другое. Пользователь может вывести на экран перечень и значения системных переменных, и большую часть из них изменить. Остальные изменяются самой системой в процессе работы.

Система AutoCAD позволяет настраивать многие элементы пользовательского интерфейса. Параметры настройки формируются уже на стадии установки AutoCAD на ваш компьютер — большая часть по умолчанию, а что-то (например, размещение папок для программного обеспечения) задает пользователь. Для изменения установок нужно воспользоваться либо командой НАСТРОЙКА (OPTIONS), либо пунктом **Настройка...** (Options...) падающего меню **Сервис** (Tools), либо пунктом **Настройка...** (Options...) контекстного меню, вызываемого по щелчку правой кнопкой мыши в зоне командных строк.

Типы примитивов

Примитивы могут быть простыми и сложными. К простым примитивам относятся следующие объекты: точка, отрезок, круг (окружность), дуга, прямая, луч, эллипс, сплайн, текст.

К сложным примитивам относятся: полилиния, мультилиния, мультитекст, размер, выноска, допуск, штриховка, вхождение блока или внешней ссылки, атрибут и растровое изображение.

Способы ввода координат точек

Можно задавать конечные точки отрезка с помощью мыши. Но этот способ ввода (указания) точек не является единственным. Больше распространен **второй способ** — ввод координат точки с клавиатуры, например: **65,113.24**

В данном примере введена точка с двумя координатами: X=65 мм, Y=113.24 мм. При вводе координат с клавиатуры запятая является разделителем между абсциссой и ординатой, а точка используется как разделитель между целой и дробной частью числа. Вводимые координаты могут быть целыми или вещественными. При вводе координат следует учитывать, где вы выбрали точку с координатами 0,0. Чаще всего это точка левого угла графического экрана (хотя в процессе работы вы перемещаетесь по рисунку, и точка 0,0 может оказаться в любом месте, даже уйти в невидимую часть чертежа).

Третий способ ввода точек — это относительный ввод в декартовых координатах с клавиатуры, например: @50,25

Данная запись означает, что новая точка задается относительно предыдущей (что определяет символ "@"), со сдвигом по оси X на +50 мм (т. е. вправо на 50 мм) и сдвигом по оси Y на +25 мм (т. е. вверх на 25 мм). Здесь запятая также является разделителем координат. Вводимые числа могут быть целыми и вещественными, положительными, нулевыми и отрицательными.

Четвертый способ ввода точек — это относительный ввод в полярных координатах с клавиатуры, например: @33.5<45

В этой форме записи уже нет запятых, зато появился символ "<", который интерпретируется как знак угла. В данном примере новая точка задается относительно предыдущей, причем расстояние между ними в плоскости равно 33,5 мм (т. е. числу влево от символа угла), а вектор из предыдущей точки в новую образует угол 45 градусов с положительным направлением оси абсцисс (угол измеряется в тех угловых единицах, которые вы задали в настройке). Расстояние должно обязательно быть положительным, а угол может быть любым числом.

Пятый способ ввода точек — это указание с помощью функций объектной привязки. Доступ к функциям объектной привязки осуществляется либо через групповую кнопку панели **Стандартная** (Standard), либо через панель **Объектная**

привязка (Object Snap). *Групповой* называется кнопка, у которой в правом нижнем углу имеется черный треугольник. Если выбрать указателем мыши такую кнопку и нажать (не отпуская!) левую кнопку мыши, то раскроется набор кнопок инструментов, которые входят в данную группу. Нужно опуститься по появившимся кнопкам до той, которая вам нужна, и только тогда отпустить нажатую левую кнопку мыши, однако объектной привязкой лучше пользоваться, имея на экране одноименную панель.

Получение справок

В процессе работы очень полезными оказываются команды получения справочной информации о создаваемых объектах. Групповая кнопка справочных операций позволяет получить информацию об:

- **Расстоянии** (Distance)
- **Площади** (Area)
- **Массе** (Mass Properties)
- **Списке** (List)
- **Координатах** (Locate Point).

Команды общего редактирования

Кнопки команд общего редактирования объектов (копирование, перенос, удлинение и т. п.) расположены в панели **Редактирование** (Modify). Каждую из этих команд можно ввести по имени с клавиатуры, а также вызвать с помощью падающего меню **Редакт** (Modify). Многие команды данной группы работают либо с набором предварительно выбранных объектов, либо при отсутствии такого набора выдают запрос **Выберите объекты:** (Select objects:). Остальные команды запрашивают редактируемые объекты в соответствующий момент времени.

Свойства.

У каждого примитива могут быть свои цвет, слой, тип линии, масштаб типа линии, стиль печати, вес линии, гиперссылка и высота — все это в AutoCAD отнесено к ***свойствам***. Напомню, что определить текущие

значения свойств объекта можно, например, с помощью команды СПИСОК (LIST).

При создании сложных рисунков возникает необходимость присвоения имен отдельным объектам или группам объектов, чтобы ими можно было удобнее оперировать в дальнейшей работе. Особенно это важно при разработке своих собственных приложений, функционирующих в среде AutoCAD. Данной цели служит еще одно свойство примитивов — слой. Более того, слой обладает неоценимой возможностью замораживания (выключения), когда ряд второстепенных в данный момент объектов можно, не удаляя, сделать невидимыми, что позволит успешнее работать с главными объектами.

Вес линии — совершенно новое свойство примитивов, которое отсутствовало в предыдущих версиях системы AutoCAD — это толщина, с которой объект будет выводиться на устройство печати (или графопостроитель). Вы можете нарисовать объекты тонкой линией, но задать ненулевой вес и получить при этом жирные линии на листе бумаги.

Высота — это свойство примитива, применяемое в трехмерных построениях. Оно задает величину выдавливания вдоль оси Z, расположенной перпендикулярно осям X и Y. Например, чтобы круг преобразовать в цилиндр, его нужно выдавить на ненулевую высоту.

В рисунках системы AutoCAD могут присутствовать описания стилей некоторых объектов, что, конечно, облегчает оформление чертежа. К таким стилям относятся: текстовые, размерные и стили мультилиний.

Вывод на плоттер.

Под понятием "плоттер" будем иметь в виду не только графопостроитель, но и любое другое устройство вывода, в том числе и принтер. Стоит заметить, что для современных устройств, использующих струйную и лазерную технологию, практически отсутствует грань между принтерами и плоттерами. Поэтому принтерами часто называют плоттеры небольшого формата (не превышающего А2). Поскольку в меню и в

документации системы AutoCAD чаще используется термин плоттер, то мы будем применять именно его для обозначения любого устройства вывода.

Любое устройство (локальное или сетевое), к которому вы обращаетесь для вывода чертежа из AutoCAD, должно быть специальным образом конфигурировано (описано) в системе AutoCAD. Операция по установке плоттеров или редактированию их настроек требует специальных знаний. Ее лучше выполнять опытным пользователям или системным программистам, обслуживающим вычислительные комплексы, на которых функционирует AutoCAD. Дополнительную информацию можно найти либо в справочной системе AutoCAD, либо в документации, поставляемой вместе с системой. Автономная настройка (т. е. не зависящая от связи с AutoCAD) самого устройства выполняется с помощью документации, поставляемой вместе с плоттером.

Для того чтобы определить, настроена ли ваша версия AutoCAD, а если настроена, то на плоттеры каких марок, следует воспользоваться командой НАСТРОЙКА (OPTIONS). Эту команду можно вызвать либо с помощью пункта **Настройка** (Options) падающего меню **Сервис** (Tools), либо с помощью контекстного меню, появляющегося при нажатии правой кнопки мыши, если ее указатель расположен в этот момент в зоне командных строк. Команда НАСТРОЙКА (OPTIONS) вызывает диалоговое окно **Настройка** (Options).

Для работы с наиболее распространенными плоттерами и форматами графических файлов в системе AutoCAD присутствуют специальные программы (драйверы), обеспечивающие передачу данных на соответствующие устройства или в соответствующие форматы. AutoCAD 2000 в стандартной поставке поддерживает большое количество перьевых и струйных типов плоттеров таких фирм, как Hewlett-Packard, Xerox, Oce, CalComp и Houston Instruments, а также наиболее распространенные форматы растровых файлов (JPEG, BMP, PNG, TGA и др.) и форматы PostScript, применяемые в лазерных устройствах печати.

Стили печати

Стили печати — это новое свойство, отсутствовавшее в предыдущих версиях AutoCAD, которое отображает графические объекты при выводе на плоттер специальным образом. Таким образом, примитив в рисунке может на экране выглядеть совсем не так, как он будет нарисован плоттером на бумаге. Изменяться может цвет, тип, а также вес линии. Можно также задать специальное оформление концов и заливки линии. Все такие установки заносятся в таблицы стилей. Система AutoCAD при установке создает ряд стандартных таблиц стилей печати, которые доступны пользователю.

Стили печати могут быть двух видов: именованные и цвето-зависимые. Именованный стиль печати может быть назначен любому объекту, а цвето-зависимый стиль используется в зависимости от цвета примитива.

Цвето-зависимые стили, которых в каждой таблице 255 (по количеству цветов системы AutoCAD), описывают, каким образом нужно выводить на плоттер объекты, имеющие данный цвет. Такие стили удобны для вывода на перьевой плоттер, который имеет ограниченное количество цветов и размеров перьев. По умолчанию, когда имя таблицы действующего цвето-зависимого стиля не задано, действует стиль, который выводит объекты в том виде, в каком они созданы в рисунке.

Блоки и внешние ссылки.

Важным инструментом автоматизации процесса разработки чертежей является использование блоков и внешних ссылок. ***Блок.*** — это сложный именованный объект, для которого создается описание, состоящее из любого количества примитивов системы AutoCAD текущего рисунка. Блок имеет базовую точку и может применяться для вставки в любое место чертежа, причем в процессе вставки возможен его поворот и масштабирование с различными коэффициентами по разным осям. Примитив, который образуется от операции вставки блока, называется ***ВХОЖДЕНИЕ БЛОКА (BLOCK REFERENCE)***. В рисунке может быть любое количество вхождений одного и того же блока.

Внешняя ссылка — это изображение внешнего файла вместе с элементами текущего рисунка, причем файл, на который Вы таким образом ссылаетесь, не переносится в основной рисунок. В результате текущий рисунок может быть насыщен большим количеством внешних изображений новых объектов, но размер текущего файла от этого практически не увеличится. Примитив, образующийся от операции вставки внешней ссылки, будем называть ***вхождением внешней ссылки*** или просто ***внешней ссылко***.

Оба упомянутых инструмента являются средством автоматизации труда конструктора и чертежника. С помощью блоков можно строить однотипные объекты, описывая полностью только один их них. Внешние ссылки дают возможность пользоваться ранее созданными файлами стандартных графических элементов.

Первый шаг к использованию блока — создать описание этого блока. Для этого нужно определиться, из каких примитивов будет состоять блок и где у него будет базовая точка. Объекты, которые были включены в блок при его описании, сохраняют свои основные свойства (слои, цвет, тип линии, вес) и во вставленном блоке. Исключением является специальное значение ПОБЛОКУ, которое может быть дано цвету, типу линии и весу. Таким образом, любые части рисунков могут сохраняться в виде отдельных файлов, а любые созданные файлы могут вставляться в текущий рисунок с образованием (или без образования) блоков.

Часто возникает необходимость вместе с блоком держать и надписи, которые могли бы менять свои значения после вставки блока. Например, если Вы рисуете схему с использованием заранее подготовленных блоков условных элементов, тогда номера или наименования вставленных графических элементов Вам нужно будет оформить в виде текстовых надписей. Однако в системе AutoCAD есть специальный примитив, называемый **ОПИСАНИЕ АТРИБУТА (ATTRIBUTE DEFINITION)**, который может быть включен в описание блока, а при операции вставки

этого блока будет запрошено его значение и создан атрибут (текстовая строка), входящий в состав блока.

Атрибуты могут содержать текстовую информацию, которая дополняет графические примитивы рисунка. Извлечение значений атрибутов может быть сделано с помощью специальной команды **АТЭКСП (ATTEXT**), которая выводит извлекаемые данные в текстовый файл. Эта операция полезна при создании систем автоматизированного проектирования на базе AutoCAD.

Внешние ссылки

Вставка с помощью команды **ВСТАВИТЬ (INSERT)** одного файла рисунка в другой рисунок, который является текущим, увеличивает его объем, т. к. в него переносятся примитивы вставляемого файла. Но есть еще один способ добавить к текущему рисунку изображение другого рисунка — вставить файл с помощью внешней ссылки. При этом вставляемый файл в текущий рисунок не переносится, а только запоминается его полное имя (обычно вместе с путем). В дальнейшем, когда AutoCAD открывает рисунок, имеющий внешнюю ссылку, то загружается сначала открываемый файл, а затем — содержимое дополнительного файла-ссылки. Таким образом, файл-ссылка не хранится вместе с основным рисунком. Разумеется, при таком варианте основной файл имеет меньший размер по сравнению с вариантом вставки файла с помощью команды **ВСТАВИТЬ (INSERT)**, но попадает в зависимость от дополнительного файла, т. к. тот должен всегда обнаруживаться на своем привычном месте и не менять своего имени. Возможны вложенные ссылки, когда ссылка выполняется на вставляемый файл, который сам содержит внешнюю ссылку на другой файл. Команда **ССЫЛКА (XREF)** управляет в текущем рисунке внешними ссылками на другие файлы.

Вставка объектов, созданных другими системами.

AutoCAD может читать ряд других графических форматов и вставлять объекты, созданные другими известными приложениями (например, Microsoft Office).

Вставка и редактирование растровых изображений

Растровое изображение — это изображение, состоящее из точек (растров), которые благодаря цветам формируют рисунок. AutoCAD может прочитать файл с растровой картинкой и вставить его в текущий рисунок в виде цветного прямоугольника (аналогично внешней ссылке). Редактировать вставленное изображение на точечном уровне AutoCAD не может, но может выполнять подрезку, масштабирование, перенос и другие простые операции редактирования. При наложении одного растрового изображения на другое можно управлять порядком их следования (переносить на передний план или убирать на задний).

Для операций с растровыми изображениями используется команда **ИЗОБ (IMAGE)**, которой соответствует кнопка **Изображение** (Image) панели **Ссылки** (Reference), а также кнопка **Изображение** (Image) панели **Вставка** (Insert) и пункт **Диспетчер изображений...** (Image Manager...) падающего меню **Вставка** (Insert).

Трехмерные построения.

AutoCAD может строить примитивы не только в плоскости XY, но в любой плоскости трехмерного пространства. Кроме того, в системе AutoCAD существует большой набор пространственных примитивов (поверхностей, тел и др.), которые позволяют выполнять построения трехмерных моделей зданий, сооружений и различных машиностроительных изделий.

Можно не только строить трехмерные объекты, но и рассматривать их в разных видах и проекциях, используя новые системы координат. Имеются такие возможности AutoCAD, как скрытие невидимых линий, тонирование и назначение объектам тех или иных материалов. Все построенные модели

можно с помощью пространства листа, оформлять красиво и удобно в виде чертежей.

Работа по модернизации чертежа

Общеизвестно, что в процессе проектирования чертежа конструктор много времени тратит на редактирование. Несмотря на то, что мы с вами, возможно, традиционным способом сможем начертить новый чертеж быстрее, чем с помощью системы AutoCAD, наверняка для исправления его нужно будет потратить немалые усилия, т.е. извести массу стирательной резинки, пожертвовать товарным видом чертежа в пользу его правильности или долго и нудно перечерчивать чертеж. С использованием редакторских возможностей AutoCAD мы можем значительно облегчить свою жизнь.

Функции редактирования позволяют:

- удалять фрагменты изображения;
- восстанавливать случайно удаленные фрагменты;
- перемещать или поворачивать фрагменты или отдельные изображения относительно других;
- копировать созданные фрагменты и располагать их в указанном месте;
- увеличивать или уменьшать объекты;
- создавать зеркально-симметричное изображение;
- изменять свойства (принадлежность к слою, цвет и тип линии) созданных объектов;
- сопрягать линии и строить фаски;
- делить объекты на равные части или размечать на сегменты с заданным интервалом;
- расчленять блоки или полилинии на составные части;
- редактировать полилинии (сглаживать, изменять свойства и т. д.);
- растягивать части рисунка;
- проводить линии, расположенные на заданном (постоянном) расстоянии относительно других.

Средства реализации систем автоматизации конструкторской документации предоставляет компьютерная графика, обеспечивающая создание, хранение и обработку моделей геометрических объектов и их графических изображений с помощью компьютера. Автоматизация особенно эффективна при разработке устройств на базе параметрически-управляемых унифицированных и типовых элементов конструкций, обеспечивающих многовариантность конструирования. Модель геометрических объектов, содержащая информацию о геометрии объекта, используется как для получения двумерной геометрической модели, так и для расчета различных характеристик объекта и технологических параметров его изготовления. Отсюда следует, что геометрическое моделирование является ядром автоматизированного конструирования и технологической подготовки производства.

Система автоматизации разработки и выполнения конструкторской документации выполняет ввод, хранение, обработку и вывод графической информации в виде конструкторских документов. Для реализации системы необходимы: документы, регламентирующие работу системы, исходная информация для формирования информационной базы, информационная база, содержащая модели геометрических объектов и графических изображений, элементы оформления чертежа по ГОСТу, технические и программные средства создания моделей геометрических объектов и графических изображений и их вывода, интерфейс пользователя в виде графического диалога с компьютером.

Построение таких систем значительно упрощается, если они создаются на базе универсальной, открытой среды проектирования для реализации графических возможностей САПР. Примером данной среды является AutoCad – универсальная графическая система, в основу структуры которой положен принцип открытой архитектуры, позволяющей адаптировать и развивать многие функции AutoCad применительно конкретным задачам и требованиям.

Можно выделить два подхода к конструированию на основе компьютерных технологий. Первый подход базируется на двумерной геометрической модели графического изображения и использовании компьютера как электронного кульмана, позволяющего значительно ускорить процесс конструирования и улучшить качество оформления конструкторской документации. Центральное место при таком подходе занимает чертеж, который служит средством представления изделия и содержит информацию для решения графических задач, а также для изготовления изделия. В таком случае получение графического изображения за компьютером будет рациональным и достаточно эффективным, если созданное графическое изображение используется многократно.

В основе второго подхода лежит пространственная геометрическая модель изделия, которая является более наглядным способом представления оригинала и более мощным и удобным инструментом для решения геометрических задач. Чертеж в этом случае играет вспомогательную роль, а способы его создания основаны на методах компьютерной графики и отображения пространственной модели.

При первом подходе обмен информацией осуществляется на основе конструкторской, нормативно-справочной и технологической документации, при втором подходе – на основе компьютерного представления геометрических объектов, общей базы данных, что способствует эффективному функционированию программного обеспечения САПР конкретного изделия.

Можно выделить два основных вида геометрических объектов:

- постоянный, с постоянными размерами и геометрической формой. Например: графические изображения условных графических обозначений радиоизделий электрических схем или печатных плат.
- параметрических заданный, с переменными размерами и геометрической формой. Например: радиоизделие, зависящее от

типоразмеров, типовые и унифицированные несущие конструкции радиоэлектронных устройств, конструктивные элементы типовых деталей.

Постоянные геометрические объекты могут быть сформированы с использованием графического редактора AutoCad. Методы описания параметрических заданных геометрических объектов характеризуются большими затратами на формирование компьютерного представления. Чтобы сократить эти затраты, при описании некоторых групп технических объектов можно пользоваться одним из двух принципиально различных методов: вариантным или генерирующим.

Вариантный метод основан на том, что для определенного класса изделий выявляется модель-образец, с помощью которой можно получить все геометрические формы этого класса изделий. Исполнение изделия определяется заданными параметрами, обнуление которых приводит к исключению составных элементов геометрического объекта. В простейшем случае изменяют только размеры, а конструкция отдельных вариантов класса изделий остается неизменной. Такой вид конструирования называют принципиальным. В этом методе данные технологической документации не подготавливаются каждый раз заново, а закреплены за уже имеющимися соответствующими чертежами. Применение такого метода предполагает, что выбор геометрии для проектируемого изделия уже сделан. Затраты на описание типовой модели превышают затраты на получение вариантов, поэтому во многих системах используется принцип вложенности моделей: один раз описанные типовые модели используются для формирования других типовых моделей в качестве макрокоманд.

В противоположность вариантному методу при генерирующем определяются различные сочетания конструктивных и технологических элементов и выбирается наилучшее решение. Принцип работы системы, использующей генерирующий метод, основан на разделении геометрического объекта на элементы и создании новых геометрических

объектов из имеющихся элементов. Различают следующие группы элементов: основные (функциональные), вспомогательные (конструктивные геометрические и элементы формы) и технологические. С помощью основных элементов создается геометрическая форма детали (наружные и внутренние поверхности), проточки (внутренние и наружные). Это дает прежде всего общее описание детали. С помощью вспомогательных элементов, которые непосредственно связаны с основными, осуществляется более подробное описание детали, что позволяет полностью передать ее геометрическую форму. Технологические элементы или характеристики относятся и к основным, и к вспомогательным элементам. Они также влияют на простановку размеров. САПР, работающие по генерирующему принципу, обладают высокой гибкостью и пригодны для решения множества задач. Использование данного метода эффективно, так как большинство конструкторских разработок, называемых новыми конструкциями, создается путем ранее неиспользовавшегося сочетания элементов, давно известных как по принципу формирования, так и по исполнению.

Вопросы для самоконтроля

1. Компьютерная графика
2. Виды компьютерной графики.
3. Графическое моделирование.
4. В каких режимах может работать программа **AutoCAD**?
5. Что такое модель и лист?
6. Элементы интерфейса программа **AutoCAD**
7. Вставка и редактирование растровых изображений
8. Команды общего редактирования

2.4. Особенности имитационного моделирования в технических системах

Основные модули

- Имитационные модели.
- Области применения имитационного моделирования.
- Применение имитационного моделирования.
- Этапы имитационного моделирования

Имитационные модели представляют собой, совокупность программ, программный комплекс, позволяющие с помощью последовательности вычислений и графического отображения их результатов воспроизводить (имитировать) процессы функционирования объекта при условии воздействия на него различных (включая случайные) факторов. Эвристические модели базируются на правилах, которые выбираются интуитивно или имперически (правилах), которые позволяют улучшить уже имеющиеся решения.

Имитационное моделирование (ИМ) - это распространенная разновидность аналогового моделирования, реализуемого с помощью набора математических инструментальных средств, специальных имитирующих компьютерных программ и технологий программирования, позволяющих посредством процессов-аналогов провести целенаправленное исследование структуры и функций реального сложного процесса в памяти компьютера в режиме «имитации», выполнить оптимизацию некоторых его параметров.

Имитационная модель - специальный программный комплекс, позволяющий имитировать деятельность какого-либо сложного объекта.

Современные программные комплексы имитационного моделирования:

- «Process Charter» (разработчик - фирма «Scitor», США);
- «Powersim» (фирма «Modell Data», Норвегия);
- «Ithink» (фирма «High Performance Systems», США);

- «Extend+BPR» (фирма «Imagine That!», США);
- «ReThink» (фирма «Gensym», США);
- «Pilgrim» (разработчики системы - Московский институт статистики и информатики; фирма «МегаТрон», Россия; фирма «Keisy», Нидерланды; фирма «Enit», Эстония).

При моделировании процессов не обязательно преобразовывать математическую модель в специальную систему уравнений относительно искомых величин. Для имитационного моделирования характерно воспроизведение явлений, описываемых математической моделью, с сохранением их логической структуры, последовательности чередования во времени, а иногда и физического содержания, выполняемое при помощи специальных моделирующих программ. В противоположность аналитическому и численному методам содержание операций, выполняемых при моделировании, слабо зависит от того, какие величины выбраны в качестве искомых. Для оценки искомых величин может быть использована любая подходящая информация, циркулирующая в модели, если только она доступна регистрации и последующей обработке. Все имитационные модели представляют собой модели типа так называемого «черного ящика». Это означает, что они обеспечивают выдачу выходного сигнала системы, если на ее взаимодействующие подсистемы поступает входной сигнал. Поэтому для получения необходимой информации или результатов необходимо осуществлять «прогон» имитационных моделей, а не «решать» их. Имитационные модели не способны формировать свое собственное решение в том виде, в каком это имеет место в аналитических моделях, а могут лишь служить в качестве средства для анализа поведения системы в условиях, которые определяются экспериментатором.

Среди методов прикладного системного анализа имитационное моделирование является самым мощным инструментом исследования сложных систем, управление которыми связано с принятием решений в

условиях неопределенности. По сравнению с другими методами такое моделирование позволяет рассматривать большое число альтернатив, улучшать качество управленческих решений и точнее прогнозировать их последствия.

Отметим области применения имитационного моделирования.

1. Теоретические задачи в различных областях науки (математика, физика, химия). Например, при вычислении площади фигур, ограниченными кривыми, при обращении матриц, при вычислении, при решении дифференциальных уравнений в частных производных; при анализе диффузии частиц, при получении решения системы дифференциальных уравнений.

2. Практические задачи организационного управления, возникающие в различных сферах человеческой деятельности:

а) задачи ИМ произвольно-технологических процессов. Например, анализ химических процессов управления запасами, проектирование систем технического обслуживания оборудования, создание СМО;

б) задачи социального и социально-психологического характера. Например, задачи миграции населения, исследование группового поведения;

в) задачи ИМ биомедицинских систем. Например, исследование кровообращения, деятельность мозга;

г) задачи анализа последствий реализации военной стратегии или тактики.

3. Задачи ИМ систем экономического характера:

- для управления сложным бизнес-процессом, когда имитационная модель управляемого экономического объекта используется в качестве инструментального средства в контуре адаптивной системы управления, создаваемой на основе информационных (компьютерных) технологий;

- при проведении экспериментов с дискретно-непрерывными моделями сложных экономических объектов для получения и отслеживания их динамики в экстренных ситуациях, связанных с рисками, натурное

моделирование которых нежелательно или невозможно.

Например, процессы планирования экономического прогнозирования, инвестиционные процессы.

Примеры типовых задач, решаемых средствами имитационного моделирования при управлении экономическими объектами:

- моделирование процессов логистики для определения временных и стоимостных параметров;
- управление инвестиционными проектами на различных этапах его жизненного цикла с учетом возможных рисков;
- прогнозирование финансовой деятельности предприятий на конкретный период времени;
- анализ адаптивных свойств, расчет параметров надежности и задержек обработки информации в банковских информационных системах;
- оценка параметров надежности и задержек в централизованных информационных системах с коллективным доступом (кассы продаж ж/д и авиабилетов, системы бронирования и т.д.;
- анализ эксплуатационных параметров корпоративных информационных систем предприятий, пропускной способности информационных каналов и узлов обработки информации;
- моделирование действий курьерской службы в регионе;
- анализ пропускной способности обслуживания населения (торговые комплексы, комбинаты бытового обслуживания, государственные структуры и т.д.).

Отметим **преимущества ИМ в плане целесообразности применения**:

1. Не существует законченной математической постановки данной задачи, либо еще не разработаны аналитические методы решения сформулированной математической модели. К этой категории относятся многие модели массового обслуживания, связанные с рассмотрением очередей.

2. Аналитические методы имеются, но математические процедуры столь сложны и трудоемки, что ИМ даст более простой способ решения задачи.

3. Аналитические решения существуют, но их реализация невозможна вследствие недостаточной математической подготовки имеющегося персонала. В этом случае следует сопоставить затраты на проектирование, испытания и работу на имитационной модели с затратами, связанными с приглашением специалистов со стороны.

4. Кроме оценки определенных параметров, желательно осуществить на имитационной модели наблюдение за ходом процесса в течение определенного периода.

5. Имитационное моделирование может оказаться единственной возможностью вследствие трудности постановки экспериментов и наблюдения явлений в реальных условиях; соответствующим примером может служить изучение поведения космических кораблей в условиях межпланетных полетов.

6. Для долговременно действующих систем или процессов может понадобиться сжатие временной шкалы. ИМ дает возможность полностью контролировать время изучаемого процесса, поскольку явление может быть замедленно или ускоренно по желанию.

Дополнительным преимуществом ИМ можно считать широчайшие возможности его применения в сфере образования и профессиональной подготовки. Разработка и

использование имитационной модели позволяют экспериментатору видеть и «разыгрывать» на модели реальные процессы и ситуации. Это в свою очередь должно в значительной мере помочь ему понять и прочувствовать проблему, что стимулирует процесс поиска нововведений.

К недостаткам ИМ можно отнести следующее:

1. Разработка хорошей ИМ часто обходится дорого и требует много времени, а также наличия высокоодаренных специалистов, которых в данной

фирме может и не оказаться.

1. Ввиду работы со случайными данными для обеспечения точности получаемых результатов необходимо производить многочисленные прогоны модели, что влияет на техническое средство работы с имитационной моделью.

2. Имитационная модель в принципе не точна, и мы не в состоянии измерить степень этой неточности. Это затруднение может быть преодолено лишь частично путем анализа чувствительности модели к изменению определенных параметров._

Идея ИМ одинакова привлекательна и для руководителей, и для исследователей систем благодаря своей простоте. Поэтому метод ИМ в настоящее время стремятся применить для решения практически каждой задачи, с которой приходится сталкиваться на практике.

Применение имитационного моделирования

Имитационное моделирование (от англ. simulation) — это распространенная разновидность аналогового моделирования, реализуемого с помощью набора математических инструментальных средств, специальных имитирующих компьютерных программ и технологий программирования, позволяющих посредством процессов-аналогов провести целенаправленное исследование структуры и функций реального сложного процесса в памяти компьютера в режиме «имитации», выполнить оптимизацию некоторых его параметров.

Имитационной моделью называется специальный программный комплекс, который позволяет имитировать деятельность какого-либо сложного объекта. Он запускает в компьютере параллельные взаимодействующие вычислительные процессы, которые являются по своим временным параметрам (с точностью до масштабов времени и пространства) аналогами исследуемых процессов.

Для этого вида моделирования используется синоним компьютерное моделирование. Так как имитационную модель нужно создавать, то для этого

необходимо специальное программное обеспечение — система моделирования (simulation system). Специфика такой системы определяется технологией работы, набором языковых средств, сервисных программ и приемов моделирования.

Имитационная модель должна отражать большое число параметров, логику закономерности поведения моделируемого объекта во времени (временная динамика) и в пространстве (пространственная динамика). Моделирование объектов экономики связано с понятием финансовой динамики объекта. С точки зрения специалиста (информатика-экономиста, математика-программиста или экономиста-математика), имитационное моделирование контролируемого процесса или управляемого объекта — это высокоуровневая информационная технология, которая обеспечивает два вида действий, выполняемых с помощью компьютера: работы по созданию или модификации имитационной модели; эксплуатацию имитационной модели и интерпретацию результатов; имитационное (компьютерное) моделирование экономических процессов обычно применяется в двух случаях:

для управления сложным бизнес-процессом, когда имитационная модель управляемого экономического объекта используется в качестве инструментального средства в контуре адаптивной системы управления, создаваемой на основе информационных (компьютерных) технологий;

при проведении экспериментов с дискретно-непрерывными моделями сложных экономических объектов для получения и отслеживания их динамики в экстренных ситуациях, связанных с рисками, натурное моделирование которых нежелательно или невозможно.

Этапы имитационного моделирования

Имитационное моделирование как особая информационная технология состоит из следующих основных этапов:

1.Структурный анализ процессов. Проводится формализация структуры сложного реального процесса путем разложения его на подпроцессы, выполняющие определенные функции и имеющие взаимные функциональные связи согласно легенде, разработанной рабочей экспертной группой. Выявленные подпроцессы, в свою очередь, могут разделяться на другие функциональные подпроцессы.

Структура общего моделируемого процесса может быть представлена в виде графа, имеющего иерархическую многослойную структуру, в результате появляется формализованное изображение имитационной модели в графическом виде. Структурный анализ особенно эффективен при моделировании экономических процессов, где (в отличие от технических) многие составляющие подпроцессы не имеют физической основы и протекают виртуально, поскольку оперируют с информацией, деньгами и логикой (законами) их обработки.

2. Формализованное описание модели. Графическое изображение имитационной модели, функции, выполняемые каждым подпроцессом, условия взаимодействия всех подпроцессов и особенности поведения моделируемого процесса (временная, пространственная и финансовая динамика) должны быть описаны на специальном языке для последующей трансляции.

3. Построение модели. Обычно это трансляция и редактирование связей (сборка модели), верификация (калибровка) параметров. Трансляция осуществляется в различных режимах: в режиме интерпретации, или в режиме компиляции. Каждый режим имеет свои особенности. Режим интерпретации проще в реализации. Специальная универсальная программа-интерпретатор на основании формализованного описания модели запускает все имитирующие подпрограммы.

Данный режим не приводит к получению отдельной моделирующей программы, которую можно было бы передать или продать заказчику (продавать пришлось бы и модель, и систему моделирования, что не всегда

возможно). Режим компиляции сложнее реализуется при создании моделирующей системы. Однако это не усложняет процесс разработки модели. В результате можно получить отдельную моделирующую программу, которая работает независимо от системы моделирования в виде отдельного программного продукта. Верификация (калибровка) параметров модели выполняется в соответствии с легендой, на основании которой построена модель, с помощью специально выбранных тестовых примеров.

4. Проведение экстремального эксперимента для оптимизации определенных параметров реального процесса.

Возможен другой подход к определению основных этапов моделирования:

1. Разработка имитационной модели;

2. Разработка методики моделирования (планирование имитационного эксперимента);

3. Программная реализация модели (выбор средств — универсальных языков программирования либо специализированных языков моделирования);

4. Выполнение имитационного моделирования, анализ и обобщение результатов, принятие решений.

Вопросы для самоконтроля

1. Имитационные модели.
2. Области применения имитационного моделирования.
3. Применение имитационного моделирования.
4. Этапы имитационного моделирования

ГЛАВА III. СЕТЕВЫЕ ВОЗМОЖНОСТИ В ТЕХНИЧЕСКИХ СИСТЕМАХ

3.1. Сетевые технологии и облачные сервисы

Основные модули

- Компьютерные сети.
- Соединение компьютеров.
- Локальные, региональные и глобальные сети.
- Структура и архитектура компьютерных сетей.
- Кабельные и беспроводные сети.
- Облачные технологии

Компьютерная сеть - объединение нескольких ЭВМ для совместного решения информационных, вычислительных, учебных и других задач.

Все компьютерные сети без исключения имеют одно назначение- обеспечение совместного доступа к общим ресурсам. Ресурсы бывают трех видов: *аппаратные, программные, информационные.*

Аппаратные ресурсы – это, когда все участники компьютерной сети пользуются одним аппаратом, например, принтером или используют один компьютер с увеличенной емкостью жесткого диска (файловый сервер), на котором хранят свои архивы и результаты работы.

Компьютерные сети позволяют совместно использовать ***программные ресурсы***. Так, например, для выполнения сложных и продолжительных расчетов можно подключиться к удаленной большой ЭВМ и отправить вычислительное задание на нее, по окончании расчетов получить результат обратно.

Данные, хранящиеся на удаленных компьютерах, образуют **информационный ресурс**, например, Интернет.

По способу организации сети подразделяются на *реальные и искусственные.*

Искусственные сети (псевдосети) позволяют связывать компьютеры вместе через последовательные или параллельные порты и не нуждаются в дополнительных устройствах.

Реальные сети позволяют связывать компьютеры с помощью специальных устройств коммутации и физической среда передачи данных.

По территориальной распространенности сети могут быть *локальными, глобальными, региональными и городскими.*

Глобальная (крупномасштабная) вычислительная сеть WAN (Wide Area Network) представляет собой множество географически удаленных друг от друга компьютеров, совместное взаимодействие которых обеспечивается коммуникационной сетью передачи данных и сетевым программным обеспечением. Основу WAN составляют мощные вычислительные системы, являющиеся различного рода серверами, а также специализированные компьютеры, выполняющие функции коммуникационных узлов. Пользователи персональных компьютеров становятся абонентами сети посредством подключения своих ЭВМ именно к этим вычислительным или коммуникационным узлам.

WAN может носить как ведомственный, так и общенациональный и даже интернациональный характер. Общими признаками WAN являются, во-первых, значительный масштаб сети (как по территориальному распределению, так и по числу узлов), а во-вторых, неоднородность сети (т. е. различный тип архитектуры и программного обеспечения узлов), что и определяет дополнительные сложности организации взаимодействия сетевых элементов. В частности, масштаб WAN требует решения проблем общей адресации сетевых узлов и маршрутизации передачи данных между ними.

Интернет - вычислительная сеть, объединяющая миллионы компьютеров по всему миру, фактически является конгломератом многих глобальных, региональных, университетских и учрежденческих сетей, а

также сетей коммерческих фирм (провайдеров), которые предоставляют доступ к Интернету индивидуальным клиентам. В Интернете нет центрального управляющего органа, а следовательно, выход из строя любого из существующих узлов или появление новых узлов не оказывают никакого влияния на общую работоспособность сети. Однако архитектура коммуникационной системы Интернет имеет вполне определенный иерархический характер. В этой иерархической архитектуре ограниченный набор дорогостоящих магистральных каналов с высокой пропускной способностью, составляющих так называемую опорную или базовую сеть, соединяет между собой сети со средней пропускной способностью, к которым, в свою очередь, подключаются отдельные организации со своими клиентами.

Фактически Internet состоит из множества локальных и глобальных сетей, принадлежащих различным компаниям и предприятиям, связанных между собой различными линиями связи. Как и во всякой другой сети в Internet существует 7 уровней взаимодействия между компьютерами: *физический, логический, сетевой, транспортный, уровень сеансов связи, представительский и прикладной уровень.*

Протоколы физического уровня определяют вид и характеристики линий связи между компьютерами. В Internet используются практически все известные в настоящее время способы связи от простого провода (витая пара) до волоконно-оптических линий связи (ВОЛС).

Для каждого типа линий связи разработан соответствующий ***протокол логического уровня***, занимающийся управлением передачей информации по каналу. К протоколам логического уровня для телефонных линий относятся протоколы SLIP (Serial Line Interface Protocol) и PPP (Point to Point Protocol).

Протоколы сетевого уровня отвечают за передачу данных между устройствами в разных сетях, то есть занимаются маршрутизацией пакетов в сети. К протоколам сетевого уровня принадлежат IP (Internet Protocol) и ARP (Address Resolution Protocol).

Протоколы транспортного уровня управляют передачей данных из одной программы в другую. К протоколам транспортного уровня принадлежат TCP (Transmission Control Protocol) и UDP (User Datagram Protocol).

Протоколы уровня сеансов связи отвечают за установку, поддержание и уничтожение соответствующих каналов. В Internet этим занимаются уже упомянутые TCP и UDP протоколы, а также протокол UUCP (Unix to Unix Copy Protocol).

Протоколы представительского уровня занимаются обслуживанием прикладных программ. К программам представительского уровня принадлежат программы, запускаемые, к примеру, на Unix-сервере, для предоставления различных услуг абонентам. К таким программам относятся: telnet-сервер, FTP-сервер, Gopher-сервер, NFS-сервер, NNTP (Net News Transfer Protocol), SMTP (Simple Mail Transfer Protocol), POP2 и POP3 (Post Office Protocol) и т.д.

К протоколам прикладного уровня относятся сетевые услуги и программы их предоставления.

Локальные вычислительные сети (ЛВС) или LAN (Local Area Network), обеспечивая взаимодействие небольшого количества однородных компьютеров на небольшой территории, имеют по сравнению с WAN менее развитую архитектуру и используют более простые методы управления взаимодействием узлов сети. При этом небольшие расстояния между узлами сети и простота управления системой связи позволяют обеспечивать в LAN более высокие скорости передачи данных (рис.3.1.1).

Основные элементы локальной сети

Сервер – компьютер, "руководящий обслуживанием" в сети с помощью своих устройств, программ и данных, предоставляющий другим компьютерам (рабочим станциям сети, клиентам) услуги по связи, получению, пересылке и обработке информации, а также совместно используемые ресурсы.

Рис. 3.1.1. Организация локальной сети

Строго говоря, сервером называется программа, устанавливаемая на компьютер для обслуживания совместной работы в сети других компьютеров. Но поскольку через подобный компьютер "протекает" большое количество информации, его аппаратную часть стараются сделать более мощной. Увеличивают объемы оперативной и дисковой памяти, применяют более быстродействующие процессоры, устанавливают либо несколько обычных сетевых карг, либо сетевые устройства: коммутаторы (свитчи), маршрутизаторы (роутеры). По этой причине сервером называют и компьютер, "руководящий обслуживанием" в сети. Обычно сервер работает круглосуточно для обеспечения бесперебойного доступа к размещенной на нем информации.

Рабочая станция, или хост (host), – компьютер, подключенный к сети и имеющий в сети собственный адрес. Это может быть как сервер, так и клиентский компьютер.

Клиент – компьютер в локальной сети, на котором пользователь запускает прикладные программы и с которого обращается к серверу за обеспечением связи с другими компьютерами и доступом к сетевым ресурсам (файлам, программам и устройствам). В отличие от сервера клиент хотя и подключен физически к сети, в отдельные моменты времени может быть логически (программно) отключен от нее. Еще одно отличие – у клиента в разные моменты времени может быть как постоянный, так и

разный (меняющийся в каждом сеансе работы в сети) адрес.

Кроме основных действующих лиц (клиентов, серверов), в сети имеется много других служебных устройств, с которыми пользователь непосредственно не работает, но от которых серьезно зависит и скорость работы в сети, и ее безопасность. Поэтому пользователь должен знать, какие устройства можно установить самостоятельно для усиления защищенности своего компьютера или сегмента сети, если они отсутствуют на участке: компьютер пользователя – сервер провайдера.

Повторитель (repeater) – устройство в сети, позволяющее восстановить амплитуду и мощность предаваемого сигнала, которые уменьшаются вследствие наличия потерь в линиях связи.

Концентратор (concentrator), или хаб (hub), – многовходовый (или многопортовый) повторитель, позволяющий обслуживать сразу несколько компьютеров в сети.

Мост (bridge) – программное или аппаратное средство для преобразования информации при обмене ею между однотипными сетями или их частями (логическими сегментами).

Коммутатор, или свитч (switch, switching hub), – коммуникационное устройство, в котором возможна параллельная независимая обработка информации, поступающей на разные порты (входы). Это отличает его от моста, где информация, поступающая с разных портов, обрабатывается друг за другом (последовательно).

Маршрутизатор (router) – комплекс программных и аппаратных средств, обеспечивающих в сети передачу по назначению (по заданному маршруту) пакетов данных и разделяющий информационные потоки отдельных частей сети друг от друга.

Шлюз (gateway) – устройство для соединения разнотипных сетей, работающих с отличающимся сетевым программным обеспечением и по разным протоколам.

Термин **internet** (со строчной буквы) обозначает локальную или региональную сетевую среду, объединенную с помощью средств маршрутизации, которые управляют пересылкой данных на основе общего пространства логических адресов узлов, т. е. обеспечение основных сетевых сервисов Интернета в пределах локальной или региональной сети.

Термин **intranet** обозначает изолированное пределами одной организации обеспечение сетевого доступа к общим данным при поддержке их разделения между отдельными подразделениями. Часто под intranet подразумевается обеспечение основных сетевых сервисов Интернета в пределах корпоративной ЛВС.

Термин **extranet** обозначает сетевое объединение нескольких организаций, обеспечивающее прямой доступ к приложениям каждой из сторон. Первоначально такое объединение осуществлялось за счет выделенных сетевых соединений. В настоящее время прямые выделенные соединения вытесняются виртуальными частными сетями VPN (Virtual Private Networks). По мере развития в Интернете средств ведения электронной коммерции и стандартов шифрования данных необходимость использования выделенных соединений, по всей видимости, полностью исчезнет.

Конфигурация локальной сети называется ***топологией***. Под топологией (компоновкой, конфигурацией, структурой) компьютерной сети обычно понимается физическое расположение компьютеров сети друг относительно друга и способ соединения их линиями связи. Важно отметить, что понятие топологии относится, прежде всего, к локальным сетям, в которых структуру связей можно легко проследить. В глобальных сетях структура связей обычно скрыта от пользователей и не слишком важна, так как каждый сеанс связи может производиться по собственному пути.

Существует два основных типа топологий (рис. 3.1.2):

- физическая
- логическая

Логическая топология описывает правила взаимодействия сетевых станций при передаче данных.

Физическая топология определяет способ соединения носителей данных.

Термин "топология сети" характеризует физическое расположение компьютеров, кабелей и других компонентов сети.

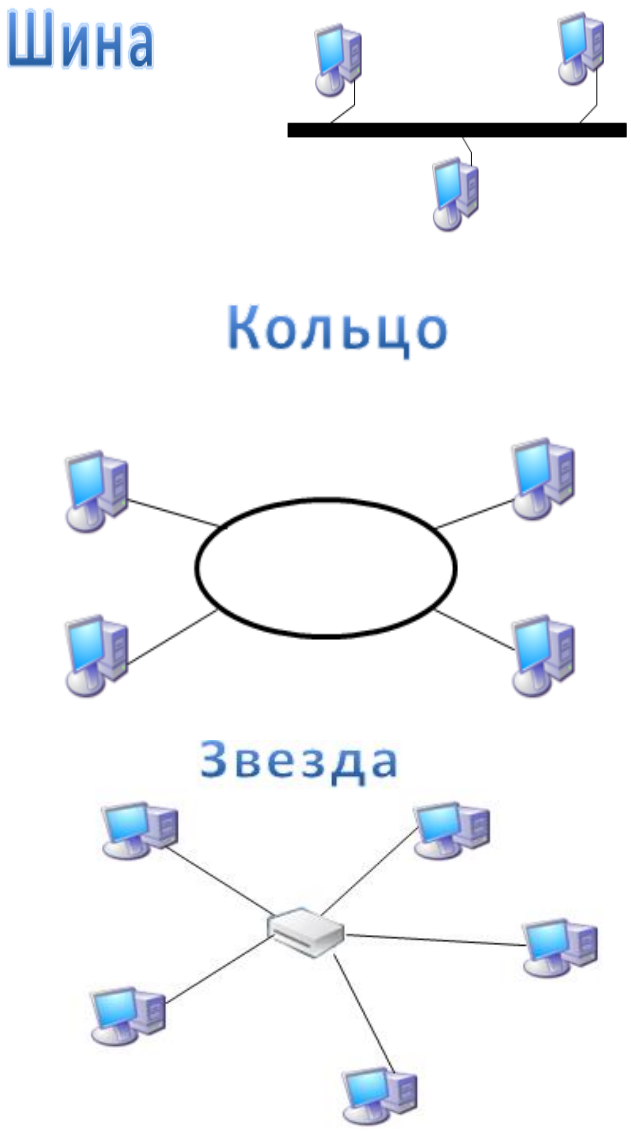


Рис. 3.1.2. Сетевая топология

Топология сети обуславливает ее характеристики.

Выбор той или иной топологии влияет на:

- состав необходимого сетевого оборудования

- характеристики сетевого оборудования
- возможности расширения сети
- способ управления сетью

Существует три базовые топологии, на основе которых строится большинство сетей.

Шина (bus) — все компьютеры параллельно подключаются к одной линии связи. Информация от каждого компьютера одновременно передается всем остальным компьютерам.

Звезда (star) — к одному центральному компьютеру присоединяются остальные периферийные компьютеры, причем каждый из них использует отдельную линию связи. Информация от периферийного компьютера передается только центральному компьютеру, от центрального — одному или нескольким периферийным.

Кольцо (ring) — компьютеры последовательно объединены в кольцо. Передача информации в кольце всегда производится только в одном направлении. Каждый из компьютеров передает информацию только одному компьютеру, следующему в цепочке за ним, а получает информацию только от предыдущего в цепочке компьютера.

Локальная сеть создаётся для рационального использования компьютерного оборудования и эффективной работы сотрудников.

Характерная особенность локальных сетей - наличие связывающего всех абонентов высокоскоростного канала связи для передачи информации в цифровом виде. Существуют проводные и беспроводные каналы (рис.3.1.3).

Каждый из них характеризуется определенными значениями существенных с точки зрения организации локальных сетей *параметров:*

- скорости передачи данных;
- максимальной длины линии;
- помехозащищенности;
- механической прочности;
- удобства и простоты монтажа, стоимости.

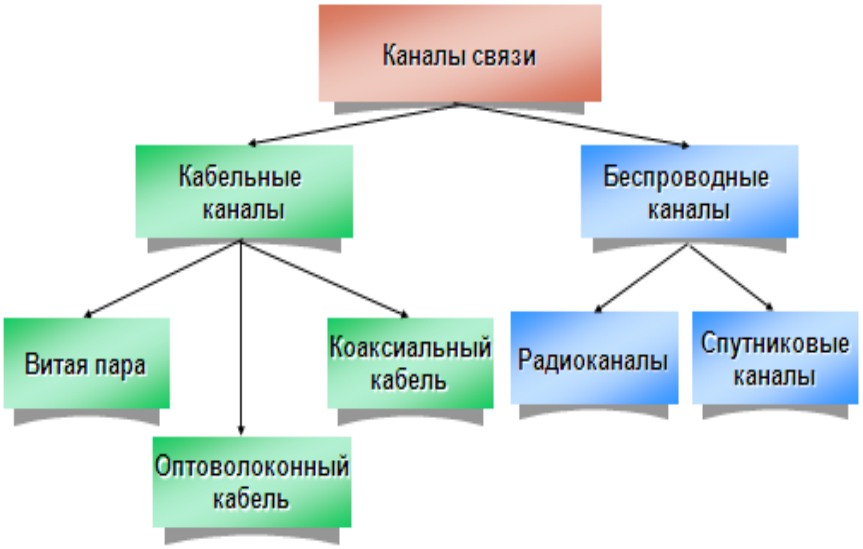


Рис. 3.1.3. Каналы связи

Существуют проводные и беспроводные каналы. В настоящее время обычно применяют четыре типа *сетевых кабелей:*

- коаксиальный кабель;
- незащищенная витая пара;
- защищенная витая пара;
- волоконно-оптический кабель.

Первые три типа кабелей передают электрический сигнал по медным проводникам. Волоконно-оптические кабели передают свет по стеклянному волокну.

Беспроводная связь на радиоволнах СВЧ диапазона может использоваться для организации сетей в пределах больших помещений типа ангаров или павильонов, там, где использование обычных линий связи затруднено или нецелесообразно. Для обеспечения согласованной работы в сетях передачи данных используются различные коммуникационные *протоколы* передачи данных – наборы правил, которых должны придерживаться передающая и принимающая стороны для согласованного обмена данными.

Протоколы – это наборы правил и процедур, регулирующих порядок осуществления некоторой связи. Протоколы – это правила и технические процедуры, позволяющие нескольким компьютерам при объединении в сеть общаться друг с другом.

Существует множество протоколов. Протоколы работают на разных уровнях модели взаимодействия открытых систем OSI/ISO. Среди множества протоколов наиболее распространены следующие:

- NetBEUI;
- XNS;
- IPX/SPX и NWLmk;
- Набор протоколов OSI.

Городские (региональные) сети (или сети мегаполисов) - Metropolitan Area Networks (MAN) - являются менее распространенным типом сетей. Эти сети появились сравнительно недавно. Они предназначены для обслуживания территории крупного города - мегаполиса. В то время как локальные сети наилучшим образом подходят для разделения ресурсов на коротких расстояниях, а глобальные сети обеспечивают работу на больших расстояниях, но с ограниченной скоростью и небогатым набором услуг, сети мегаполисов занимают некоторое промежуточное положение. Они используют цифровые магистральные линии связи, часто оптоволоконные, со скоростями от 45 Мбит/с, и предназначены для связи локальных сетей в масштабах города и соединения локальных сетей с глобальными.

Облачные технологии: базовые понятия и классификации

Сегодня трудно себе представить компьютер без Интернета. Действительно, компьютер стал устройством, ориентированным преимущественно на работу с Интернетом. Получение информации по любым вопросам, заказ товаров в интернет-магазинах, покупка билетов на поезда и самолеты, запись к врачу, просмотр фильмов online, прослушивание музыки и много другое - уже неотъемлемая часть нашей жизни. В общем-то многое, что мы делаем

сегодня в том самом Интернете, связано с понятием «облачные технологии» или «облако».

Облачные технологии (cloud technologies) - это технологии распределённой обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как интернет-сервис.

Проще говоря, облачные технологии - это такое технологическое решение, которое предполагает, что хранение и использование информации, программного обеспечения и различных сервисов не предусматривает задействования компьютерных жестких дисков. Эти диски используются только для начальной установки клиентского программного обеспечения (преимущественно операционной системы) с целью доступа к облачным сервисам.

В различных источниках даются определения: ***облачные вычисления*** (англ. cloud computing) - это «модель обеспечения удобного сетевого доступа по требованию к некоторому общему фонду конфигурируемых вычислительных ресурсов (например, сетям передачи данных, серверам, устройствам хранения данных, приложениям и сервисам - как вместе, так и по отдельности), которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами или обращениями к провайдеру».

Отмечается, что использование облачных вычислений может «значительно уменьшить расходы на инфраструктуру информационных технологий (в краткосрочном и среднесрочном планах) и гибко реагировать на изменения вычислительных потребностей, используя свойства вычислительной эластичности (англ. elastic computing) облачных услуг».

Собственно, вся разница между традиционными компьютерными технологиями и облачными технологиями (вычислениями) заключается исключительно в методе хранения и обработке данных. Если все операции происходят на Вашем компьютере (с использованием его мощностей), то это

не «облако», а если процесс происходит на сервере в сети, то это именно то, что принято называть «облачными технологиями».

Другими словами, облачные технологии - это различные аппаратные, программные средства, методологии и инструменты, которые предоставляются пользователю, как интернет-сервисы, для реализации своих целей, задач, проектов.

Графически понятие «Облачные технологии» / «облачные сервисы» / «облачные вычисления» изображают в виде облачка (рис. 3.1.4.).

Рис 3.1.4. Графическое представление «облака»

Видимо, более понятной их структура будет в виде пирамиды (рис. 3.1.5).

Рис.3.1.5. Графическое представление «облака» в виде пирамиды

Основанием пирамиды является инфраструктура, под которой понимается набор физических устройств: серверы, жесткие диски, линии связи и т. д.

Над ней выстраивается «платформа» - набор услуг, а на самом верху - программное обеспечение, доступное по запросу пользователей.

Можно предположить, что в ближайшем будущем компьютеры будут представлять собой один лишь экран с микропроцессором, а все расчеты и мощности будут расположены и производится удаленно, на далеких серверах, а именно в упомянутом неоднократно облаке.

Классификация облачных технологий

В настоящее время облачные сервисы подразделяют по следующим направлениями:

• Программное обеспечение как услуга (Software as a Service, сокращённо SaaS) - бизнес-модель предоставления или продажи программного обеспечения, при которой владелец (поставщик) ПО предоставляет доступ к нему пользователям (заказчикам) через Интернет. Примерами такого ПО являются MS Office 365, LearningApps.org, Zarafa и др.

• Оборудование (вычислительные мощности) как услуга (Hardware as a Service, сокращённо HaaS) - предоставление вычислительных ресурсов оборудования (его процессорного времени, места под хранения данных и т. д.) в виде сервисов с использованием технологий виртуализации. Сервисы обычно предлагаются как эквивалент реальным вычислительным системам, таким как серверы, суперкомпьютеры и др. Над программной реализацией этой идеи полностью или частично работают проекты Google, Yandex, OpenVZ, FreeVPS, Linux- VServer, Apache Hama, GlusterFS Open Source Project, а также Moose File System (MooseFS) и др., а предоставляет такой сервис на базе OpenSource решений компания Linode и некоторые другие.

• Коммуникация как Сервис (Communications as a Service, сокр. CaaS) - построенное в облаке коммуникационное решение, которое обеспечивает передачу речевого сигнала по сети Интернет или по любым другим IP-сетям (VoIP), обмен мгновенными сообщениями (IM), видеоконференции. Модель CaaS позволяет клиентам выборочно разворачивать средства коммуникаций и услуг на основании оплаты услуг в срок для используемых сервисов. Как

правило, многие из этих сервисов имеют бесплатные версии, но работающие с ограниченным количеством респондентов (например, ICQ, Skype, ooVoo). С этим же направлением тесно связаны такие FOSS-проекты, как Ekiga, iLBC, Speex.

• Мониторинг как Сервис (Monitoring-as-a-Service, сокращённо MaaS) является обслуживаемым в облаке программным обеспечением для мониторинга и обеспечения безопасности. Такими Open Source-решениями на сегодняшний день являются Ganglia, Zabbix, Hyperic HQ. Сюда же с некоторыми оговорками можно отнести и Nagios.

• Инфраструктура как услуга (Infrastructure as a Service, сокращённо IaaS) - это предоставление компьютерной инфраструктуры (как правило, в форме виртуализации) как услуги на основе концепции облачных вычислений. По сути, IaaS является комбинацией SaaS, HaaS, так как она включает в себя и то и другое, причем обычно во множественном числе, а также CaaS и иногда MaaS с целью объедения и мониторинга всей системы, и поэтому используется в основном предприятиями.

• Платформа как услуга (Platform as a Service, сокр. PaaS) - предоставление программной платформы и инструментов с определенными характеристиками, необходимых для разработки, тестирования, развертывания, поддержки различных приложений. Сюда же входят и готовые к использованию облачные сервисы, которые вместе образуют программную платформу. Яркими примерами из мира Open Source в настоящее время являются Xen Cloud Platform, Cloud Foundry, Apache Hadoop, Apache Hive и др.

• Компьютер (виртуальный рабочий стол) как услуга (Desktop as a Service, сокращённо DaaS) - предоставление виртуального компьютера, который каждый пользователь может индивидуально настраивать под свои задачи. Таким образом, пользователь, приходя на работу, просто вводит свои данные (обычно логин и пароль) и может работать, используя вычислительные мощности стороннего сервера, а не своего ПК.

• Рабочее окружение как услуга (Workspace as a Service, сокращённо WaaS) - предоставление комплекта SaaS, предназначенного для создания рабочего окружения. В отличие от DaaS в этом случае пользователь получает доступ только к ПО, в то время как все вычисления происходят непосредственно на его машине. По сути, данная категория является гибридом SaaS и PaaS, так как в отличие от последней является платформой, направленной не на разработку и тестирование ПО, а на офисную работу, но при этом в реализации не использует технологий виртуализации.

• Все как услуга (Everything as a service, сокращённо EaaS) - концептуальная модель, включающая в себя элементы всех перечисленных решений. На данный момент полной её реализации не существует - она по сути является идеалом для крупных облачных компаний, таких как Google и Microsoft. Классификация моделей облачных вычислений по группам пользователей представлена на рис.3.1.6.

Рис.3.1.6. Модель работы с облаками для разных групп пользователей

Приведенные классификации универсальны и рассматривают все направления использования облачных технологий.

Сегодня образовательные облачные сервисы открывают такие возможности, как создание виртуальных лабораторий в среде Интернет, проведение интернет-конференций и вебинаров, управление различными процессами виртуального пространства образовательной организации.

Современное информационное образовательное пространство образовательной организации анализируется в контексте электронного отражения в глобальной сети Интернет различных сторон ее деятельности.

Однако облачные технологии могут стать не только основой дистанционного и поддержкой реального образования. Образовательная организация представляет собой механизм с отлаженными алгоритмами взаимодействия: образовательный процесс тесно переплетен с процессами обеспечения бухгалтерского учета, учета персонала, договорными отношениями.

Схематично сферы применения облачных технологий в образовании представлены на рис.3.1.7.

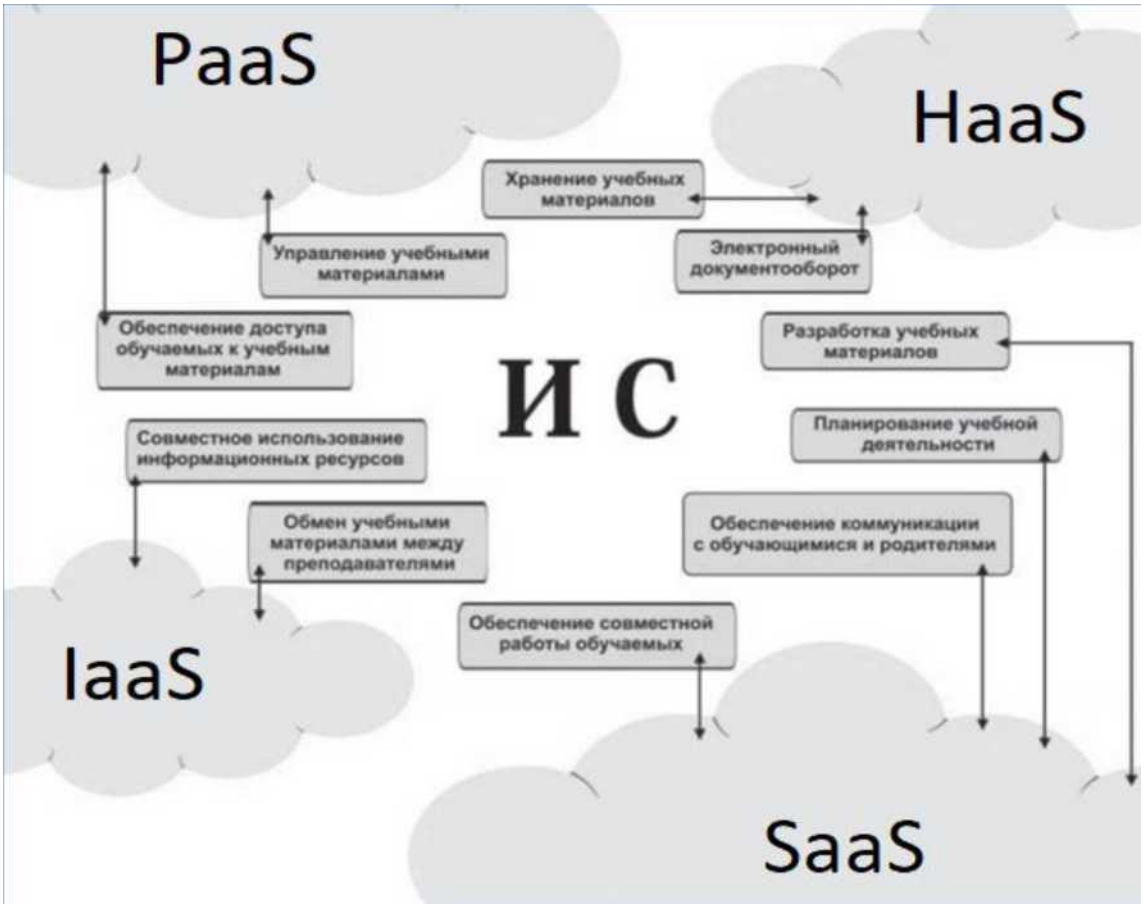


Рис.3.1.7. Схема интеграции облачных сервисов и образовательных информационных систем

В деятельности образовательных организаций используются следующие модели обслуживания: HaaS, PaaS, IaaS и SaaS. Перечисленные модели

позволяют использовать необходимое для создания учебных материалов или организации учебного процесса программное обеспечение на основе облачной парадигмы. Облачные сервисы, поддерживающие, например, модель HaaS, находят повсеместное применение в учебном процессе. Они предоставляют возможность разместить на виртуальном диске учебные и методические материалы, ссылки на полезные электронные ресурсы, домашние или контрольные задания, журналы посещаемости и успеваемости, аудио- и видео-ресурсы и открыть к ним доступ некоторой группе пользователей.

Достоинства и недостатки облачных технологий

Облачные технологии сегодня являются неотъемлемой частью современного мира. Эти технологии активно используются и системе образования. Облачные технологии предлагают учебным заведениям место для хранения данных, современное программное обеспечение и широкий спектр готовых образовательных ресурсов. Как любые технологии, облачные технологии имеют как свои достоинства, так и недостатки. Достоинства отчасти перечислены выше, но подчеркнем их еще. При использовании облачных технологий:

- не требуются мощные компьютеры, что снижает цену на ПК, поскольку используются ресурсы серверов;
- не нужно самостоятельно устанавливать и настраивать ПО, так как для доступа к облачным сервисам достаточно обычного веб-браузера;
- экономится дисковое пространство ПК;
- теряется смысл «использования пиратского ПО»;
- предоставляется достаточно большое пространство для хранения данных, что решает вопрос переноса данных с одного компьютера на другой (его просто не требуется);
- появляется возможность совместной работы в рамках одного документа;
- становится реальной организация элементов дистанционного обучения;

• экономия средств на оплату технических специалистов.

Разумеется, у облачных технологий есть и недостатки, к которым можно отнести:

• зависимость от подключения к сети Интернет. В случае, если Интернет не доступен, становится недоступным и необходимый ресурс (если его копии нет на жестком диске компьютера);

• из-за вопросов безопасности не все данные можно доверить стороннему провайдеру, тем более, не только для хранения, но и для обработки;

• далеко не каждое облачное приложение позволяет сохранить полученные результаты в удобном для пользователя виде на нужный носитель данных;

• есть риск, что провайдер онлайновых сервисов однажды не сделает резервную копию данных, и они будут утеряны в результате аварии на сервере.

Уже сегодня облачные технологии и предоставляемые ими сервисы фактически являются той основой, на которой базируется современное образование.

Вопросы для самоконтроля

1. Компьютерные сети
2. Структура и архитектура компьютерных сетей
3. Топология сетей
4. Организация локальной и региональной сетей
5. Возможности глобальной сети
6. Кабельные и беспроводные сети.
7. Облачные технологии
8. Классификация облачных технологий

3.2. Гипертекстовые, мультимедийные информационные технологии.

Основные модули

- Гипертекстовые технологии.
- Программные средства разработки Web-страниц
- Технология создания веб страниц.
- Мультимедийные технологии.
- Свойства мультимедийных ресурсов
- Мультимедиа и интерактивность

Гипертекстовая технология - это технология преобразования текста из линейной формы в иерархическую форму.

Использование гипертекстовой технологии (по сравнению с представлением информации в обычной книге) позволяет кардинально изменить способ просмотра и способ восприятия информации. Так, читая текст в книге, мы просматриваем его последовательно, страница за страницей. И если в процессе чтения, мы встретим термин, значение которого объяснялось раньше, то в этом случае нам придется листать страницы книги в обратном порядке до тех пор, пока не найдем нужное нам определение непонятного термина. Использование же гипертекстовой технологии позволяет значительно упростить работу с текстом и найти нужное определение за считанные секунды.

В настоящее время гипертекстовая технология широко используется для построения подсистем помощи пользователям при работе с диалоговыми компьютерными программами, а также для построения различных справочников, энциклопедий.

Если рассмотреть наиболее простую ***технологию построения гипертекста***, то она будет состоять из следующих *пяти основных шагов*:

Шаг 1 . Нужно разбить текст на отдельные главы/ темы.

Шаг 2 . Нужно представить себе некоторый основной путь чтения гипертекста и расставить, соответственно, поля-ссылки, ведущие читателя от темы к теме по этому основному пути.

Шаг 3 . Нужно выделить в тексте слова-ссылки, точнее, нужно найти ситуации (моменты) в процессе чтения текста, когда пользователь может захотеть перейти от основного пути чтения текста к другим возможным путям чтения.

Шаг 4 . В результате шага 3 могут появиться слова-ссылки, для которых еще не написаны соответствующие главы/темы. Такие главы нужно дописать.

Шаг 5 . Нужно связать ссылки с существующими темами.

Гипертексты дают текстам два дополнительных смысловых пространства. В тексте выделяются особые поля-ссылки, которые могут "сразу" привести читателя к нужным главам/темам, рисункам, описаниям. Благодаря этому процесс чтения становится принципиально иным - гипертекст можно читать/просматривать многими различными путями, и читатель сам выбирает тот путь просмотра, который ему наиболее удобен.

Простота концепции гипертекста обуславливает и формальную простоту общепринятой, технологии создания гипертекстов. Имея простейшую систему построения гипертекстов, можно быстро собрать из нескольких текстовых фрагментов гипертекст и формально получить самостоятельную гипертекстовую информационную систему, программный продукт или подсистему подсказки.

Но в силу видимой простоты гипертекстовой технологии очень легко создать гипертекстовую информационную систему с низким качеством.

Гипертексты обладают определенной семантической (смысловой) сетевой структурой. При многочисленном просмотре, если гипертекст используется как учебник, эта структура будет сильно влиять на структуру знаний пользователя по изучаемому вопросу. Поэтому при построении гипертекстовых систем следует уделять внимание не только тому, как

разбить исходный текст на части, но и тому, насколько пользователю будет понятно, легко и удобно работать с этими частями текста.

Web-сайты

Слово "*сайт*" (**site**) буквально означает "место", "местоположение". Web-сайты называют еще "узлами", "узлами Всемирной паутины".

Web-страница - *файл, хранящийся на сервере и используемый Web-сервером для показа на браузере клиента.*

Web-сайт - *это набор документов, хранящийся на сервере, управляемый Web-сервером и имеющий имя - адрес URL.*

Серверы, как правило, имеют большой объем дискового пространства и оперативной памяти, высокое быстродействие и работают круглосуточно. Работу web-сервера обеспечивает *администратор web-сервера*. Web-серверы бывают разными. Не все web-серверы подходят для той или иной операционной системы.

Основное назначение web-сервера - это *выполнение запроса клиента на предоставление ему нужной страницы*. Вызываемая страница может существовать на сервере *физически*, либо генерироваться *динамически* в соответствии с информацией, передаваемой клиентом.

Программные средства разработки Web-страниц

Создание web-сайтов, их поддержка и развитие осуществляется с помощью специализированного ПО.

В следующей таблице представлены наиболее популярные программные средства, предназначенные для разработки Web-сайтов (таблица 3.2.1).

Таблица 3.2.1

Microsoft FrontPage	**WYSIWYG** -редактор. Недостаток: автоматически вырабатываемый html-код документа, созданного

	разработчиком в визуальном режиме
Macromedia Flash	Технология Flash становится очень популярной. Она позволяет создавать очень эффектные web-страницы, содержащие FLASH-объекты или исполняемые файлы, содержащие большое количество векторной графики, анимационные ролики. За счет применения векторной графики Flash-страницы быстрее загружаются на компьютеры клиента, чем традиционные (содержащие растровую графику) и одинаково воспринимаются на различных платформах: Windows, Macintosh, Solaris, Unix. Имеется возможность передачи данных из HTML-документа FLASH-объекту и наоборот, что позволяет создавать управляемые FLASH-объекты, а также делать более эффектными HTML-страницы (например, формы).
Macromedia Director	Лидер рынка мультимедийных средств. Объединяет графику, звук, анимацию, текст и видео для интерактивных информационных каналов, которые можно разместить как на web-страницах, так и на CD- или DVD-дисках. От технологии Flash отличается более *развитым встроенным языком программирования*.
Macromedia Dreamweaver	Профессиональное решение для web-дизайна и разработки web-сайтов. Имеет очень удобный, простой интерфейс (в стиле PageMaker/ Illustrator/PhotoShop). Автоматизирует работу над проектом. Создаваемый код почти не отличается от написанного программистом. Содержит встроенные средства работы с графикой. Позволяет непосредст*венно внутри пакета* создавать FLASH-анимации. Обеспечивает средства отладки JavaScript-сценариев для браузеров MS

	Internet Explorer и Netscape Navigator. Допускает расширение возможностей за счет дополнительных модулей. Библиотека дополнительных компонент (более 150) входят в комплект поставки при вводе кодов создает список значений тэгов и атрибутов в виде всплывающей подсказки.
Macromedia Fireworks	Профессиональное приложение для создания графических изображений и их размещения в Интернет. Позволяет обрабатывать изображения, полученные с помощью других графических редакторов, цифровые фотографии, отсканированные изображения. Позволяет создавать эффекты анимации, использовать динамические стили. Совместное использовании Dreamweaver и Fireworks сокращает время разработки за счет взаимной автоматизации повторяющихся действий.
Allaire HomeSite	Позволяет легко и быстро создавать эффектные web-сайты. Удобный понятный интерфейс, богатую палитру инструментов. Содержит средства контроля качества: проверку синтаксиса html-кода, верификацию ссылок
Macromedia Dream Weaver UltraDev	Первая визуальная среда, позволяющая быстро разрабатывать Web-приложения для доступа к серверным базам данных. БД могут поддерживаться на различных серверных платформах. Достаточно просто создаются системы электронной коммерции, такие как электронные витрины, системы регистрации клиентов. Продукт уже имеет награды как лучший в своем классе средств разработки.
Macromedia ColdFusion.	Объединение среды разработки Cold **Fusion Studio** и среды **DreamWeaver UltraDev**. Содержит мощные

UltraDev 4 Studio	инструменты визуальной разработки приложений для размещения на платформе **ColdFusion Server**, визуального представления серверного источника данных (набора записей, переменной, директория и пр.), средства отладки сценариев.
Adobe PhotoShop	Мировой стандарт обработки изображений как для печати, так и для web.

Размещение web-страниц в Интернете

После того как вы подготовите дизайн нового сайта и будут завершены работы по верстке HTML-документов, надо загрузить готовый проект на сервер. Это необходимо для того, чтобы вашу информацию смогли получить заинтересованные пользователи Сети.

В некоторых случаях сайты разрабатываются для того, чтобы их просматривали не через Интернет, а с жестких носителей информации (например, компакт-дисков). Обычно таким образом выполняются презентации или каталоги товаров, сверстанные в формате HTML Гипертекст позволяет использовать простые в применении и широко распространенные технологии для быстрой подготовки документов. Так как средства для работы с Интернетом и web-страницами в той или иной степени присутствуют на большинстве компьютеров, изготовленную таким способом презентацию можно будет просмотреть в любом месте, не ломая голову над тем, откуда взять нужную для ее просмотра программу.

Если вы готовите страницу для просмотра без подключения к Интернету, то вопроса взаимодействия с различными серверами и службами сети перед вами не стоит. Если же разрабатываемый сайт предназначен для размещения в Интернете, то вам необходимо выполнить ряд действий, которые позволят вам сделать сайт доступным широкому кругу посетителей.

Прежде всего вам надо выбрать место для размещения сайта. То есть вам надо найти место на диске компьютера, где вы сможете поместить нужные файлы. Разумеется, этот компьютер должен иметь соединение с Интернетом. Если в вашем распоряжении есть компьютер, постоянно подключенный к Всемирной Сети, то вы можете организовать сервер прямо у себя дома или в офисе. Для этого достаточно установить на таком компьютере специальную программу (сервер). Она возьмет на себя функции взаимодействия с пользователями, выдачи им нужной информации и предотвращения попыток получения доступа к сведениям, которые вы не пожелаете раскрывать. Этот способ имеет ряд недостатков. В первую очередь, далеко не всегда есть возможность обеспечить качественный канал связи между вашим компьютером сервером; и магистральной линией связи, обслуживающей Интернет. Если связь плохая (например, скорость передачи данных резко ограничена), работа с вашим сайтом может оказаться для пользователя довольно утомительным занятием. Кроме того, не всегда удобно размещать в квартире или офисе компьютер с источниками бесперебойного питания и обеспечивать ему соответствующее обслуживание.

Более распространенный способ размещения web-страниц в Интернете - аренда места на сервере компании, специализирующейся на предоставлении подобных услуг. Размещение на сервере сайтов других компаний и частных лиц называют хостингом (от английского слова host - хозяин). Такие компании предоставляют своим клиентам некоторый объем на жестком диске для размещения сайта, а также набор услуг, в который обычно входит доступ к файлам клиента по протоколу FTP (это позволяет легко копировать файлы на сервер, изменять их или удалять), обслуживание электронной почты, обработка команд программ, входящих в состав сайта и т. д. Вы можете подключаться к серверу компании через Интернет и осуществлять управление своим сайтом (загружать на него новые страницы, включать и выключать обслуживающие функции и т. д.).

Как правило, такие услуги предоставляются за определенную плату. Компании, предоставляющие вам **хостинг** (хостинг-провайдеры), обычно заводят у себя специальный счет, на который вы вносите оплату за поддержку сайта. С этого счета снимается абонентская плата. Кроме того, некоторые провайдеры могут взимать плату за дополнительные услуги, например, за превышение установленного обмена данными между вашим сайтом и компьютерами пользователей.

В некоторых случаях размещение страниц на сервере производится бесплатно. Существуют компании, которые предоставляют всем желающим место на своих серверах, не требуя при этом оплаты. Чаще всего такие серверы используются для размещения домашних страничек. Предоставляемого такими компаниями объема вполне хватает для размещения небольшого фотоархива и биографии. Часто встречаются и так называемые «гостевые книги» - страницы, на которых пользователь может не только ознакомиться с текстом, но и оставить запись от своего имени

Технология создания веб страниц.

Знакомство со средой WordPress.

WordPress - система управления содержимым сайта (CMS) с открытым исходным кодом.

«Wordpress» - это свободно распространяемая система программ, написанных на скриптовом языке PHP. В ней применяются и CSS-стили, позволяющие мгновенно менять внешний вид сайта, доступно множество готовых тем - наборов стилей для оформления текста и страницы в целом, плюс необходимые графические элементы дизайна, наборы скриптов и специальных дополнений — виджетов, помогающих создать на сайте меню, удобную систему рубрик, архивы записей, поиск по сайту и прочие дополнительные удобства. Темы весьма разнообразные, выглядят солидно и профессионально. Каждая тема состоит из нескольких файлов-шаблонов, которые разрешается редактировать для изменения оформления сайта или создания своих собственных тем.

Одной из главных особенностей «Wordpress» является структура организации базы данных. Гибкость и функциональность связей позволяют создавать и выводить на страницу материал любого вида с любыми параметрами.

Встроенная система «тегирования» создает дополнительные связи для материалов сайта, что при необходимости, позволяет оперировать всеми записями, соответствующими определенным условиям.

При выборе для создания сайта системы «Wordpress», мы можем сразу заметить следующие ее преимущества:

- простота в установке и настройке;
- наличие удобного, настраиваемого административного интерфейса;
- легкость при непосредственном создании сайта;
- в дальнейшем, простое управление сайтом и его редактирование;
- наличие простого и удобного консоля;
- поддержка «тем», позволяющих легко менять как внешний вид, так и способы вывода данных;
- наличие громадных библиотек «тем» и «плагинов»;
- наличие системы контроля безопасности сайта;
- наличие системы автосохранения набираемого в редакторе текста, для предотвращения потери информации из-за программного или аппаратного сбоя;
- наличие инструмента автоматического обновления до более свежей версии.

Создание сайта на примере пропедевтического курса

В разработке сайта, в качестве инструмента создания онлайн-ресурса, размещения файлов использовать конструктор создания сайта wixsite.com.

ШАГ 1: нажимаем вход, регистрируемся и вводим пароль (рис.3.2.1).

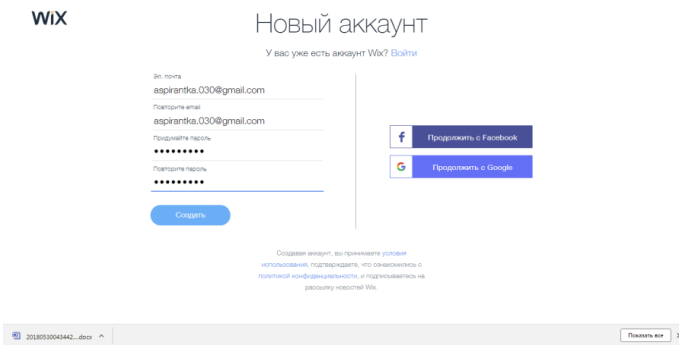


Рис.3.2.1. Регистрация на специализированном конструкторе сайтов

ШАГ 2: открываем вкладку TASK, очищаем страницу. Выбираем необходимый шаблон для оформления сайта (рис.3.2.2).

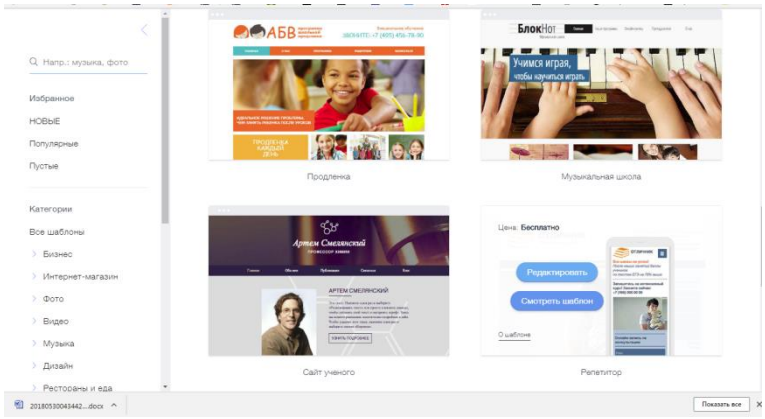


Рис.3.2.2. Выбор шаблона для оформления сайта

ШАГ 3: создаём необходимые категории пропедевтического курса «ПКП ПО» и загружаем файлы (рис.3.2.3-3.2.5).

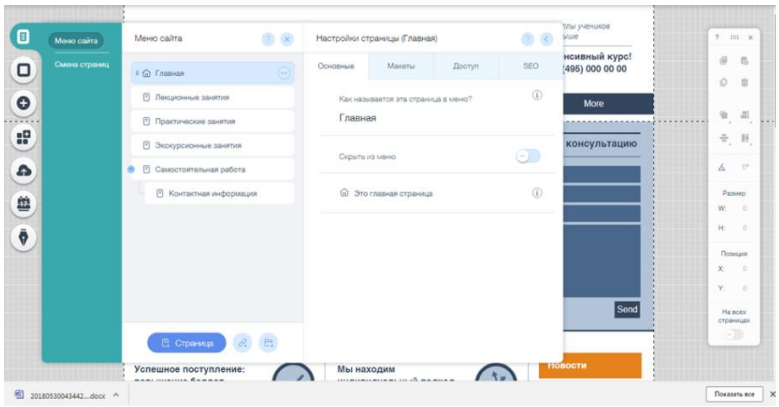


Рис.3.2.3. Выбор выпадающего меню сайта

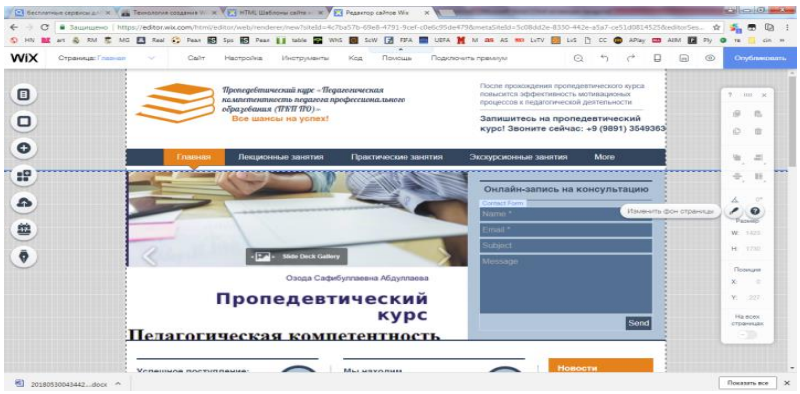


Рис.3.2.4. Создание иконок сайта

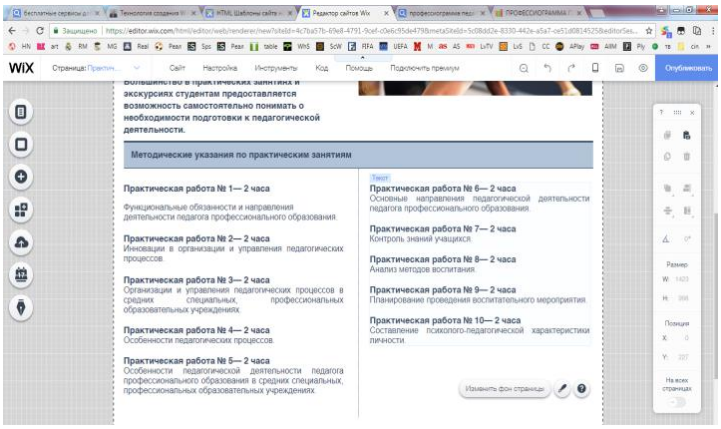


Рис.3.4.5. Загрузка и помещение файлов в базу сайта

ШАГ 4: Получаем ссылку на онлайн-ресурс https://aspirantka030.wixsite.com/propevcourse/prakticheskie-zanyatiya/ (рис.3.2.6).

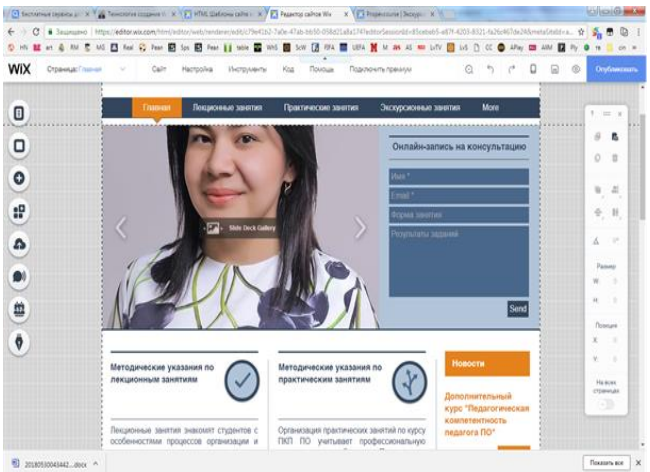


Рис.3.2.6. Онлайн-ресурс «Пропедевтический курс»

Понятие «мультимедиа»

Мультимедиа-технологии являются одним из многочисленных приемов представления информации, на которых следует остановиться более подробно. Очевидно, что рассмотрение видов и форм представления информации следует начинать с изучения собственно понятия информация.

Изучение информации, особенностей ее обработки, безусловно, должно начинаться с *методов представления информации*, поскольку именно спецификой методов определяется дальнейшая технология передачи и обработки информации, возможностей ее использования в традиционном и открытом образовании. *В связи с этим, вопросы, связанные с эффективным представлением информации, в том числе и учебного материала, являются одними из важнейших проблем обучения.* Особую значимость они приобрели в настоящий период, в связи с использованием информационных технологий в процессе дистанционного обучения и необходимостью представления учебной информации на экране компьютеров. Данная проблема осложняется еще и тем фактом, что в последние годы значительно увеличился объем информации по всему циклу учебных дисциплин, а время, отводимое на их изучение, не изменилось. Иначе говоря, увеличилась плотность потока учебной информации, изучение которой требуется для подготовки специалистов на всех уровнях системы образования.

Всю информацию по способу восприятия обучаемыми, можно разделить на три основные группы:

- Информация, воспринимаемая слуховым аппаратом человека, так называемая *звуковая информация*;
- Информация, воспринимаемая зрением человека, так называемая *зрительная или визуальная информация*, включающая текст и графические изображения-картинки.

• Информация, частично воспринимаемая сенсорной системой человека при работе с помощью специальных технических средств с видеороликами, телеобъектами и др. – *сенсорная или тактильная информация*.

В основе ***гипертекстового представления информации*** лежит идея расширения традиционного понятия текста, путем введения понятия нелинейного текста, в котором между выделенными текстовыми фрагментами устанавливаются перекрестные связи и определяются правила перехода от одного фрагмента текста к другому. При этом получается сеть, которая называется гипертекстом или нелинейным текстом.

Гипертекст не является высшей и наиболее универсальной стадией представления и организации информации, поскольку увязывает с учетом некоторой структуры информацию только одного типа - текстовую. Внедрение телекоммуникаций и повсеместное использование информационных технологий в образовании привели к созданию более прогрессивных информационных средств - систем гипермедиа.

Гиперсредой или гипермедиа называется гипертекст, в состав которого входит структурированная информация разных типов (текст, иллюстрации, звук, видео). Неслучайно, одной из основных сфер применения систем гипермедиа является открытое образование. Подобные средства играют огромную роль в процессе самообучения.

Представление разнотипной и, как правило, структурированной информации с использованием современных средств ИКТ стало возможным, благодаря появлению специализированной технологии ***мультимедиа.***

В широком смысле «мультимедиа» означает спектр информационных технологий, использующих различные программные и технические средства с целью наиболее эффективного воздействия на пользователя (ставшего одновременно и читателем, и слушателем, и зрителем).

Согласно наиболее распространенного определения *мультимедиа (мультимедиа средства)* представляет собой компьютерные средства создания, хранения, обработки и воспроизведения в оцифрованном виде информа-

ции разных типов: текста, рисунков, схем, таблиц, диаграмм, фотографий, видео- и аудио- фрагментов и т.п.

Технологии мультимедиа позволяют осмысленно и гармонично сочетать многие виды мультимедийной информации. Это позволяет с помощью компьютера представлять знания в различных формах, таких как:

- изображения, включая отсканированные фотографии, чертежи, карты и слайды;
- звукозаписи голоса, звуковые эффекты и музыка;
- видео, сложные видеоэффекты и анимационное имитирование;
- анимации и симуляции.

Свойства мультимедийных ресурсов (мультимедиа и интерактивность)

Интерактивность мультимедийных средств подразумевает широкий круг возможностей воздействия на процесс обучения и содержание учебных материалов со стороны пользователя, в числе которых:

- манипулирование экранными объектами;
- линейная навигация - скроллинг в рамках экрана;
- иерархическая навигация - выбор содержательных подразделов с помощью иерархически организованной системы меню;
- функция интерактивной справки, вызываемая специальными кнопками на панели навигации. Наиболее эффективна контекстно-зависимая справка;
- взаимодействие с пользователем, когда средство обладает возможностью ответа на запросы и действия пользователей;
- конструктивное взаимодействие, когда мультимедийное средство предоставляет возможность создания или конфигурирования экранных объектов;
- рефлективное взаимодействие, когда мультимедийное средство учитывает действия пользователя для последующего анализа (например, для того чтобы на основе этой информации рекомендовать учащемуся оптимальную

последовательность изучения материала), выбор между «экспертным» или «ознакомительным» вариантом изучения;

• симулятивная интерактивность в том случае, когда экранные объекты связаны друг с другом и взаимодействуют таким образом, что настройка этих объектов определяет их «поведение» (симулирующее реальное функционирование технических устройств, социальные процессы, и т.п.);

• неуглубленная контекстная интерактивность, благодаря которой учащийся вовлекается в различные виды деятельности, имеющие неявное дидактическое значение. Этот тип интерактивности используется во многочисленных развлекательно-обучающих мультимедийных программах и в различных мультимедиа-играх;

• углубленная контекстная интерактивность, сводимая к специфике функционирования систем виртуальной реальности, в которых пользователь погружается в симулируемый трехмерный мир.

Различают *три основных типа интерактивности*, используемых мультимедийными средствами обучения *реактивное, активное и двустороннее взаимодействие* (рис.3.2.7).

Рис.3.2.7. Особенности организации диалога человека и мультимедийного ресурса

Следует учитывать, что взаимодействие пользователя с любым мультимедийным средством не является диалогом в полном смысле этого слова. Согласно общепринятого определения *диалог* – это развитие темы, позиции,

точки зрения совместными усилиями двух и более людей, находящихся во взаимодействии и общении по поводу определенного или неизвестного в тех или иных деталях содержания.

Траектория этого совместного общение заранее не прогнозируема и задается смыслами и содержательными направлениями, которые порождаются в ходе самого диалога. В большинстве компьютерных программ заранее задаются те «ветви дерева», по которым движется процесс, инициируемый пользователями конкретных мультимедийных средств. Если пользователь попадет не на ту «ветвь», компьютер выдаст «реплику» о том, что пользователь ошибся и «забрел» не туда, куда предусмотрено логикой программы и что необходимо повторить попытку или начать с просмотра другой ветви, что совершенно не характерно межличностному общению людей.

Вопросы для самоконтроля

1. Гипертекстовые технологии.
2. Программные средства разработки Web-страниц
3. Технология создания веб страниц.
4. Мультимедийные технологии.
5. Свойства мультимедийных ресурсов.
6. Мультимедиа и интерактивность.

3.3. Криптографические методы защиты данных.

Основные модули

- Понятие безопасности информации.
- Политика информационной безопасности.
- Технические и программные средства защиты информации.
- Способы защиты информации.
- Компьютерные вирусы и их виды.
- Защита от компьютерных вирусов.

Информационной безопасностью называют комплекс организационных, технических и технологических мер по защите информации от неавторизованного доступа, разрушения, модификации, раскрытия и задержек в доступе.

Информационная безопасность дает гарантию того, что достигаются следующие цели:

- **конфиденциальность** информации (свойство информационных ресурсов, в том числе информации, связанное с тем, что они не станут доступными и не будут раскрыты для неуполномоченных лиц);
- **целостность** информации и связанных с ней процессов (неизменность информации в процессе ее передачи или хранения);
- **доступность** информации, когда она нужна (свойство информационных ресурсов, в том числе информации, определяющее возможность их получения и использования по требованию уполномоченных лиц);
- **учет** всех процессов, связанных с информацией.

Обеспечение **безопасности информации** складывается из трех составляющих:

- Конфиденциальности,
- Целостности,
- Доступности.

Точками приложения процесса защиты информации к информационной системе являются:

- аппаратное обеспечение,
- программное обеспечение
- обеспечение связи (коммуникации).

Сами процедуры(механизмы) защиты разделяются на

- защиту физического уровня,
- защиту персонала

- организационный уровень.

Угроза безопасности компьютерной системы - это потенциально возможное происшествие (преднамеренное или нет), которое может оказать нежелательное воздействие на саму систему, а также на информацию, хранящуюся в ней.

Политика безопасности - это комплекс мер и активных действий по управлению и совершенствованию систем и технологий безопасности, включая информационную безопасность.

Виды информационных угроз

Организационная защита

- *организация режима и охраны*.
- *организация работы с сотрудниками* (подбор и расстановка персонала, включая ознакомление с сотрудниками, их изучение, обучение правилам работы с конфиденциальной информацией, ознакомление с мерами ответственности за нарушение правил защиты информации и др.)
- *организация работы с документами* и документированной информацией (разработка, использование, учет, исполнение, возврат, хранение и уничтожение документов и носителей конфиденциальной информации)
- *организация использования технических средств* сбора, обработки, накопления и хранения конфиденциальной информации;
- *организация работы по анализу внутренних и внешних угроз* конфиденциальной информации и выработке мер по обеспечению ее защиты;
- *организация работы по проведению систематического контроля за работой персонала* с конфиденциальной информацией, порядком учета, хранения и уничтожения документов и технических носителей.

Технические средства защиты информации

Для защиты периметра информационной системы создаются:

- системы охранной и пожарной сигнализации;
- системы цифрового видео наблюдения;
- системы контроля и управления доступом (СКУД).

Защита информации от ее утечки техническими каналами связи обеспечивается следующими средствами и мероприятиями:

- использованием экранированного кабеля и прокладка проводов и кабелей в экранированных конструкциях;
- установкой на линиях связи высокочастотных фильтров;
- построение экранированных помещений («капсул»);
- использование экранированного оборудования;
- установка активных систем зашумления;
- создание контролируемых зон.

Аппаратные средства защиты информации

- Специальные регистры для хранения реквизитов защиты: паролей, идентифицирующих кодов, грифов или уровней секретности;
- Устройства измерения индивидуальных характеристик человека (голоса, отпечатков) с целью его идентификации;
- Схемы прерывания передачи информации в линии связи с целью периодической проверки адреса выдачи данных.
- Устройства для шифрования информации (криптографические методы).
- Системы бесперебойного питания:
  - Источники бесперебойного питания;
  - Резервирование нагрузки;
  - Генераторы напряжения.

Программные средства защиты информации

- Межсетевые экраны.
- Криптографические средства:
  - Шифрование;
  - Цифровая подпись.
- Системы резервного копирования.
- Системы аутентификации:
  - Пароль;
  - Ключ доступа (физический или электронный);
  - Сертификат;
  - Биометрия.
- Инструментальные средства анализа систем защиты:

– Мониторинговый программный продукт (рис 3.3.1.).

Рис 3.3.1. Виды информационных угроз

Компьютерные вирусы и их виды. Защита от компьютерных вирусов

Компьютерный вирус — это специально написанная, как правило, небольшая по размерам программа, которая может записывать (внедрять) свои копии (возможно, измененные) в компьютерные программы, расположенные в исполнимых файлах, системных областях дисков, драйверах, документах и т.д., причем эти копии сохраняют возможность к «размножению».

Процесс внедрения вирусом своей копии в другую программу (системную область диска и т.д.) называется заражением, а программа или объект, содержащий вирус — зараженным.

Компьютерные вирусы являются программами, которые могут «размножаться» и скрытно внедрять свои копии в файлы, загрузочные секторы дисков и документы.

Обязательным свойством компьютерного вируса является способность к размножению (самокопированию) и незаметному для пользователя внедрению в файлы, загрузочные секторы дисков и документы. Название «вирус» по отношению к компьютерным программам пришло из биологии именно по признаку способности к саморазмножению.

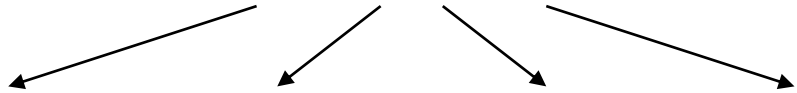
Классификация компьютерных вирусов

- по среде обитания;
- по операционным системам;
- по алгоритму работы;
- по деструктивным возможностям.

По операционным системам

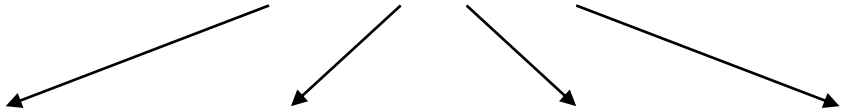
Для каждой операционной системы создаются свои вирусы, которые будут «работать» только в ней. Но существуют и универсальные вирусы, которые способны внедряться в различные операционные системы.

По среде обитания



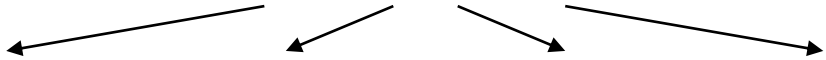
Файловые вирусы	***Загрузочные вирусы***	***Макро-вирусы***	***Сетевые вирусы***
Наносят вред файлам. Создают файл-двойник с именем оригинала.	внедряются в загрузочный сектор диска. Операционная система при этом загружается с ошибками и сбоями	«Портят» документы Word, Excel и других прикладных программ операционной системы Windows.	Распространяются по Internet через электронные письма или после посещения сомнительных сайтов.

По алгоритму работы



Резидентность	***Самошиф-рование и полиморфизм***	***Стелс–алгоритм***	***Нестандарт-ные приемы***
Вирусы, обладающие этим свойством действуют постоянно пока компьютер включен.	Вирусы-полиморфики изменяют свой код или тело программы, что их трудно обнаружить.	Вирусы-невидимки «прячутся» в оперативной памяти и антивирусная программа их не может обнаружить.	Принципиально новые методы воздействия вируса на компьютер.

По деструктивным возможностям



Безвредные	***Неопасные***	***Опасные***	***Очень опасные***
не наносят никакого вреда	наносят моральный ущерб пользователю	уничтожают информацию в файлах.	сбивают процесс загрузки ОС, после чего требуется ее переустановка; или «портят» винчестер, что его требуется форматировать

Вредоносные программы

- *Троянский конь* - это программа, содержащая в себе некоторую разрушающую функцию, которая активизируется при наступлении некоторого условия срабатывания. Обычно такие программы маскируются под какие-нибудь полезные утилиты. Виды деструктивных действий:
  - Уничтожение информации. (Конкретный выбор объектов и способов уничтожения зависит только от фантазии автора такой программы и возможностей ОС. Эта функция является общей для троянских коней и закладок).
  - Перехват и передача информации. (паролей, набираемых на клавиатуре).
  - Целенаправленное изменение программы.
- *Червями* называют вирусы, которые распространяются по глобальным сетям, поражая целые системы, а не отдельные программы. Это самый опасный вид вирусов, так как объектами нападения в этом случае становятся информационные системы государственного масштаба. С

появлением глобальной сети Internet этот вид нарушения безопасности представляет наибольшую угрозу, т.к. ему в любой момент может подвергнуться любой из компьютеров, подключенных к этой сети. Основная функция вирусов данного типа – взлом атакуемой системы, т.е. преодоление защиты с целью нарушения безопасности и целостности.

Для обнаружения, удаления и защиты от компьютерных вирусов разработано несколько видов специальных программ, которые позволяют обнаруживать и уничтожать вирусы. Такие программы называются антивирусными.

Виды антивирусных программ

- ***Детекторы*** позволяют обнаруживать файлы, заражённые одним из нескольких известных вирусов. Некоторые программы-детекторы также выполняют эвристический анализ файлов и системных областей дисков, что часто (но отнюдь не всегда) позволяет обнаруживать новые, не известные программе-детектору, вирусы.
- ***Фильтры*** - это резидентные программы, которые оповещают пользователя о всех попытках какой-либо программы записаться на диск, а уж тем более отформатировать его, а также о других подозрительных действиях.
- ***Программы-доктора*** *или* ***фаги*** не только находят зараженные вирусами файлы, но и «лечат» их, т.е. удаляют из файла тело программы-вируса, возвращая файлы в исходное состояние.
- ***Ревизоры*** запоминают сведения о состоянии файлов и системных областей дисков, а при последующих запусках – сравнивают их состояние исходным. При выявлении несоответствий об этом сообщается пользователю.

- ***Сторожа*** *или* ***фильтры*** располагаются резидентно в оперативной памяти компьютера и проверяют на наличие вирусов запускаемые файлы и вставляемые USB-накопители.
- ***Программы-вакцины*** *или* ***иммунизаторы*** модифицируют программы и диски таким образом, что это не отражается на работе программ, но тот вирус, от которого производится вакцинация, считает эти программы или диски уже заражёнными.

Недостатки антивирусных программ

- Ни одна из существующих антивирусных технологий не может обеспечить полной защиты от вирусов.
- Антивирусная программа забирает часть вычислительных ресурсов системы, нагружая центральный процессор и жёсткий диск. Особенно это может быть заметно на слабых компьютерах.
- Антивирусные программы могут видеть угрозу там, где её нет (ложные срабатывания).
- Антивирусные программы загружают обновления из Интернета, тем самым расходуя трафик.
- Различные методы шифрования и упаковки вредоносных программ делают даже известные вирусы не обнаруживаемыми антивирусным программным обеспечением. Для обнаружения этих «замаскированных» вирусов требуется мощный механизм распаковки, который может дешифровать файлы перед их проверкой. Однако во многих антивирусных программах эта возможность отсутствует и, в связи с этим, часто невозможно обнаружить зашифрованные вирусы.

Установка, обновление базы ESET NOD32, проверка памяти с ее помощью, очистка вирусов.

Для установки антивируса ESET NOD32 необходимо нажать на exe файл. Для соглашения лицензионного соглашения необходимо нажать на галочку на «Я принимаю условия лицензионного соглашения» и нажать на кнопку “Далее” (рис.3.3.2-3.3.5).

В данному диалоговом окне необходимо ввести “Имя пользователя” и “Пароль” для автоматического обновления, или нажать на “Установить параметры обновления позже” затем нажать на кнопку “Далее”. В результате установится программа и появится ее логотип.

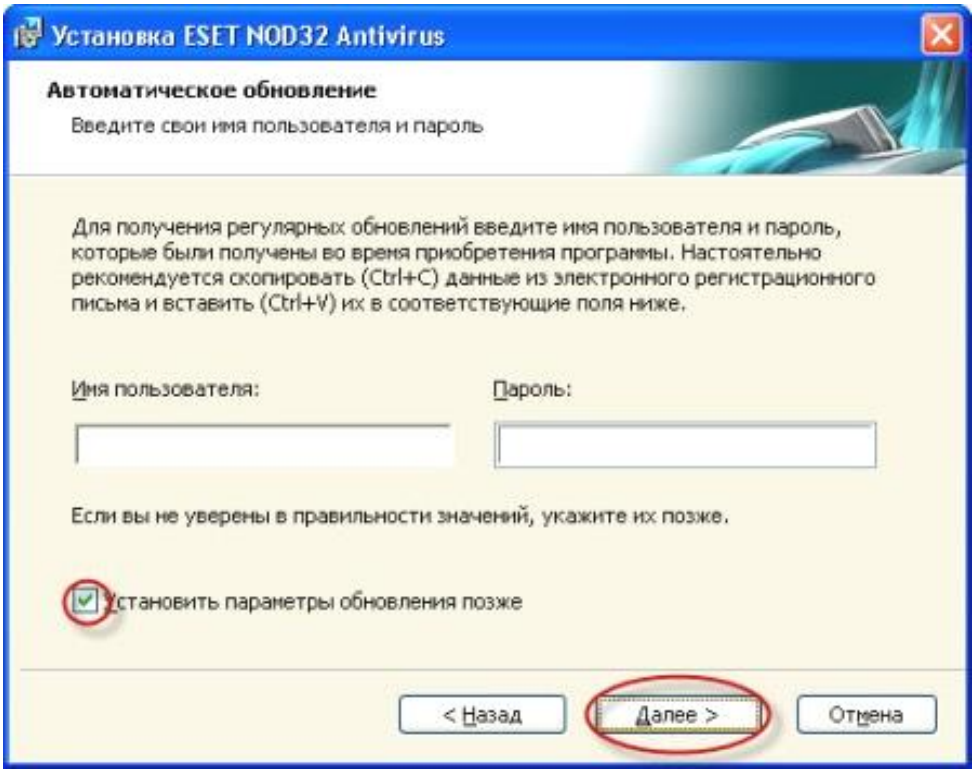


Рис.3.3.2. Установка ESET NOD32

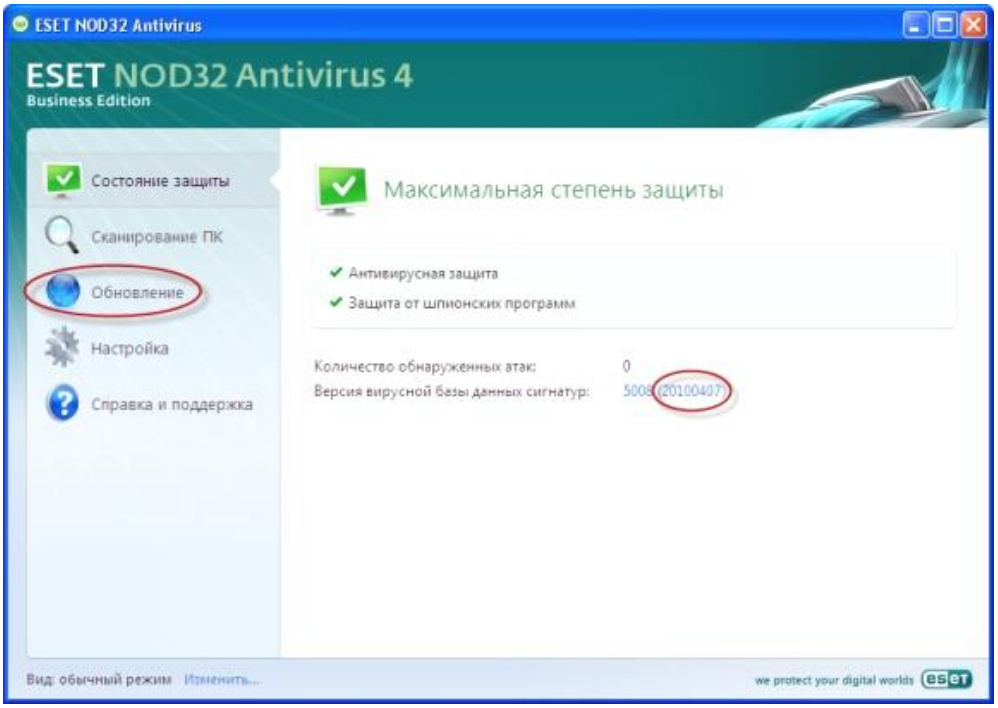


Рис.3.3.3. Установка ESET NOD32

Для обновления базы необходимо на рабочем столе нажать на логотип ESET NOD32 или выполнить действия «Пуск»-«Все программы»-«ESET»-«ESET NOD32 Antivirus»-«ESET NOD32 Antivirus».

В открывшемся окне в разделе “Обновлении” появятся данные о последнем обновлении программы.

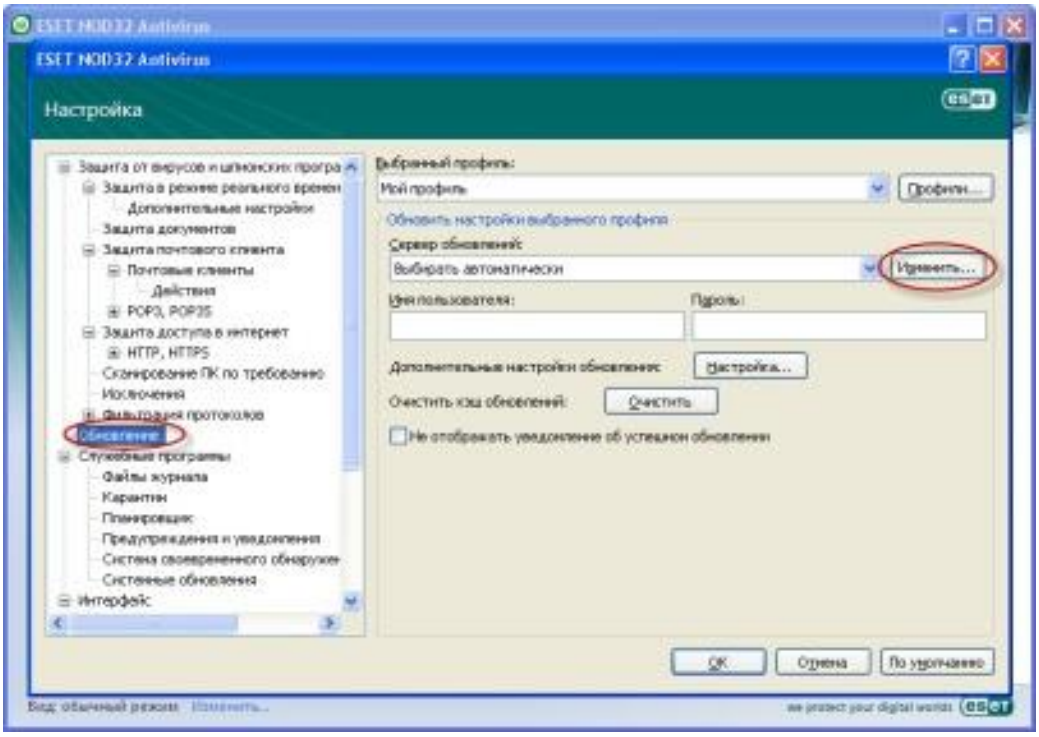


Рис.3.3.4. Установка ESET NOD32

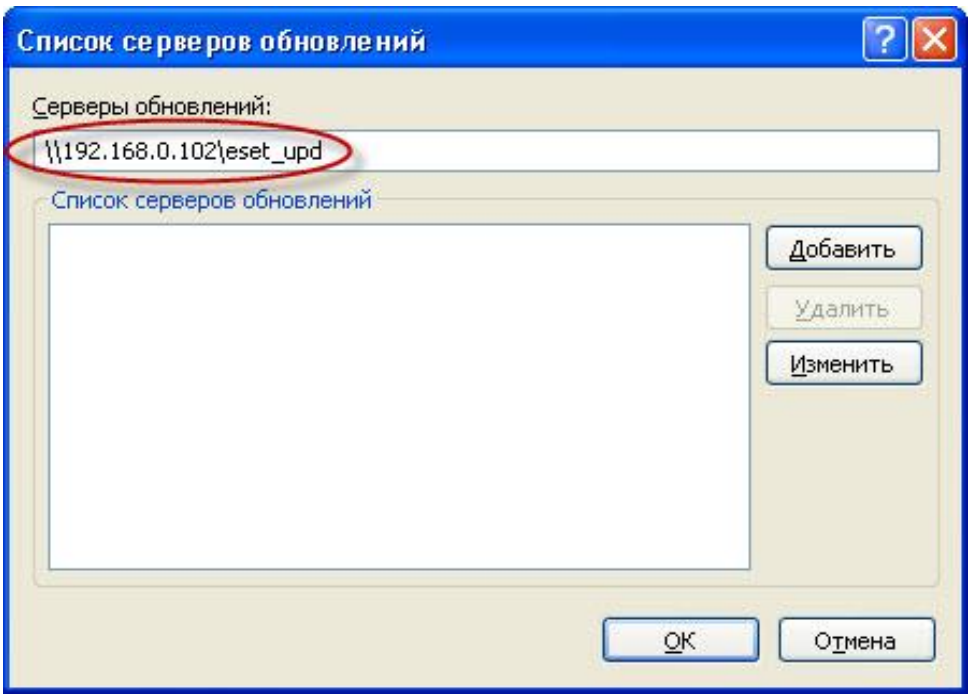


Рис.3.3.5. Установка ESET NOD32

Нажмите на кнопку F5. В открывшемся окне необходимо указать расположение базы для антивируса. Для этого необходимо нажать на кнопку «Изменить».

Для проверки (fleshki, жесткого диска, чипов) необходимо нажать на правую кнопку мыши и выбрать «Расширенные функции/ очистка файлов».

В данному диалоговом окне необходимо ввести “Имя пользователя” и “Пароль” для автоматического обновления, или нажать на “Установить параметры обновления позже” затем нажать на кнопку “Далее”. В результате установится программа и появится ее логотип.

Вопросы для самоконтроля

1. Понятие безопасности информации.
2. Политика информационной безопасности.
3. Криптографические методы защиты информации.
4. Технические и программные средства защиты информации.
5. Способы защиты информации.

3.4. Защита информации в компьютерных сетях. Средства информационной безопасности

Основные модули

- Основные понятия безопасности.
- Планирование безопасности сети и данных.
- Средства обеспечения безопасности.
- Базовые технологии безопасности.
- Аутентификация (Authentication).
- Идентификация субъектов и объектов доступа.
- Авторизация (Authorization).
- Аудит (Auditing).

• Схемы образования защищенного канала.

• Средства безопасности, предоставляемые операционными системами.

Сетевая безопасность охватывает множество мер и должна рассматриваться как часть общей политики, проводимой организацией (предприятием, компанией, фирмой) по информационной безопасности. В обеспечении безопасности сети занято много служб и используются различные средства. По сетевой безопасности написано огромное количество книг и статей, затрагивающих широкий.

Эффективность компьютерной сети во многом зависит от степени защищенности обрабатываемой и передаваемой информации. Степень защищенности информации от различного вида угроз при ее получении, обработке, хранении, передаче и использовании называют безопасностью информации.

Актуальность проблеме сетевой безопасности придает широкое использование компьютерных технологий во всех сферах жизни современного общества, а также переход от использования выделенных каналов к публичным сетям (Internet, Frame Relay), который наблюдается при построении корпоративных сетей.

Безопасная сеть (или безопасная связь) обладает свойствами:

• конфиденциальности (Confidentiality), т.е. защищает данные от несанкционированного доступа, предоставляя доступ к секретным данным только авторизованным пользователям, которым этот доступ разрешен;

• доступности (Availability), что означает обеспечение постоянного доступа к данным авторизованным пользователям. Безопасная связь характеризуется свойством аутентичности, т.е. способностью отправителя и получателя подтвердить свою личность: отправитель и получатель должны быть уверены в том, что каждый из них является тем, за кого он себя выдает;

• целостности (Integrity), гарантирующей сохранность данных, которая обеспечивается запретом для неавторизованных пользователей каким-либо

образом изменять, модифицировать, разрушать или создавать данные.

Политика безопасности, включающая в себя совокупность норм и правил, регламентирующих процесс обработки информации, формируется на этапе развертывания сети с учетом таких основополагающих принципов, как:

- комплексный подход к обеспечению безопасности, начиная с организационно-административных запретов и заканчивая встроенными средствами сетевой защиты;
- предоставление каждому сотруднику предприятия (пользователю компьютеров, информационной системы, сети) того минимального уровня привилегий на доступ к данным, который необходим ему для выполнения своих должностных обязанностей;
- принцип баланса возможного ущерба от реализации угрозы и затрат на ее предотвращение. Например, в некоторых случаях можно отказаться от дорогостоящих аппаратных средств защиты, ужесточив административные меры.

Основная задача политики безопасности состоит в защите от несанкционированного доступа к ресурсам информационной системы. Политика безопасности является эффективным средством, заставляющим всех пользователей корпоративной сети следовать раз и навсегда установленным правилам безопасности. Ее реализация начинается с выявления уязвимых компонентов и угроз, и принятия соответствующих контрмер.

Уязвимым является такой компонент, некорректное использование или сбой которого может поставить под угрозу безопасность всей сети. К уязвимым компонентам относят пользователей сети, которые могут нанести вред сознательно, случайно или в силу отсутствия опыта.

Если информация нерегулярно резервируется, перед всей корпоративной сетью возникает вполне реальная угроза потери данных в результате умышленного или случайного повреждения основного

накопителя.

Угроза - это потенциальная попытка использования недостатков уязвимого компонента для нанесения вреда. Примерами угроз могут служить взломщики, вирусы, пожары, природные катаклизмы.

После оценки возможных угроз (рисков) переходят к выработке контрмер. Под контрмерой понимают действие, позволяющее минимизировать риск от определенного уязвимого компонента или некоторой угрозы. Одной из самых эффективных контрмер минимизации риска потери данных является создание надежной системы резервного копирования.

Результаты оценки рисков и выработанные контрмеры используются для создания плана безопасности, который должен в мельчайших подробностях описывать системные стратегии организации, имеющие непосредственное и отдаленное отношение к вопросам безопасности.

Планирование безопасности сети и данных.

Высокая степень безопасности может быть достигнута путем использования плана, предусматривающего применение различных мер и средств обеспечения безопасности.

Оценка требований к безопасности сетевых данных является первым этапом разработки плана по принятию мер их защиты. При этом должны быть учтены характер деятельности организации и хранящихся в сети данных, стратегия и стиль управления организацией, которые должен знать сетевой администратор и реализовать его в подведомственной ему сети.

Высокий уровень безопасности данных должен поддерживаться в организациях, располагающих данными, которые являются строго конфиденциальными по своей природе. Примером могут служить коммерческие организации, предоставляющие услуги или выпускающие продукцию в областях с высоким уровнем конкуренции. Некоторые виды данных должны быть защищены независимо от характера деятельности

организации. К ним относятся бухгалтерская документация, налоговая информация, промышленные секреты (планы деятельности организаций и коммерческие планы, рецепты, технологии изготовления, тексты программ).

Для принятия мер по защите данных в сети нужно выявить главные источники угроз их безопасности.

Существуют следующие виды угроз:

• непреднамеренные, к которым относятся ошибочные действия лояльных сотрудников, стихийные бедствия, ненадежность работы программно-аппаратных средств и др.;

• преднамеренные, которые явно направлены на причинение ущерба информационной безопасности;

• внешние, которые проявляются в таких формах, как несанкционированное использование паролей и ключей; атаки DoS (Denial of Service — отказ в обслуживании), направленные на разрыв сетевого соединения или приведение его в неработоспособное состояние; подмена адреса; компьютерные вирусы и черви;

• внутренние, к которым можно отнести промышленный шпионаж, интриги и недовольство служащих, случайные нарушения и т.п.

В плане безопасности должны быть самым детальным образом перечислить процедуры, выполнение которых предписывается политикой безопасности. Каждый сотрудник, отвечающий за выполнение конкретной процедуры, должен быть предупрежден о возможных последствиях в случае отступления от предписанного способа выполнения процедуры. Рекомендуется взять с сотрудника письменное подтверждение того, что он понимает смысл стратегии безопасности, согласен с ней и обязуется ей следовать, а также регулярно обновлять план, т.е. пересматривать аспекты безопасности, пытаясь определить новые потенциально уязвимые компоненты, угрозы и контрмеры для борьбы с ними, и отражать изменения в плане.

Средства обеспечения безопасности

Для безопасности сети используется широкий набор различных средств и технологий. Рассмотрим некоторые из них.

В разных программных и аппаратных продуктах, предназначенных для защиты данных, часто используются одинаковые подходы, приемы и технические решения, которые в совокупности образуют технологию безопасности.

Криптозащита. Разработкой методов преобразования информации в целях ее защиты занимается криптография.

Преобразование общедоступных (понятных для всех) данных к виду, затрудняющему их распознавание, называется шифрованием (Encryption), а обратное преобразование - дешифрованием (Decryption). Шифрование является доступным средством для администраторов и пользователей и одним из эффективных средств обеспечения конфиденциальности информации. Следует выделить два основных способа шифрования данных: перестановку (Transposition), когда в исходных данных изменяют последовательность символов, и замену (Substitution), при которой с помощью некоторого шаблона производят замену всех символов используемого алфавита, например буквы заменяют цифрами.

Операции шифрования и дешифрования данных (информации) осуществляются с помощью ключей, которые создаются с привлечением математических формул.

Метод, при котором для обеих операций используется один ключ, называется симметричной криптографией (Symmetric Cryptography). При асимметричной криптографии (Asymmetric Cryptography) каждый пользователь сети должен располагать двумя ключами: общим (Public key) и частным (Private key). Оба ключа связаны друг с другом с помощью некоторой математической функции. Общий ключ известен каждому пользователю. Зашифрованное с помощью общего ключа сообщение может быть прочитано только с помощью частного ключа. Поскольку

предполагается, что пользователь, которому адресуется сообщение, не разглашает свой ключ, он является единственным человеком, который может прочитать сообщение.

Популярны два алгоритма шифрования: симметричный DES (Data Encryption Standard — стандарт шифрования данных, который является официальным стандартом правительства США) и несимметричный RSA, разработанный учеными Rivest, Shamir, Adle- man и названный по начальным буквам их фамилий.

Для шифрования, аутентификации и проверки целостности передаваемых по сети пакетов разработан протокол IPSec (IP Security), включающий в себя протокол AH (Authentication Header), позволяющий проверять идентичность отправителя, и протокол ESP (Encapsulating Security Payloads), обеспечивающий конфиденциальность самих данных. Протокол IPSec поддерживают маршрутизаторы компании Cisco Systems и ОС Windows 2000/XP.

Для передачи через Internet зашифрованных, аутентифицированных сообщений используется протокол SSL (Secure Sockets Layer — уровень защищенных сокетов, или гнезд). В этом протоколе криптографическая система с открытым ключом комбинируется с блочным шифрованием данных.

Аутентификация (Authentication).

Это процедура установления подлинности пользователя при запросе доступа к ресурсам системы (компьютеру или сети). Аутентификация предотвращает доступ нежелательных лиц и разрешает доступ всем легальным пользователям. В процедуре аутентификации участвуют две стороны, одна из которых доказывает свое право на доступ (аутентичность), предъявляя некоторые аргументы, другая — проверяет эти аргументы и принимает решение. Для доказательства аутентичности может использоваться некоторое известное для обеих сторон слово (пароль) или уникальный физический предмет (ключ), а также собственные

биохарактеристики (отпечатки пальцев или рисунок радужной оболочки глаза).

Наиболее часто при аутентификации используют вводимые с клавиатуры пароли.

Пароль представляет собой зашифрованную последовательность символов, которая держится в секрете и предъявляется при обращении к информационной системе.

Объектами аутентификации могут быть не только пользователи, но и различные устройства, приложения, текстовая и другая информация.

Идентификация субъектов и объектов доступа.

Идентификация предусматривает закрепление за каждым субъектом доступа уникального имени в виде номера, шифра или кода, например, персональный идентификационный номер (Personal Identification Number — PIN), социальный безопасный номер (Social Security Number — SSN) и т. п. Идентификаторы пользователей должны быть зарегистрированы в информационной системе администратором службы безопасности.

При регистрации в базу данных системы защиты для каждого пользователя заносятся такие данные, как фамилия, имя, отчество и уникальный идентификатор пользователя, имя процедуры для установления подлинности и пароль пользователя, полномочия пользователя по доступу к системным ресурсам и др. Идентификацию следует отличать от аутентификации. Идентификация заключается в сообщении пользователем системе своего идентификатора, в то время как аутентификация является процедурой доказательства пользователем того, что именно ему принадлежит введенный им идентификатор.

Авторизация (Authorization).

Это процедура предоставления каждому из пользователей тех прав доступа к каталогам, файлам и принтерам, которыми его наделил

администратор. Кроме того, система авторизации может контролировать возможность выполнения пользователями различных системных функций, таких как установка системного времени, создание резервных копий данных, локальный доступ к серверу, выключение сервера и т. п.

Система авторизации наделяет пользователя сети правами выполнять определенные действия над определенными ресурсами. Для этого могут быть использованы два подхода к определению прав доступа:

- избирательный, при котором отдельным пользователям (или группам), явно указанным своими идентификаторами, разрешаются или запрещаются определенные операции над определенным ресурсом;
- мандатный, при котором вся информация в зависимости от степени секретности делится на уровни, а все пользователи сети — на группы, образующие иерархию в соответствии с уровнем допуска к этой информации.

Процедуры авторизации реализуются программными средствами по централизованной схеме, в соответствии с которой пользователь один раз логически входит в сеть и получает на все время работы некоторый набор разрешений по доступу к ресурсам сети, и децентрализованной схеме, когда доступ к каждому приложению должен контролироваться средствами безопасности самого приложения или средствами той операционной среды, в которой оно работает.

Поскольку системы аутентификации и авторизации совместно выполняют одну задачу, необходимо предъявлять к ним одинаковый уровень требований. Ненадежность одной системы не может быть компенсирована высоким качеством другой.

Аудит (Auditing). Это фиксация в системном журнале событий, связанных с доступом к защищаемым системным ресурсам. Аудит используется для обнаружения неудачных попыток взлома системы. При попытке выполнить противоправные действия система аудита идентифицирует нарушителя и пишет сообщение в журнал регистрации.

Анализ накопившейся и хранящейся в журнале информации может оказаться действенной мерой защиты от несанкционированного доступа.

Процедура рукопожатий. Для установления подлинности пользователей широко используется процедура рукопожатий (Handshaking — согласованный обмен, квитирование), построенная по принципу вопрос-ответ. Она предполагает, что правильные ответы на вопросы дают только те пользователи, для которых эти вопросы предназначены. Для подтверждения подлинности пользователя система последовательно задает ему ряд случайно выбранных вопросов, на которые он должен дать ответ. Опознание считается положительным, если пользователь правильно ответил на все вопросы.

Технологии защищенного канала широко используются в виртуальных частных сетях, которые требуют принятия дополнительных мер по защите передаваемой информации. Требование конфиденциальности особенно важно, потому что пакеты, передаваемые по публичной сети, уязвимы для перехвата при их прохождении через каждый из узлов (серверов) на пути от источника к получателю. Технология защищенного канала включает в себя:

- взаимную аутентификацию абонентов при установлении соединения;
- защиту передаваемых по каналу сообщений от несанкционированного доступа;
- подтверждение целостности поступающих по каналу сообщений.

В зависимости от места расположения программного обеспечения защищенного канала различают две схемы его образования.

Схема с конечными узлами (рис.3.4.1). В этой схеме защищенный канал образуется программными средствами, установленными на двух удаленных компьютерах. Компьютеры принадлежат двум разным АС одной организации и связаны между собой через публичную сеть.

Схема с оборудованием поставщика услуг публичной сети, расположенным на границе между частной и публичной сетями (рис. 3.4.1).

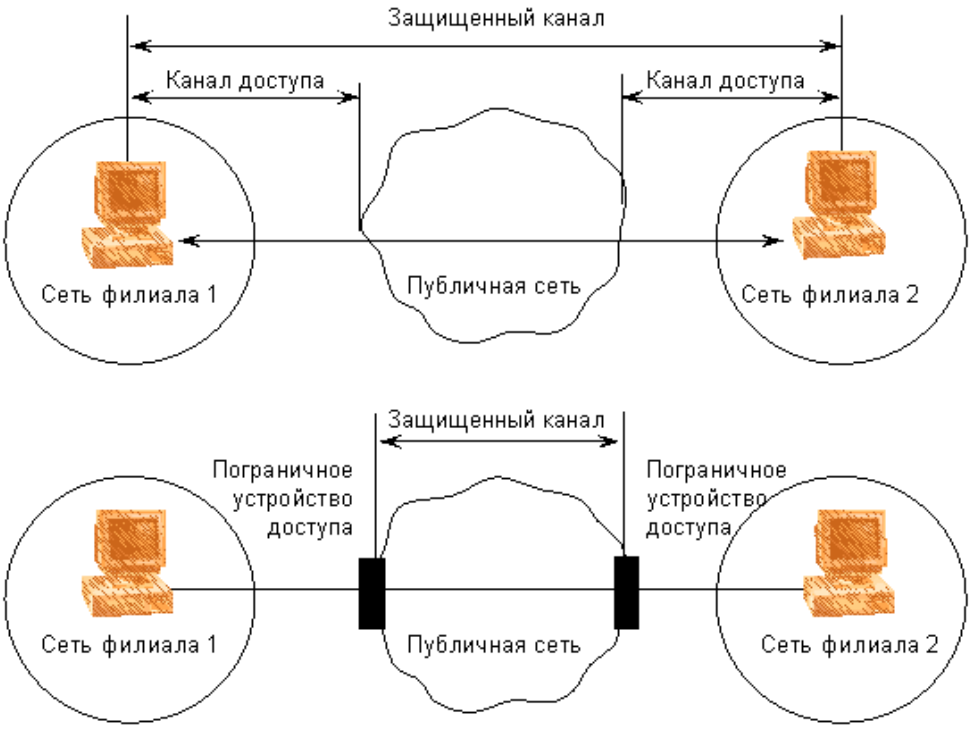


Рис.3.4.1. Две схемы образования защищенного канала

В этой схеме защищенный канал прокладывается только внутри публичной сети с коммутацией пакетов. Средствами защиты являются пограничные устройства доступа (ПУД).

Средства безопасности, предоставляемые операционными системами

Современные операционные системы (ОС) способны обеспечить доступ к одному компьютеру и сетевым ресурсам многим пользователям. Для этого используются отдельные учетные записи, которым присвоены разные пароли. После правильного ввода регистрационной информации пользователь может получить доступ к ОС и сети; читать, изменять ресурсы и выполнять любые другие действия, которые соответствуют правам его учетной записи, создавать желаемую конфигурацию пользовательского интерфейса (рабочую среду) и т.д.

Выбор (или назначение) паролей подчинен стратегии обеспечения сетевой безопасности. Пароли должны удовлетворять определенным требованиям. Многие сетевые ОС позволяют администратору задавать длину

и время жизни пароля; проверять пароль на наличие заданного пароля в словаре и, если он есть, предотвращать использование пароля; следить, чтобы пароль пользователя не повторялся. Кроме того, администратору предоставляются широкие возможности контроля за доступом к ресурсам. Например, одной и той же учетной записи он может одновременно разрешить просматривать содержимое файла filel. doc, но запретить вносить в него изменения; предоставить право читать, изменять, удалять файл file2. doc и даже устанавливать права доступа к нему других пользователей, а к файлу file3.doc отменить все права доступа.

В файловых системах с высоким уровнем безопасности права доступа можно устанавливать как на разделение ресурсов по сети, так и на использование этих ресурсов на одном и том же локальном компьютере. Локальные и сетевые права доступа могут не совпадать. Например, пользователю можно предоставить право полного контроля над файлом file4 .doc, когда он регистрируется на компьютере, хранящем этот файл, но ограничить право доступа того же пользователя к file4. doc при попытке получить к нему доступ с другого компьютера сети.

Аппаратные средства защиты

Основой надежной защиты данных от многих неисправностей аппаратных средств является избыточность. При выходе из строя некоторого сетевого устройства начинает функционировать его резервный дублер. Потерю данных при выходе из строя винчестера можно восполнить файлами, хранящимися в системе резервного копирования.

Некоторые серверы поддерживают возможность установки избыточных устройств, автоматически передающих полномочия отказавшего компонента исправному. Такая избыточность применима к охлаждающим вентиляторам, источникам питания, сетевым адаптерам, жестким дискам и центральным процессорам.

При резервировании электропитания используют избыточные

источники электроэнергии - устройство бесперебойного питания наряду с электросетью. Резервное копирование данных предполагает создание избыточных копий ценных файлов на дополнительных (резервных) носителях. В системах отказоустойчивых дисков данные записываются на избыточных дисках. Высшей степенью избыточности является кластеризация, когда несколько серверов объединяются в группу. В сети кластер серверов виден пользователям как один сервер. Если один из серверов кластера выходит из строя, его обязанности выполняет другой сервер. Пользователи не замечают этого перехода.

Резервное копирование данных. Оно осуществляется с помощью специальных программ и является действенной мерой защиты от возможной их потери при регулярном выполнении этой процедуры. Наличие резервной копии позволяет быстро восстановить утерянные данные.

Используются следующие способы резервного копирования:

• полное, при котором копируются все данные заданных дисков независимо от того, когда их копирование выполнялось последний раз и вносились ли с тех пор изменения;

• дифференциальное, когда копируются все файлы, которые изменялись со времени последнего полного копирования. Дифференциальное копирование выполняется в промежутках между полным копированием, благодаря этому экономится время. Для обновления данных нужно восстанавливать две последних копии - полную и дифференциальную;

• инкрементное. При этом способе копируются все файлы, которые изменялись со времени любого последнего копирования (а не последнего полного копирования). Это наиболее быстрый способ, однако он сложнее и занимает много времени на восстановление данных, так как необходимо восстанавливать последнюю полную копию и все инкрементные копии, созданные со времени последнего полного копирования.

Вопросы для самоконтроля

1. Основные понятия безопасности.
2. Планирование безопасности сети и данных.
3. Средства обеспечения безопасности.
4. Базовые технологии безопасности.
5. Аутентификация (Authentication).
6. Идентификация субъектов и объектов доступа.
7. Авторизация (Authorization).
8. Аудит (Auditing).
9. Схемы образования защищенного канала.
10. Средства безопасности, предоставляемые операционными системами.
11. Аппаратные средства защиты.

ГЛАВА IV. СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

4.1. Алгоритмизация и программирование процессов в технических системах. Этапы решения проблем на компьютере.

Основные модули

- Виды и методы создание алгоритма.
- Графический способ записи алгоритмов.
- Этапы решения задач на компьютере.
- Современные технологии программирования.

Понятие алгоритма такое же основополагающее для информатики, как и понятие информации. Именно поэтому важно в нем разобраться.

Название **"алгоритм"** произошло от латинской формы имени величайшего среднеазиатского математика **Мухаммеда ибн Муса ал-Хорезми** (Alhorithmi), жившего в 783—850 гг.

Человек ежедневно встречается с необходимостью следовать тем или иным правилам, выполнять различные инструкции и указания. Например, переходя через дорогу на перекрестке без светофора надо сначала посмотреть направо. Если машин нет, то перейти полдороги, а если машины есть, ждать, пока они пройдут, затем перейти полдороги. После этого посмотреть налево и, если машин нет, то перейти дорогу до конца, а если машины есть, ждать, пока они пройдут, а затем перейти дорогу до конца.

В математике для решения типовых задач мы используем определенные правила, описывающие последовательности действий. Например, правила сложения дробных чисел, решения квадратных уравнений и т. д. Обычно любые инструкции и правила представляют собой последовательность действий, которые необходимо выполнить в определенном порядке. Для решения задачи надо знать, что дано, что следует получить и какие действия и в каком порядке следует для этого выполнить.

Предписание, определяющее порядок выполнения действий над данными с целью получения искомых результатов, и есть алгоритм.

Алгоритм — заранее заданное понятное и точное предписание возможному исполнителю совершить определенную последовательность действий для получения решения задачи за конечное число шагов.

Алгоритм – точный набор инструкций, описывающих порядок действий исполнителя (компьютера), от допустимых исходных данных для достижения результата решения задачи за конечное время. В старой трактовке вместо слова "порядок" использовалось *слово* "последовательность", но *по* мере развития параллельности в работе компьютеров *слово* "последовательность" стали заменять более общим словом "порядок". Это связано с тем, что работа каких-то инструкций алгоритма может быть зависима от других инструкций или результатов их работы. Таким образом, некоторые инструкции должны выполняться строго после завершения работы инструкций, от которых они зависят. Независимые инструкции или инструкции, ставшими независимыми из-за завершения работы инструкций, от которых они зависят, могут выполняться в произвольном порядке, параллельно или одновременно, если это позволяют используемые *процессор* и *операционная система*.

Какими свойствами обладают алгоритмы?

Основные свойства алгоритмов следующие:

Понятность для исполнителя — исполнитель алгоритма должен понимать, как его выполнять. Иными словами, имея алгоритм и произвольный вариант исходных данных, исполнитель должен знать, как надо действовать для выполнения этого алгоритма.

Дискретность (прерывность, раздельность) — алгоритм должен представлять процесс решения задачи как последовательное выполнение простых (или ранее определенных) шагов (этапов).

Определенность — каждое правило алгоритма должно быть четким, однозначным и не оставлять места для произвола. Благодаря этому свойству выполнение алгоритма носит механический характер и не требует никаких дополнительных указаний или сведений о решаемой задаче.

Результативность (или конечность) состоит в том, что за конечное число шагов алгоритм либо должен приводить к решению задачи, либо после конечного числа шагов останавливаться из-за невозможности получить решение с выдачей соответствующего сообщения, либо неограниченно продолжаться в течение времени, отведенного для исполнения алгоритма, с выдачей промежуточных результатов.

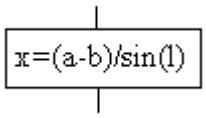
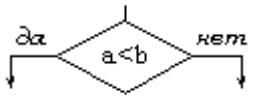
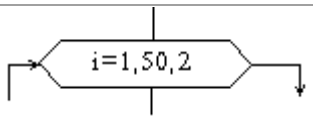
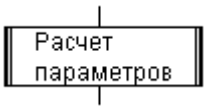
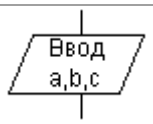
Массовость означает, что алгоритм решения задачи разрабатывается в общем виде, т.е. он должен быть применим для некоторого класса задач, различающихся лишь исходными данными. При этом исходные данные могут выбираться из некоторой области, которая называется областью применимости алгоритма.

Графический способ представления алгоритмов является более компактным и наглядным по сравнению со словесным.

При графическом представлении алгоритм изображается в виде последовательности связанных между собой функциональных блоков, каждый из которых соответствует выполнению одного или нескольких действий.

Такое графическое представление называется схемой алгоритма или **блок-схемой**. В блок-схеме каждому типу действий (вводу исходных данных, вычислению значений выражений, проверке условий, управлению повторением действий, окончанию обработки и т.п.) соответствует геометрическая фигура, представленная в виде **блочного символа**. Блочные символы соединяются **линиями переходов**, определяющими очередность выполнения действий. В таблице приведены наиболее часто употребляемые символы (таблица 4.1.1).

Таблица 4.1.1

Название символа	Обозначение и пример заполнения	Пояснение
Процесс		Вычислительное действие или последовательность действий
Решение		Проверка условий
Модификация		Начало цикла
Предопределенный процесс		Вычисления по подпрограмме, стандартной подпрограмме
Ввод-вывод		Ввод-вывод в общем виде
Пуск-останов		Начало, конец алгоритма, вход и выход в подпрограмму
Документ		Вывод результатов на печать

1. Блок **"процесс"** применяется для обозначения действия или последовательности действий, изменяющих значение, форму представления или размещения данных. Для улучшения наглядности схемы несколько отдельных блоков обработки можно объединять в один блок. Представление отдельных операций достаточно свободно.

2. Блок **"решение"** используется для обозначения переходов управления по условию. В каждом блоке "решение" должны быть указаны вопрос, условие или сравнение, которые он определяет.

3. Блок **"модификация"** используется для организации циклических конструкций. (Слово модификация означает видоизменение, преобразование). Внутри блока записывается параметр цикла, для которого указываются его начальное значение, граничное условие и шаг изменения значения параметра для каждого повторения.

4. Блок **"предопределенный процесс"** используется для указания обращений к вспомогательным алгоритмам, существующим автономно в виде некоторых самостоятельных модулей, и для обращений к библиотечным подпрограммам.

Способы представления алгоритма, вспомогательные алгоритмы

Базовые структуры алгоритмов — это определенный набор блоков и стандартных способов их соединения для выполнения типичных последовательностей действий.

К основным структурам относятся следующие:

- линейные
- разветвляющиеся
- циклические

Линейными называются алгоритмы, в которых действия осуществляются последовательно друг за другом. Стандартная блок-схема линейного алгоритма приводится ниже (рис. 4.1.1.):

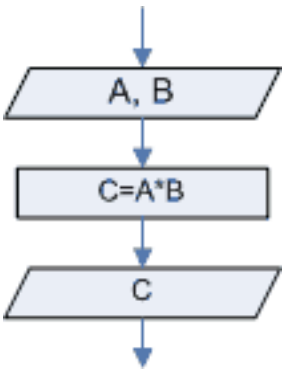


Рис.4.1.1. Блок-схема линейного процесса

Разветвляющимся называется алгоритм, в котором действие выполняется по одной из возможных ветвей решения задачи, в зависимости от выполнения условий. В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в разветвляющиеся алгоритмы входит условие, в зависимости от выполнения или невыполнения которого выполняется та или иная последовательность команд (действий).

В качестве условия в разветвляющемся алгоритме может быть использовано любое понятное исполнителю утверждение, которое может соблюдаться (быть истинно) или не соблюдаться (быть ложно). Такое утверждение может быть выражено как словами, так и формулой. Таким образом, алгоритм ветвления состоит из условия и двух последовательностей команд.

В зависимости от того, в обоих ветвях решения задачи находится последовательность команд или только в одной разветвляющиеся алгоритмы делятся на полные и не полные (сокращенные).

Стандартные блок-схемы разветвляющегося алгоритма приведены ниже (рис. 4.1.2.):

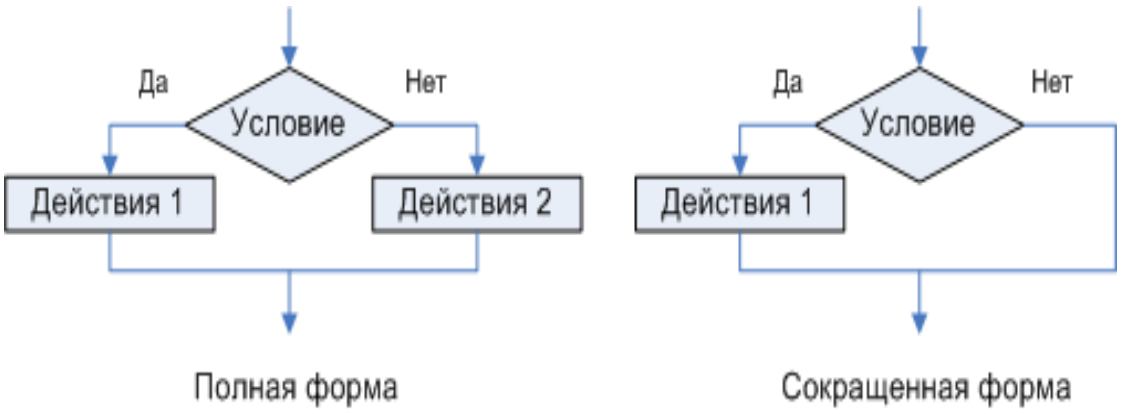


Рис.4.1.2. Блок-схема разветвленного процесса

Циклическим называется алгоритм, в котором некоторая часть операций (тело цикла — последовательность команд) выполняется многократно. Однако слово «многократно» не значит «до бесконечности». Организация циклов, никогда не приводящая к остановке в выполнении алгоритма, является нарушением требования его результативности — получения результата за конечное число шагов.

Перед операцией цикла осуществляются операции присвоения начальных значений тем объектам, которые используются в теле цикла. В цикл входят в качестве базовых следующие структуры:

1. блок проверки условия
2. блок, называемый телом цикла

Существуют три типа циклов:

1. Цикл с предусловием
2. Цикл с постусловием
3. Цикл с параметром (разновидность цикла с предусловием)

Если тело цикла расположено после проверки условий, то может случиться, что при определенных условиях тело цикла не выполнится ни разу. Такой вариант организации цикла, управляемый предусловием, называется циклом с предусловием.

Возможен другой случай, когда тело цикла выполняется по крайней мере один раз и будет повторяться до тех пор, пока не станет ложным

условие. Такая организация цикла, когда его тело расположено перед проверкой условия, носит название цикла с постусловием.

Цикл с параметром является разновидностью цикла с предусловием. Особенностью данного типа цикла является то, что в нем имеется параметр, начальное значение которого задается в заголовке цикла, там же задается условие продолжения цикла и закон изменения параметра цикла. Механизм работы полностью соответствует циклу с предусловием, за исключением того, что после выполнения тела цикла происходит изменение параметра по указанному закону и только потом переход на проверку условия.

Стандартные блок-схемы циклических алгоритмов приведены ниже (рис. 4.1.3.):

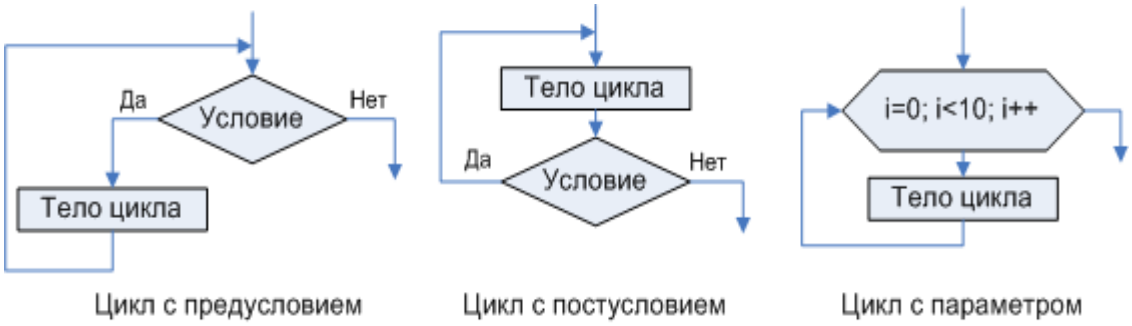


Рис. 4.1.3. Блок-схема циклического процесса

Современные технологии программирования.

Основной **компонент программного обеспечения - программа** - упорядоченная в соответствии с некоторым алгоритмом последовательность команд (инструкций) компьютера для решения задачи пользователя. Чаще всего образ программы хранится в виде исполняемого модуля (отдельного файла или группы файлов).

Пользователь – лицо, заинтересованное в решении некоторой задачи средствами вычислительной техники. *По* отношению к программному обеспечению компьютерные пользователи делятся на следующие группы:

- системные программисты, занимающиеся разработкой, эксплуатацией и сопровождением системного программного обеспечения;

• прикладные программисты. Выполняют разработку и отладку программ решения задач из различных прикладных сфер деятельности пользователей;

• конечные пользователи. Используют прикладное программное обеспечение для решения задач в своей повседневной деятельности. Различаются по уровню своей подготовки в части знания и использования компьютерной техники;

• администраторы. Как правило, это высококвалифицированные компьютерные специалисты, отвечающие за работу вычислительной сети, баз данных, корпоративной информационной системы в целом, безопасность и защиту данных. Могут иметь определенную специализацию: управление сетевым каталогом, политикой учетных записей, политикой аудита и т.п.

Задача (*problem,* task) – проблема, подлежащая решению в интересах пользователя.

Термин "задача" в программировании означает единицу работы вычислительной системы, требующую выделения вычислительных ресурсов (процессорного времени, оперативной и внешней памяти, файлов и т.п.).

Приложение (*application*) – *программная реализация* решения задачи на компьютере. *Приложение* может состоять из одной или нескольких взаимосвязанных и взаимодействующих программ.

Принято (весьма условно) делить программы на небольшие (простые), *средней сложности* и большие.

Программа считается небольшой как по размерам, так и по другим признакам, если она удовлетворяет следующим признакам:

- решает одну четко поставленную задачу в хорошо известных ограничениях, к тому же, не очень существенную для какой-либо практической или исследовательской деятельности;
- неважно, насколько быстро она работает;

- ущерб от неправильной работы программы – практически нулевой (за исключением возможности обрушения ею системы, в которой выполняются и другие, более важные задачи);
- не требуется дополнять программу новыми возможностями, практически никому не нужно разрабатывать ее новые версии или исправлять найденные ошибки;
- в связи со сказанным выше не очень нужно прилагать к программе подробную и понятную документацию – для человека, который ею заинтересуется, не составит большого труда понять, как ею пользоваться, просто по исходному коду.

Сложные, или большие, программы, называемые также программными системами, программными комплексами, программными продуктами, отличаются от небольших не столько *по* размерам (хотя обычно они значительно больше), сколько наличием дополнительных факторов. Эти факторы связаны с их востребованностью и готовностью пользователей платить деньги, как за приобретение самой программы, так и за ее сопровождение и даже за специальное обучение работе с ней.

Примером большой программы может служить *стандартная библиотека* классов *Java,* C# или Phyton, соответствующих систем программирования.

Строго говоря, ни одно из указанных свойств не является обязательным для того, чтобы программу можно было считать большой, но при наличии двух-трех из них достаточно уверенно можно утверждать, что она большая. На основании некоторых из перечисленных свойств можно сделать *вывод*, что большая *программа* или программная система чаще всего представляет собой не просто код или *исполняемый файл*, а включает еще и набор проектной и пользовательской документации.

Программирование (*programming*) - теоретическая и практическая *деятельность*, связанная с созданием программ.

Разработка программных систем (ПС), т.е. *программирование*, имеет ряд специфических особенностей. Прежде всего, следует отметить некоторое противостояние: неформальный характер требований к ПС (постановки задачи) и понятия ошибки в нем, но формализованный основной *объект* разработки – программы ПС. Тем самым разработка ПС содержит определенные этапы формализации, а переход от неформального к формальному существенно неформален.

Разработка ПС носит творческий характер (на каждом шаге приходится делать какой-либо выбор, принимать какое-либо решение), а не сводится к выполнению некоей последовательности регламентированных действий. Тем самым эта разработка ближе к процессу проектирования сложных устройств, но никак не к их массовому производству. Этот творческий характер разработки ПС сохраняется до самого ее конца.

Технологии программирования

В процессе разработки программных систем используются различные технологии программирования. В соответствии с обычным значением слова "технология" под технологией программирования (programming technology) понимается совокупность производственных процессов, приводящая к созданию требуемой ПС, а также описание этой совокупности процессов. Другими словами, технология программирования понимается здесь в широком смысле как технология разработки программных средств, включая в нее все процессы, начиная с момента зарождения идеи этого средства до создания необходимой программной документации. Каждый процесс этой совокупности базируется на использовании каких-либо методов и средств, например, компьютера (в этом случае речь идет о компьютерной технологии программирования).

Решение задач с помощью компьютера включает в себя следующие основные этапы, часть из которых осуществляется без участия компьютера.

1. **Постановка задачи:**
   - сбор информации о задаче;
   - формулировка условия задачи;
   - определение конечных целей решения задачи;
   - определение формы выдачи результатов;
   - описание данных (их типов, диапазонов величин, структуры и т.п.).

Постановка задачи (*problem definition*) – это точная формулировка требований (функциональных и нефункциональных), предъявляемых к работе программы, с описанием *входной* и выходной информации, и, возможно, описание подходов к решению задачи.

2. **Анализ и исследование задачи, модели:**
   - анализ существующих аналогов;
   - анализ технических и программных средств;
   - разработка математической модели;
   - разработка структур данных.
3. **Разработка алгоритма:**
   - выбор метода проектирования алгоритма;
   - выбор формы записи алгоритма (блок-схемы, псевдокод и др.);
   - выбор тестов и метода тестирования;
   - проектирование алгоритма.
4. **Программирование:**
   - выбор языка программирования;
   - уточнение способов организации данных;
   - запись алгоритма на выбранном языке программирования.
5. **Тестирование и отладка:**
   - синтаксическая отладка;
   - отладка семантики и логической структуры;
   - тестовые расчеты и анализ результатов тестирования;
   - совершенствование программы.

6. **Анализ результатов решения задачи** и уточнение в случае необходимости математической модели с повторным выполнением этапов 2 - 5.
7. **Сопровождение программы:**

• доработка программы для решения конкретных задач;

• составление документации к решенной задаче, к математической модели, к алгоритму, к программе, к набору тестов, к использованию.

Итак, создавая математическую модель для решения задачи, нужно:

• выделить предположения, на которых будет основываться математическая модель;

• определить, что считать исходными данными и результатами;

• записать математические соотношения, связывающие результаты с исходными данными.

При построении математических моделей далеко не всегда удается найти формулы, явно выражающие искомые величины через данные. В таких случаях используются математические методы, позволяющие дать ответы той или иной степени точности.

Вопросы для самоконтроля

1. Виды и методы создание алгоритма.
2. Этапы решения задач в компьютере.
3. Графический способ записи алгоритмов.
4. Современные технологии программирования.

4.2. Язык программирования PYTHON и его синтаксис.

<u>Основные модули</u>

• Основные особенности языка Python

• Типы данных. Переменные

• Операции в программировании

- Изменение типов данных
- Ввод и вывод данных
- Ввод данных. Функция input()

Основные особенности языка

Python – интерпретируемый язык программирования. Это значит, что исходный код частями преобразуется в машинный в процессе его чтения специальной программой – интерпретатором.

Python характеризуется ясным синтаксисом. Читать код на нем легче, чем на других языках программирования, так как в Питоне мало используются такие вспомогательные синтаксические элементы как скобки, точки с запятыми. С другой стороны, правила языка заставляют программистов делать отступы для обозначения вложенных конструкций. Понятно, что хорошо оформленный текст с малым количеством отвлекающих элементов читать и понимать легче.

Python – это полноценный во многом универсальный язык программирования, используемый в различных сферах. Основная, но не единственная, поддерживаемая им парадигма, – объектно-ориентированное программирование. Однако в данном курсе мы только упомянем об объектах, а будем изучать структурное программирование, так как оно является базой. Без знания основных типов данных, ветвлений, циклов, функций нет смысла изучать более сложные парадигмы, так как в них все это используется.

Типы данных. Переменные

В реальной жизни мы совершаем различные действия над окружающими нас предметами, или объектами. Мы меняем их свойства, наделяем новыми функциями. По аналогии с этим компьютерные программы также управляют объектами, только виртуальными, цифровыми. Пока не дойдем до уровня объектно-ориентированного программирования, будем называть такие объекты **данными**.

Очевидно, данные бывают разными. Часто компьютерной программе приходится работать с числами и строками. Так на прошлом уроке мы работали с числами, выполняя над ними арифметические операции. Операция сложения выполняла изменение первого числа на величину второго, а умножение увеличивало одно число в количество раз, соответствующее второму.

Числа в свою очередь также бывают разными: целыми, вещественными, могут иметь огромное значение или очень длинную дробную часть.

При знакомстве с языком программирования Python мы столкнемся с тремя типами данных:

целые числа (тип **int**) – положительные и отрицательные целые числа, а также 0 (например, 4, 687, -45, 0).

числа с плавающей точкой (тип **float**) – дробные, они же вещественные, числа (например, 1.45, -3.789654, 0.00453). Примечание: для разделения целой и дробной частей здесь используется точка, а не запятая.

строки (тип **str**) — набор символов, заключенных в кавычки (например, "ball", "What is your name?", '6589'). Примечание: кавычки в Python могут быть одинарными или двойными; одиночный символ в кавычках также является строкой, отдельного символьного типа в Питоне нет.

Операции в программировании

Операция – это выполнение каких-либо действий над данными, которые в данном случае именуют **операндами**. Само действие выполняет **оператор** – специальный инструмент. Если бы вы выполняли операцию постройки стола, то вашими операндами были бы доска и гвоздь, а оператором – молоток (рис.4.2.1).

Рис. 4.2.1. Операции в программировании

Так в математике и программировании символ плюса является оператором операции сложения по отношению к числам. В случае строк этот же оператор выполняет операцию *конкатенации*, то есть соединения.

```
>>> 10.25 + 98.36
108.61
>>> 'Hello' + 'World'
'HelloWorld'
```

Здесь следует для себя отметить, что то, что делает оператор в операции, зависит не только от него, но и от типов данных, которыми он оперирует. Молоток в случае нападения на вас крокодила перестанет играть роль строительного инструмента. Однако в большинстве случаев операторы не универсальны. Например, знак плюса неприменим, если операндами являются, с одной стороны, число, а с другой – строка.

```
>>> 1 + 'a'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for +:
'int' and 'str'
```

Здесь в строке TypeError: unsupported operand type(s) for +: 'int' and 'str', интерпретатор сообщает, что произошла ошибка типа – неподдерживаемый операнд для типов int и str.

Изменение типов данных

Приведенную выше операцию все-таки можно выполнить, если превратить число 1 в строку "1". Для изменения одних типов данных в другие в языке Python предусмотрен ряд встроенных в него функций (что такое функция в принципе, вы узнаете в других уроках). Поскольку мы пока работаем только с тремя типами (int, float и str), рассмотрим вызовы соответствующих им функций – int(), float(), str().

```
>>> str(1) + 'a'
'1a'
>>> int('3') + 4
7
>>> float('3.2') + int('2')
5.2
>>> str(4) + str(1.2)
'41.2'
```

Эти функции преобразуют то, что помещается в их скобки соответственно в целое число, вещественное число или строку. Однако преобразовать можно не все:

```
>>> int('hi')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with
base 10: 'hi'
```

Здесь возникла ошибка значения (ValueError), так как передан литерал (в данном случае строка с буквенными символами), который нельзя преобразовать к числу с основанием 10. Однако функция int не такая простая:

```
>>> int('101', 2)
5
>>> int('F', 16)
15
```

Если вы знаете о различных системах счисления, то поймете, что здесь произошло.

Обратим внимание еще на одно. Данные могут называться **значениями**, а также **литералами**. Эти три понятия ("данные", "значение", "литерал") не обозначают одно и то же, но близки и нередко употребляются как синонимы. Чтобы понять различие между ними, места их употребления, надо изучить программирование глубже.

Переменные

Данные хранятся в ячейках памяти компьютера. Когда мы вводим число, оно помещается в какую-то ячейку памяти. Но как потом узнать, куда именно? Как впоследствии обращаться к этим данными? Нужно как-то запомнить, пометить соответствующую ячейку.

Раньше, при написании программ на машинном языке, обращение к ячейкам памяти осуществляли с помощью указания их регистров, то есть конкретно сообщали, куда положить данные и откуда их взять. Однако с появлением ассемблеров при обращении к данным стали использовать словесные **переменные**, что куда удобней для человека.

Механизм связи между переменными и данными может различаться в зависимости от языка программирования и типов данных. Пока достаточно запомнить, что в программе данные связываются с каким-либо именем и в дальнейшем обращение к ним возможно по этому имени-переменной.

Слово "переменная" обозначает, что сущность может меняться, она непостоянна. Действительно, вы увидите это в дальнейшем, одна и та же переменная может быть связана сначала с одними данными, а потом – с другими. То есть ее значение может меняться, она переменчива.

В программе на языке Python, как и на большинстве других языков, связь между данными и переменными устанавливается с помощью знака =. Такая операция называется присваивание (также говорят "присвоение"). Например, выражение sq = 4 означает, что на объект, представляющий собой число 4, находящееся в определенной области памяти, теперь ссылается переменная *sq*, и обращаться к этому объекту следует по имени *sq* (рис.4.2.2).

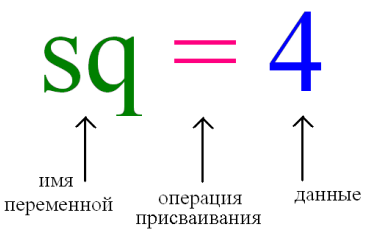


Рис. 4.2.2. Присвоение переменных

Имена переменных могут быть любыми. Однако есть несколько общих правил их написания:

Желательно давать переменным осмысленные имена, говорящие о назначении данных, на которые они ссылаются.

Имя переменной не должно совпадать с командами языка (зарезервированными ключевыми словами).

Имя переменной должно начинаться с буквы или символа подчеркивания (_), но не с цифры.

Имя переменной не должно содержать пробелы.

Чтобы узнать значение, на которое ссылается переменная, находясь в режиме интерпретатора, достаточно ее вызвать, то есть написать имя и нажать Enter.

```
>>> sq = 4
>>> sq
4
```

Вот более сложный пример работы с переменными в интерактивном режиме:

```
>>> apples = 100
>>> eat_day = 5
>>> day = 7
>>> apples = apples - eat_day * day
>>> apples
65
```

Здесь фигурируют три переменные: *apples, eat_day* и *day*. Каждой из них присваивается свое значение. Выражение apples = apples - eat_day * day сложное. Сначала выполняется подвыражение, стоящее справа от знака равенства. После этого его результат присваивается переменной *apples*, в результате чего ее старое значение (100) теряется. В подвыражении apples - eat_day * day вместо имен переменных на самом деле используются их значения, то есть числа 100, 5 и 7.

Ввод и вывод данных

Функцией print() отвечает за вывод данных, по умолчанию на экран. Если код содержится в файле, то без нее не обойтись. В интерактивном режиме в ряде случаев можно обойтись без нее.

Ввод данных в программу и их вывод важны в программировании. Без ввода программы делали бы одно и то же, исключая случаи, когда в них самих генерируются случайные значения. Вывод позволяет увидеть, использовать, передать дальше результат работы программы.

Обычно требуется, чтобы программа обрабатывала какой-то диапазон различных входных данных, которые поступают в нее из внешних источников. В качестве последних могут выступать файлы, клавиатура, сеть, выходные данные из другой программы. Вывод данных также возможен в файлы и др. Однако во многих случаях это происходит на экран монитора.

Можно сказать, что программа – это открытая система, которая обменивается чем-либо с внешней для нее средой. Если живой организм в основном обменивается веществом и энергией, то программа – данными, информацией.

Вывод данных. Функция print()

Что такое функция в программировании, узнаем позже. Пока будем считать, что print() – это такая команда языка Python, которая выводит то, что в ее скобках на экран.

```
>>> print(1032)
1032
>>> print(2.34)
2.34
>>> print("Hello")
Hello
```

В скобках могут быть любые типы данных. Кроме того, количество данных может быть различным:

```
>>> print("a:", 1)
```

```
a: 1
>>> one = 1
>>> two = 2
>>> three = 3
>>> print(one, two, three)
1 2 3
```

Можно передавать в функцию print() как непосредственно литералы (в данном случае "a:" и 1), так и переменные, вместо которых будут выведены их значения. Аргументы функции (то, что в скобках), разделяются между собой запятыми. В выводе вместо запятых значения разделены пробелом.

Если в скобках стоит выражение, то сначала оно выполняется, после чего print() уже выводит результат данного выражения:

```
>>> print("hello" + " " + "world")
hello world
>>> print(10 - 2.5/2)
8.75
```

В print() предусмотрены дополнительные параметры. Например, через параметр **sep** можно указать отличный от пробела разделитель строк:

```
>>> print("Mon", "Tue", "Wed", "Thu",
... "Fri", "Sat", "Sun", sep="-")
Mon-Tue-Wed-Thu-Fri-Sat-Sun
>>> print(1, 2, 3, sep="//")
1//2//3
```

Параметр end позволяет указывать, что делать, после вывода строки. По-умолчанию происходит переход на новую строку. Однако это действие можно отменить, указав любой другой символ или строку:

```
>>> print(10, end="")
10>>>
```

Обычно end используется не в интерактивном режиме, а в скриптах, когда несколько выводов подряд надо разделить не переходом на новую строку, а, скажем, запятыми. Сам переход на новую строку обозначается символом '\n'. Если присвоить это значение параметру end, то никаких изменений в работе функции print вы не увидите, так как это значение и так присвоено по умолчанию:

```
>>> print(10, end='\n')
10
>>>
```

Однако, если надо отступить на одну дополнительную строку после вывода, то можно сделать так:

```
>>> print(10, end='\n\n')
10
>>>
```

Следующее, что стоит рассказать о функции print – это использование форматирования строк. На самом деле оно никакого отношения к print не имеет, а применяется к строкам. Однако часто форматирование используется в сочетании с вызовом функции print().

Форматирование может выполняться в так называемом старом стиле или с помощью строкового метода format. Старый стиль также называют Си-стилем, так как он схож с тем, как происходит вывод на экран в языке C. Рассмотрим пример:

```
>>> pupil = "Ben"
>>> old = 16
>>> grade = 9.2
>>> print("It's %s, %d. Level: %f" %
... (pupil, old, grade))
It's Ben, 16. Level: 9.200000
```

Здесь вместо трех комбинаций символов %s, %d, %f подставляются значения переменных *pupil*, *old*, *grade*. Буквы s, d, f обозначают типы данных

– строку, целое число, вещественное число. Если бы требовалось подставить три строки, то во всех случаях использовалось бы сочетание %s.

Хотя в качестве значения переменной *grade* было указано число 9.2, на экран оно вывелось с дополнительными нулями. Однако мы можем указать, сколько требуется знаков после запятой, записав перед буквой **f** точку с желаемым числом знаков в дробной части:

```
>>> print("It's %s, %d. Level: %.1f"
...  % (pupil, old, grade))
It's Ben, 16. Level: 9.2
```

Теперь посмотрим на метод format():

```
>>> print("This is a {0}. It's {1}."
... .format("ball", "red"))
This is a ball. It's red.
>>> print("This is a {0}. It's {1}."
... .format("cat", "white"))
This is a cat. It's white.
>>> print("This is a {0}. It's {1} {2}."
... .format(1, "a", "number"))
This is a 1. It's a number.
```

В строке в фигурных скобках указаны номера данных, которые будут сюда подставлены. Далее к строке применяется метод format(). В его скобках указываются сами данные (можно использовать переменные). На нулевое место подставится первый аргумент метода format(), на место с номером 1 – второй и т. д.

На самом деле возможности метода format() существенно шире, и для их изучения понадобился бы отдельный урок. Нам пока будет достаточно этого.

Ввод данных. Функция input()

За ввод в программу данных с клавиатуры в Python отвечает функция input. Когда вызывается эта функция, программа останавливает свое

выполнение и ждет, когда пользователь введет текст. После этого, когда он нажмет Enter, функция input() заберет введенный текст и передаст его программе, которая уже будет обрабатывать его согласно своим алгоритмам.

Если в интерактивном режиме ввести команду input(), то ничего интересного вы не увидите. Компьютер будет ждать, когда вы что-нибудь введете и нажмете Enter или просто нажмете Enter. Если вы что-то ввели, это сразу же отобразиться на экране:

```
>>> input()
Yes!
'Yes!'
```

Функция input() передает введенные данные в программу. Их можно присвоить переменной. В этом случае интерпретатор не выводит строку сразу же:

```
>>> answer = input()
No, it is not.
```

В данном случае строка сохраняется в переменной *answer*, и при желании мы можем вывести ее значение на экран:

```
>>> answer
'No, it is not.'
```

При использовании функции print() кавычки в выводе опускаются:

```
>>> print(answer)
No, it is not.
```

Куда интересней использовать функцию input() в скриптах – файлах с кодом. Рассмотрим такую программу (рис.4.2.3).

При запуске программы, компьютер ждет, когда будет введена сначала одна строка, потом вторая. Они будут присвоены переменным *nameUser* и *cityUser*. После этого значения этих переменных выводятся на экран с помощью форматированного вывода.

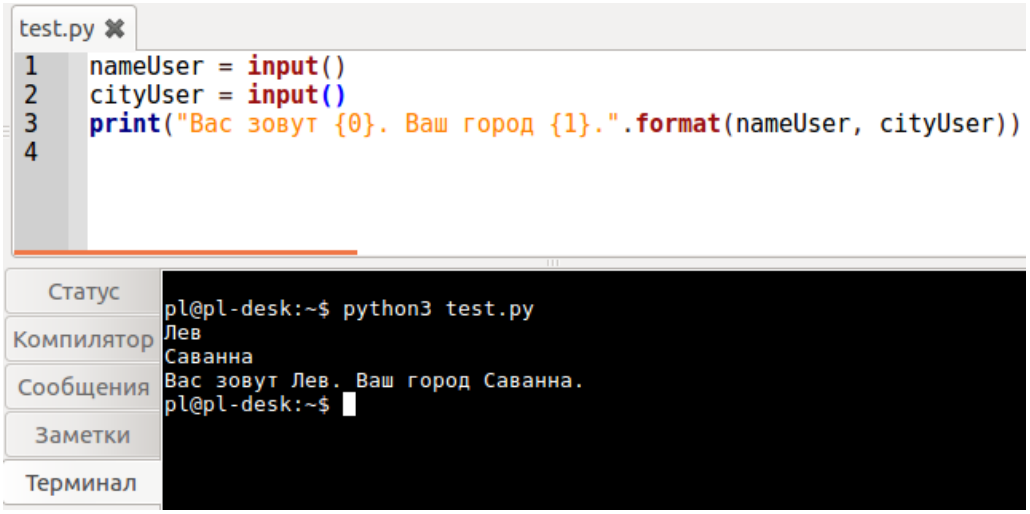


Рис. 4.2.3. Использование функции input()

Вышеприведенный скрипт далек от совершенства. Откуда пользователю знать, что хочет от него программа? Чтобы не вводить человека в замешательство, для функции input предусмотрен специальный параметр-приглашение. Это приглашение выводится на экран при вызове input(). Усовершенствованная программа может выглядеть так (рис.4.2.4):

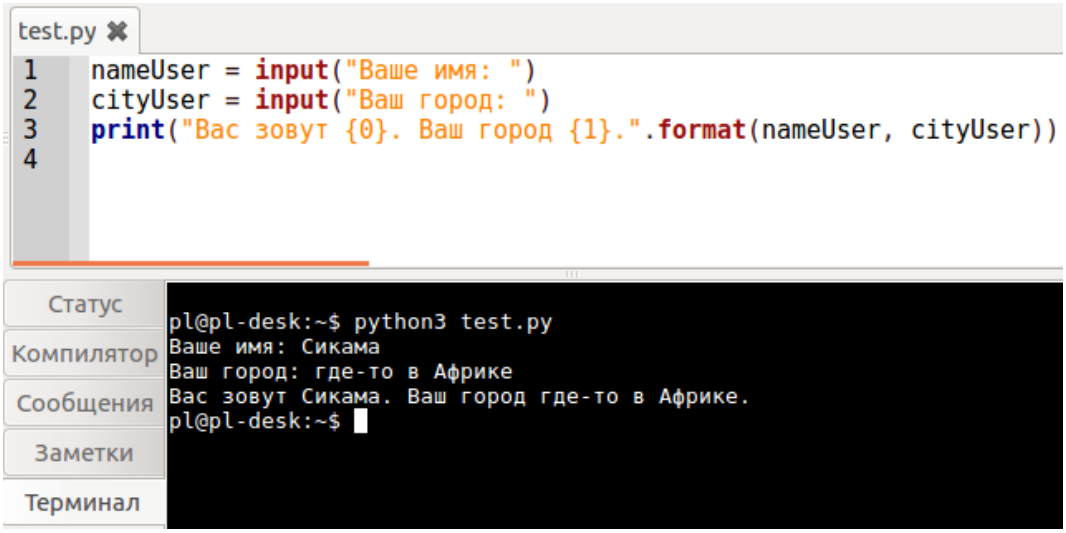


Рис. 4.2.4. Программирование с использованием функции input

Обратите внимание, что в программу поступает строка. Даже если ввести число, функция input() все равно вернет его строковое представление. Но что делать, если надо получить число? Ответ: использовать функции преобразования типов (рис.4.2.5).

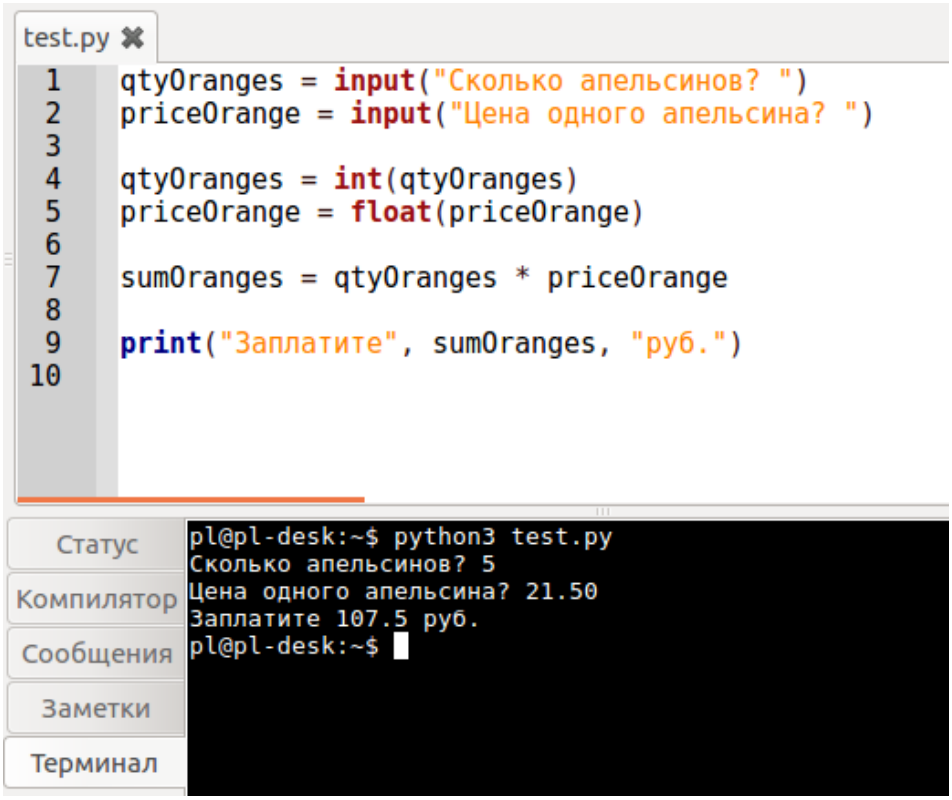


Рис.4.2.5. Программирование с использованием функции input

В данном случае с помощью функций int() и float() строковые значения переменных *qtyOranges* и *priceOrange* преобразуются соответственно в целое число и вещественное число. После этого новые численные значения присваиваются тем же переменным.

Программный код можно сократить, если преобразование типов выполнить в тех же строках кода, где вызывается функция input():

```
qtyOranges = int(input("Сколько апельсинов? "))
priceOrange = float(input("Цена одного? "))
sumOranges = qtyOranges * priceOrange
print("Заплатите", sumOranges, "руб.")
```

Сначала выполняется функция input(). Она возвращает строку, которую функция int() или float() сразу преобразует в число. Только после этого происходит присваивание переменной, то есть она сразу получает численное значение.

Вопросы для самоконтроля

1. Основные особенности языка Python
2. Типы данных. Переменные
3. Операции в программировании
4. Изменение типов данных
5. Ввод и вывод данных
6. Ввод данных. Функция input()

4.3. Проектирование и программирование линейных, разветвляющих, циклических процессов на языке программирования PYTHON.

Основные модули

- Логические выражения
- Логические операторы
- Сложные логические выражения
- Ветвление. Условный оператор
- Циклы в программировании.
- Цикл while
- Цикл for
- Функция range()

Логические выражения и операторы

Часто в реальной жизни мы соглашаемся с каким-либо утверждением или отрицаем его. Например, если вам скажут, что сумма чисел 3 и 5 больше 7, вы согласитесь, скажете: «Да, это правда». Если же кто-то будет утверждать, что сумма трех и пяти меньше семи, то вы расцените такое утверждение как ложное.

Подобные фразы предполагают только два возможных ответа – либо "да", когда выражение оценивается как правда, истина, либо "нет", когда утверждение оценивается как ошибочное, ложное. В программировании и

математике если результатом вычисления выражения может быть лишь истина или ложь, то такое выражение называется логическим.

Например, выражение 4>5 является логическим, так как его результатом является либо правда, либо ложь. Выражение 4 + 5 не является логическим, так как результатом его выполнения является число.

Ознакомимся с четвертым логическим типом данных (тип bool). Его также называют булевым. У этого типа всего два возможных значения: True (правда) и False (ложь).

```
>>> a = True
>>> type(a)
<class 'bool'>
>>> b = False
>>> type(b)
<class 'bool'>
```

Здесь переменной *a* было присвоено значение True, после чего с помощью встроенной в Python функции type() проверен ее тип. Интерпретатор сообщил, что это переменная класса bool. Понятия "класс" и "тип данных" в данном случае одно и то же. Переменная *b* также связана с булевым значением.

В программировании False обычно приравнивают к нулю, а True – к единице. Чтобы в этом убедиться, можно преобразовать булево значение к целочисленному типу:

```
>>> int(True)
1
>>> int(False)
0
```

Возможно и обратное. Можно преобразовать какое-либо значение к булевому типу:

```
>>> bool(3.4)
```

```
True
>>> bool(-150)
True
>>> bool(0)
False
>>> bool(' ')
True
>>> bool('')
False
```

И здесь работает правило: всё, что не 0 и не пустота, является правдой.

Логические операторы

Говоря на естественном языке (например, русском) мы обозначаем сравнения словами "равно", "больше", "меньше". В языках программирования используются специальные знаки, подобные тем, которые используются в математике: > (больше), < (меньше), >= (больше или равно), <= (меньше или равно), == (равно), != (не равно).

Не путайте операцию присваивания значения переменной, обозначаемую в языке Python одиночным знаком "равно", и операцию сравнения (два знака "равно"). Присваивание и сравнение – разные операции.

```
>>> a = 10
>>> b = 5
>>> a + b > 14
True
>>> a < 14 - b
False
>>> a <= b + 5
True
>>> a != b
True
>>> a == b
```

```
False
>>> c = a == b
>>> a, b, c
(10, 5, False)
```

В данном примере выражение c = a == b состоит из двух подвыражений. Сначала происходит сравнение (==) переменных *a* и *b*. После этого результат логической операции присваивается переменной *c*. Выражение a, b, c просто выводит значения переменных на экран.

Сложные логические выражения

Логические выражения типа kByte >= 1023 являются простыми, так как в них выполняется только одна логическая операция. Однако, на практике нередко возникает необходимость в более сложных выражениях. Может понадобиться получить ответа "Да" или "Нет" в зависимости от результата выполнения двух простых выражений. Например, "на улице идет снег или дождь", "переменная *news* больше 12 и меньше 20".

В таких случаях используются специальные операторы, объединяющие два и более простых логических выражения. Широко используются два оператора – так называемые логические И (and) и ИЛИ (or).

Чтобы получить True при использовании оператора and, необходимо, чтобы результаты обоих простых выражений, которые связывает данный оператор, были истинными. Если хотя бы в одном случае результатом будет False, то и все сложное выражение будет ложным.

Чтобы получить True при использовании оператора or, необходимо, чтобы результат хотя бы одного простого выражения, входящего в состав сложного, был истинным. В случае оператора or сложное выражение становится ложным лишь тогда, когда ложны оба составляющие его простые выражения.

Допустим, переменной *x* было присвоено значение 8 (x=8), переменной *y* присвоили 13 (y=13). Логическое выражение y<15 and x>8 будет выполняться следующим образом. Сначала выполнится

выражение y<15. Его результатом будет True. Затем выполнится выражение x > 8. Его результатом будет False. Далее выражение сведется к True and False, что вернет False.

```
>>> x = 8
>>> y = 13
>>> y < 15 and x > 8
False
```

Если бы мы записали выражение так: x>8 and y<15, то оно также вернуло бы False. Однако сравнение y<15 не выполнялось бы интерпретатором, так как его незачем выполнять. Ведь первое простое логическое выражение (x>8) уже вернуло ложь, которая, в случае оператора and, превращает все выражение в ложь.

В случае с оператором or второе простое выражение проверяется, если первое вернуло ложь, и не проверяется, если уже первое вернуло истину. Так как для истинности всего выражения достаточно единственного True, неважно по какую сторону от or оно стоит.

```
>>> y < 15 or x > 8
True
```

В языке Python есть еще унарный логический оператор not, то есть отрицание. Он превращает правду в ложь, а ложь в правду. Унарный он потому, что применяется к одному выражению, стоящему после него, а не справа и слева от него как в случае бинарных and и or.

```
>>> not y < 15
False
```

Здесь y < 15 возвращает True. Отрицая это, мы получаем False.

```
>>> a = 5
>>> b = 0
>>> not a
False
>>> not b
```

True

Число 5 трактуется как истина, отрицание истины дает ложь. Ноль приравнивается к False. Отрицание False дает True.

Ветвление. Условный оператор

Ход выполнения программы может быть *линейным*, то есть таким, когда выражения выполняются друг за другом, начиная с первого и заканчивая последним. Ни одна строка кода программы не пропускается.

Однако чаще в программах бывает не так. При выполнении кода, в зависимости от тех или иных условий, некоторые его участки могут быть опущены, в то время как другие – выполнены. Иными словами, в программе может присутствовать *ветвление*, которое реализуется условным оператором – особой конструкцией языка программирования.

Проведем аналогию с реальностью. Человек живет по расписанию. Можно сказать, расписание – это алгоритм для человека, его программный код, подлежащий выполнению. В расписании на 18.00 стоит поход в бассейн. Однако экземпляр биоробота класса Homo sapiens через свои рецепторы-сенсоры получает информацию, что воду из бассейна слили. Разумно было бы отменить занятие по плаванию, то есть изменить ход выполнения программы-расписания. Одним из условий посещения бассейна должно быть его функционирование, иначе должны выполняться другие действия.

Подобная нелинейность действий может быть реализована в компьютерной программе. Например, часть кода будет выполняться лишь при определенном значении конкретной переменной. В языках программирования используется приблизительно такая конструкция условного оператора:

```
if логическое_выражение {
    выражение 1;
    выражение 2;
    …
```

}

Перевести на человеческий язык можно так: если логическое выражение возвращает истину, то выполняются выражения внутри фигурных скобок; если логическое выражение возвращает ложь, то код внутри фигурных скобок не выполняется. С английского "if" переводится как "если".

Конструкция if логическое выражение называется заголовком условного оператора. Выражения внутри фигурных скобок – телом условного оператора. Тело может содержать как множество выражений, так и всего одно или даже быть пустым.

Пример использования условного оператора в языке программирования Python:

```
if n < 100:
 b = n + a
```

В Питоне вместо фигурных скобок используется двоеточие. Обособление вложенного кода, то есть тела оператора, достигается за счет отступов. В программировании принято делать отступ равным четырем пробелам. Однако также можно использовать клавишу табуляции (Tab) на клавиатуре. Большинство сред программирования автоматически делают отступ, как только вы поставите двоеточие и перейдете на новую строку. Однако при работе в интерактивном режиме надо делать отступы вручную (рис. 4.3.1).

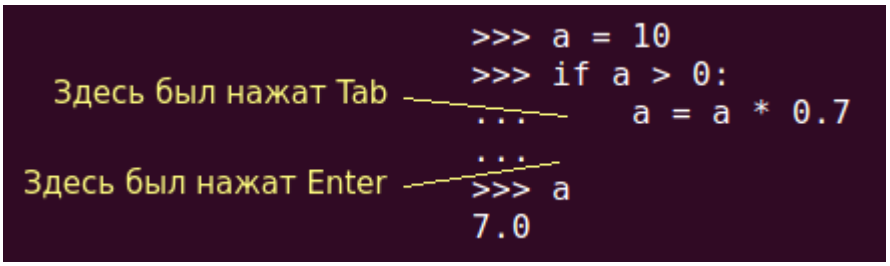


Рис. 4.3.1. Условный оператор

Нахождение в теле условного оператора здесь обозначается тремя точками. При создании файла со скриптом таких точек быть не должно, как и приглашения >>>.

Python считается языком с ясным синтаксисом и легко читаемым кодом. Это достигается сведением к минимуму таких вспомогательных элементов как скобок и точек с запятой. Для разделения выражений используется переход на новую строку, а для обозначения вложенных выражений – отступы от начала строки. В других языках данный стиль программирования также используется, но лишь для удобочитаемости кода человеком. В Питоне же он возведен в ранг синтаксического правила.

В примере выше логическим выражением является `n < 100`. Если оно возвращает истину, то выполнится строчка кода `b = n + a`. Если логическое выражение ложно, то выражение `b = n + a` не выполнится.

Данный пример вырван из контекста и сам по-себе не является рабочим. Полная версия программы могла бы выглядеть так:

```
b = 0
a = 50
n = 98
if n < 100:
b = n + a
print(b)
```

Последняя строчка кода `print(b)` уже не относится к условному оператору, что обозначено отсутствием перед ней отступа. Она не является вложенной в условный оператор, значит, не принадлежит ему.

Поскольку переменная *n* равна 98, а это меньше 100, то *b* станет равной 148-ми. Это значение будет выведено на экран. Если переменная *n* изначально была бы связана, например, со значением 101, то на экран был бы выведен 0. При *n*, равной 101, логическое выражение в заголовке условного оператора вернуло бы ложь. Значит, тело не было бы выполнено, и переменная *b* не изменилась бы.

Основная ветка программы выполняется всегда, а вложенный код лишь тогда, когда в темно-зеленой строчке, обозначающей заголовок условного оператора, случается истина.

Структуру программы можно изобразить следующим образом (рис. 4.3.2):

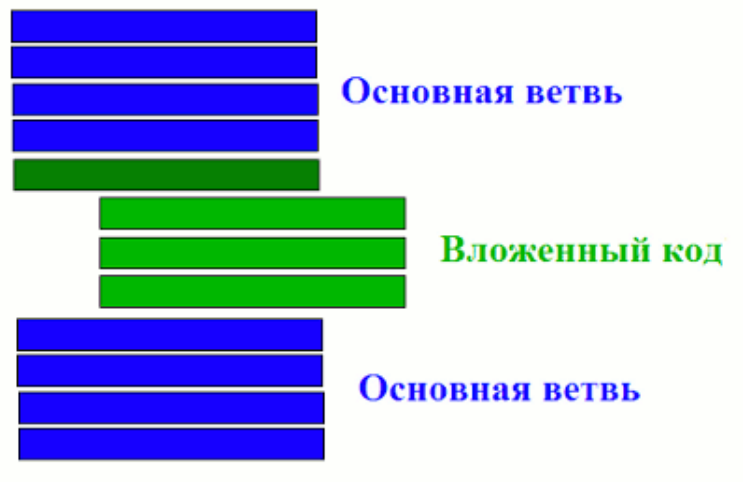


Рис. 4.3.2. Структура программы со вложенным кодом

Для небольших программ иногда чертят так называемые блок-схемы, отражающие алгоритм выполнения. В языке блок-схем определенные конструкции обозначаются своими фигурами. Так блок действий обозначается прямоугольником, а логическое выражение – ромбом. Для кода выше блок-схема может выглядеть так (рис. 4.3.3):

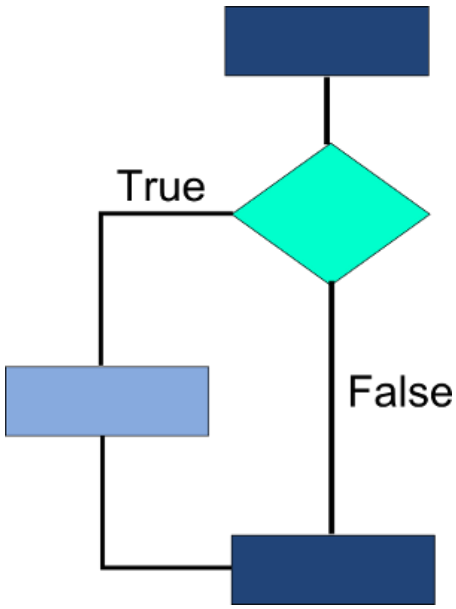


Рис. 4.3.3. Блок-схема логическое выражение

Условный оператор может включать не одну ветку, а две, реализуя тем самым полноценное ветвление.

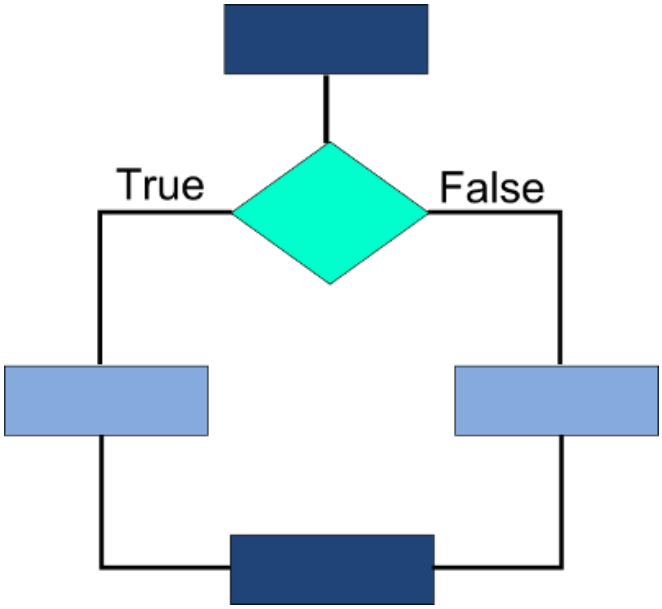


Рис. 4.3.4. Блок-схема логическое выражение

В случае возврата логическим выражением False поток выполнения программы не возвращается сразу в основную ветку. На случай False существует другой вложенный код, отличный от случая True (рис.4.3.4). Другими словами, встретившись с расширенной версией условного оператора, поток выполнения программы не вернется в основную ветку, не выполнив хоть какой-нибудь вложенный код.

В языках программирования разделение на две ветви достигается с помощью добавления блока else, получается так называемое if–else (если-иначе). Синтаксис выглядит примерно так:

```
if логическое_выражение {
    выражение 1;
    выражение 2;
    …
}
else {
    выражение 3;
```

```
    …
}
```

Если условие при инструкции if оказывается ложным, то выполняется блок кода при инструкции else. Ситуация, при которой бы выполнились обе ветви, невозможна. Либо код, принадлежащий if, либо код, принадлежащий else. Никак иначе. В заголовке else никогда не бывает логического выражения.

Пример кода с веткой else на языке программирования Python:

```
tovar1 = 50
tovar2 = 32
if tovar1 + tovar2 > 99 :
    print("99 рублей недостаточно")
else:
    print("Чек оплачен")
```

Следует иметь в виду, что логическое выражение при if может выглядеть "нестандартно", то есть не так просто, как a > b и тому подобное. Там может стоять просто одна переменная, число, слово True или False, а также сложное логическое выражение, когда два простых соединяются через логически И или ИЛИ.

```
a = ?
if a:
    a = 1
```

Если вместо знака вопроса будет стоять 0, то с логической точки зрения это False, значит выражение в if не будет выполнено. Если a будет связано с любым другим числом, то оно будет расцениваться как True, и тело условного оператора выполнится. Другой пример:

```
a = 5 > 0
if a:
    print(a)
```

Здесь *a* уже связана с булевым значением. В данном случае это True. Отметим, что в выражении a = 5 > 0 присваивание выполняется после оператора сравнения, так что подвыражение 5 > 0 выполнится первым, после чего его результат будет присвоен переменной a. На будущее, если вы сомневаетесь в последовательности выполнения операторов, используйте скобки, например так: a = (5 > 0).

Третий пример:

```
if a > 0 and a < b:
    print(b - a)
```

Тут, чтобы вложенный код выполнился, *a* должно быть больше нуля и одновременно меньше *b*. Также в Питоне, в отличие от других языков программирования, позволительна такая сокращенная запись сложного логического выражения:

```
if 0 < a < b:
    print(b - a)
```

Множественное ветвление: if-elif-else

Ранее мы рассмотрели работу условного оператора if. С помощью его расширенной версии if-else можно реализовать две отдельные ветви выполнения. Однако алгоритм программы может предполагать выбор больше, чем из двух путей, например, из трех, четырех или даже пяти. В данном случае следует говорить о необходимости множественного ветвления.

Рассмотрим конкретный пример. Допустим, в зависимости от возраста пользователя, ему рекомендуется определенный видеоконтент. При этом выделяют группы от 3 до 6 лет, от 6 до 12, от 12 до 16, 16+. Итого 4 диапазона. Как бы мы стали реализовывать задачу, имея в наборе инструментов только конструкцию if-else?

Самый простой ответ – последовательно проверять вхождение введенного числа-возраста в определенный диапазон с помощью следующих друг за другом условных операторов:

```
old = int(input('Ваш возраст: '))
 print('Рекомендовано:', end=' ')

if 3 <= old < 6:
    print('"Заяц в лабиринте"')
 if 6 <= old < 12:
    print('"Марсианин"')
 if 12 <= old < 16:
    print('"Загадочный остров"')
 if 16 <= old:
    print('"Поток сознания"')
```

Примечание. Названия фильмов выводятся на экран в двойных кавычках. Поэтому в программе для определения строк используются одинарные.

Предложенный код прекрасно работает, но есть одно существенное "но". Он не эффективен, так как каждый if в нем – это отдельно взятый оператор, никак не связанный с другими if. Процессор тратит время и "нервы" на обработку каждого из них, даже если в этом уже нет необходимости. Например, введено число 10. В первом if логическое выражение возвращает ложь, и поток выполнения переходит ко второму if. Логическое выражение в его заголовке возвращает истину, и его тело выполняется. Всё, на этом программа должна была остановиться.

Однако следующий if никак не связан с предыдущим, поэтому далее будет проверяться вхождение значения переменной old в диапазон от 12 до 16, в чем необходимости нет. И далее будет обрабатываться логическое выражение в последнем if, хотя уже понятно, что и там будет False. Что же делать?

Ответом является вложение условных операторов друг в друга:

```
old = int(input('Ваш возраст: '))
```

```
print('Рекомендовано:', end=' ')

if 3 <= old < 6:
    print('"Заяц в лабиринте"')
else:
    if 6 <= old < 12:
        print('"Марсианин"')
    else:
        if 12 <= old < 16:
            print('"Загадочный остров"')
        else:
            if 16 <= old:
                print('"Поток сознания"')
```

Рассмотрим поток выполнения этого варианта кода. Сначала проверяется условие в первом if (он же самый внешний). Если здесь было получено True, то тело этого if выполняется, а в ветку else мы даже не заходим, так как она срабатывает только тогда, когда в условии if возникает ложь.

Если внешний if вернул False, поток выполнения программы заходит в соответствующий ему внешний else. В его теле находится другой if со своим else. Если введенное число попадает в диапазон от 6 до 12, то выполнится тело вложенного if, после чего программа завершается. Если же число не попадает в диапазон от 6 до 12, то произойдет переход к ветке else. В ее теле находится свой условный оператор, имеющий уже третий уровень вложенности.

Таким образом до последней проверки (16 <= old) интерпретатор доходит только тогда, когда все предыдущие возвращают False. Если же по ходу выполнения программы возникает True, то все последующие проверки опускаются, что экономит ресурсы процессора. Кроме того, такая логика выполнения программы более правильная.

Теперь зададимся следующим вопросом. Можно ли как-то оптимизировать код множественного ветвления и не строить лестницу из вложенных друг в друга условных операторов? Во многих языках программирования, где отступы используются только для удобства чтения программистом, но не имеют никакого синтаксического значения, часто используется подобный стиль:

```
if логическое_выражение {
   … ;
}
else if логическое_выражение {
   … ;
}
else if логическое_выражение {
   … ;
}
else {
   … ;
}
```

Может показаться, что имеется только один уровень вложенности, и появляется новое расширение для if, выглядящее как else if. Но это только кажется. На самом деле if, стоящее сразу после else, является вложенным в это else. Выше приведенная схема – то же самое, что

```
if логическое_выражение {
   … ;
}
else
   if логическое_выражение {
      … ;
   }
   else
```

```
if логическое_выражение {
   … ;
}
else {
   … ;
}
```

Именно так ее "понимает" интерпретатор или компилятор. Однако считается, что человеку проще воспринимать первый вариант.

В Питоне подобный номер с поднятием вложенного if к более внешнему else не прокатит, потому что в нем отступы и переходы на новую строку имеют синтаксическое значение. Поэтому в язык Python встроена возможность настоящего множественного ветвления на одном уровне вложенности, которое реализуется с помощью веток elif.

Слово "elif" образовано от двух первых букв слова "else", к которым присоединено слово "if". Это можно перевести как "иначе если".

В отличие от else, в заголовке elif обязательно должно быть логическое выражение также, как в заголовке if. Перепишем нашу программу, используя конструкцию множественного ветвления:

```
old = int(input('Ваш возраст: '))
 print('Рекомендовано:', end=' ')
 if 3 <= old < 6:
    print('"Заяц в лабиринте"')
elif 6 <= old < 12:
    print('"Марсианин"')
elif 12 <= old < 16:
    print('"Загадочный остров"')
elif 16 <= old:
    print('"Поток сознания"')
```

Обратите внимание, в конце, после всех elif, может использоваться одна ветка else для обработки случаев, не попавших в условия ветки if и всех elif.

Блок-схему полной конструкции if-elif-…-elif-else можно изобразить так (рис.4.3.5):

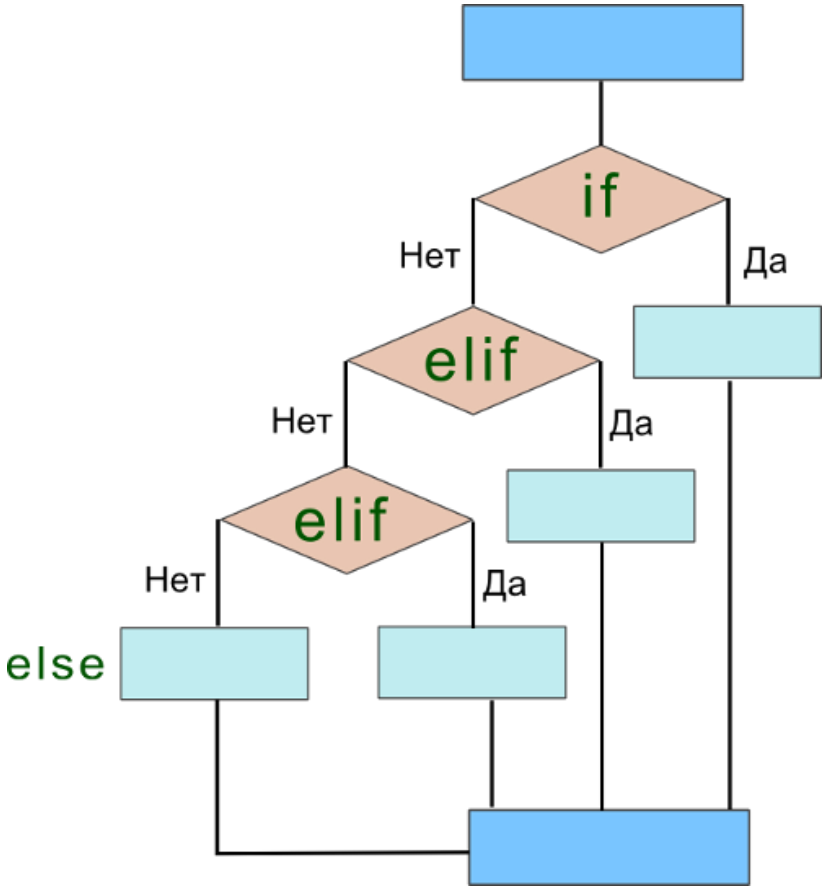


Рис.4.3.5. Блок-схема логическое выражение

Как только тело if или какого-нибудь elif выполняется, программа сразу же возвращается в основную ветку (нижний ярко-голубой прямоугольник), а все нижеследующие elif, а также else пропускаются.

Циклы в программировании. Цикл while

Циклы являются такой же важной частью структурного программирования, как условные операторы. С помощью циклов можно организовать повторение выполнения участков кода. Потребность в этом возникает довольно часто. Например, пользователь последовательно вводит числа, и каждое из них требуется добавлять к общей сумме. Или нужно вывести на экран квадраты ряда натуральных чисел и тому подобные задачи.

Цикл while

"While" переводится с английского как "пока". Но не в смысле "до свидания", а в смысле "пока имеем это, делаем то".

Можно сказать, **while** является универсальным циклом. Он присутствует во всех языках, поддерживающих структурное программирование, в том числе в Python. Его синтаксис обобщенно для всех языков можно выразить так:

```
while логическое_выражение {
   выражение 1;
   …
   выражение n;
}
```

Это похоже на условный оператор if. Однако в случае циклических операторов их тела могут выполняться далеко не один раз. В случае if, если логическое выражение в заголовке возвращает истину, то тело выполняется единожды. После этого поток выполнения программы возвращается в основную ветку и выполняет следующие выражения, расположенные ниже всей конструкции условного оператора.

В случае while, после того как его тело выполнено, поток возвращается к заголовку цикла и снова проверяет условие. Если логическое выражение возвращает истину, то тело снова выполняется. Потом снова возвращаемся к заголовку и так далее.

Цикл завершает свою работу только тогда, когда логическое выражение в заголовке возвращает ложь, то есть условие выполнения цикла больше не соблюдается. После этого поток выполнения перемещается к выражениям, расположенным ниже всего цикла. Говорят, "происходит выход из цикла".

Рассмотрите блок-схему цикла while (рис.4.3.6.). На ней ярко-голубыми прямоугольниками обозначена основная ветка программы, ромбом – заголовок цикла с логическим выражением, бирюзовым прямоугольником – тело цикла.

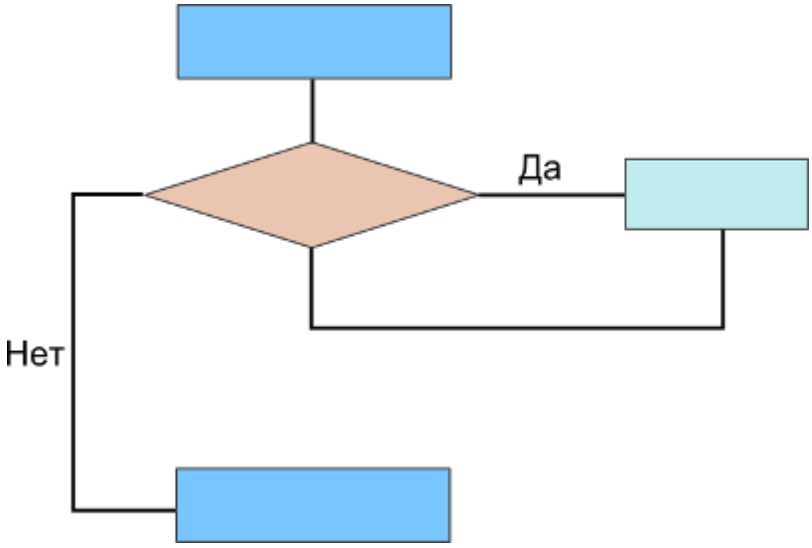


Рис. 4.3.6. Блок-схема цикла while

С циклом **while** возможны две исключительные ситуации:

Если при первом заходе в цикл логическое выражение возвращает False, то тело цикла не выполняется ни разу. Эту ситуацию можно считать нормальной, так как при определенных условиях логика программы может предполагать отсутствие необходимости в выполнении выражений тела цикла.

Если логическое выражение в заголовке while никогда не возвращает False, а всегда остается равным True, то цикл никогда не завершится, если только в его теле нет оператора принудительного выхода из цикла (break) или вызовов функций выхода из программы – quit(), exit() в случае Python. Если цикл повторяется и повторяется бесконечное количество раз, то в программе происходит **зацикливание**. В это время она зависает и самостоятельно завершиться не может.

Вспомним наш пример из урока про исключения. Пользователь должен ввести целое число. Поскольку функция input() возвращает строку, то программный код должен преобразовать введенное к целочисленному типу с помощью функции int(). Однако, если были введены символы, не

являющиеся цифрами, то возникает исключение ValueError, которое обрабатывается веткой except. На этом программа завершается.

Другими словами, если бы программа предполагала дальнейшие действия с числом (например, проверку на четность), а она его не получила, то единственное, что программа могла сделать, это закончить свою работу досрочно.

Но ведь можно просить и просить пользователя корректно вести число, пока он его не введет. Вот как может выглядеть реализующий это код:

```
n = input("Введите целое число: ")
 while type(n) != int:
    try:
        n = int(n)
    except ValueError:
        print("Неправильно ввели!")
        n = input("Введите целое число: ")
 if n % 2 == 0:
    print("Четное")
else:
    print("Нечетное")
```

Примечание 1. Не забываем, в языке программирования Python в конце заголовков сложных инструкций ставится двоеточие.

Примечание 2. В выражении type(n)!= int с помощью функции type() проверяется тип переменной n. Если он не равен int, то есть значение n не является целым числом, а является в данном случае строкой, то выражение возвращает истину. Если же тип n равен int, то данное логическое выражение возвращает ложь.

Примечание 3. Оператор % в языке Python используется для нахождения остатка от деления. Так, если число четное, то оно без остатка делится на 2, то есть остаток будет равен нулю. Если число нечетное, то остаток будет равен единице.

Проследим алгоритм выполнения этого кода. Пользователь вводит данные, они имеют строковый тип и присваиваются переменной *n*. В заголовке while проверяется тип n. При первом входе в цикл тип n всегда строковый, то есть он не равен int. Следовательно, логическое выражение возвращает истину, что позволяет зайти в тело цикла.

Здесь в ветке try совершается попытка преобразования строки к целочисленному типу. Если она была удачной, то ветка except пропускается, и поток выполнения снова возвращается к заголовку while.

Теперь *n* связана с целым числом, следовательно, ее тип int, который не может быть не равен int. Он ему равен. Таким образом логическое выражение type(n) != int возвращает False, и весь цикл завершает свою работу. Далее поток выполнения переходит к оператору **if-else**, находящемуся в основной ветке программы. Здесь могло бы находиться что угодно, не обязательно условный оператор.

Вернемся назад. Если в теле try попытка преобразования к числу была неудачной, и было выброшено исключение ValueError, то поток выполнения программы отправляется в ветку except и выполняет находящиеся здесь выражения, последнее из которых просит пользователя снова ввести данные. Переменная *n* теперь имеет новое значение.

После завершения except снова проверяется логическое выражение в заголовке цикла. Оно даст True, так как значение *n* по-прежнему строка.

Выход из цикла возможен только тогда, когда значение *n* будет успешно конвертировано в число.

Рассмотрим следующий пример:

```
total = 100
 i = 0
while i < 5:
    n = int(input())
    total = total - n
    i = i + 1
```

```
print("Осталось", total)
```

Сколько раз "прокрутится" цикл в этой программе, то есть сколько итераций он сделает? Ответ: 5.

Сначала переменная *i* равна 0. В заголовке цикла проверяется условие i < 5, и оно истинно. Тело цикла выполняется. В нем меняется значение *i*, путем добавления к нему единицы.

Теперь переменная *i* равна 1. Это меньше пяти, и тело цикла выполняется второй раз. В нем *i* меняется, ее новое значение 2.

Два меньше пяти. Тело цикла выполняется третий раз. Значение *i* становится равным трем.

Три меньше пяти. На этой итерации *i* присваивается 4.

Четыре по прежнему меньше пяти. К *i* добавляется единица, и теперь ее значение равно пяти.

Далее начинается шестая итерация цикла. Происходит проверка условия i < 5. Но поскольку теперь оно возвращает ложь, то выполнение цикла прерывается, и его тело не выполняется.

"Смысловая нагрузка" данного цикла – это последовательное вычитание из переменной *total* вводимых чисел. Переменная *i* в данном случае играет только роль счетчика итераций цикла. В других языках программирования для таких случаев предусмотрен цикл for, который так и называется: "цикл со счетчиком". Его преимущество заключается в том, что в теле цикла не надо изменять переменную-счетчик, ее значение меняется автоматически в заголовке for.

В языке Python тоже есть цикл for. Но это не цикл со счетчиком. В Питоне он предназначен для перебора элементов последовательностей и других сложных объектов. Данный цикл и последовательности будут изучены в последующих уроках.

Для while наличие счетчика не обязательно. Представим, что надо вводить числа, пока переменная *total* больше нуля. Тогда код будет выглядеть так:

```
total = 100

while total > 0:
    n = int(input())
    total = total - n
 print("Ресурс исчерпан")
```

Сколько раз здесь выполнится цикл? Неизвестно, все зависит от вводимых значений. Поэтому у цикла со счетчиком известно количество итераций, а у цикла без счетчика – нет.

Самое главное для цикла while – чтобы в его теле происходили изменения значений переменных, которые проверяются в его заголовке, и чтобы хоть когда-нибудь наступил случай, когда логическое выражение в заголовке возвращает False. Иначе произойдет зацикливание.

Примечание 1. Не обязательно в выражениях total = total - n и i = i + 1 повторять одну и ту же переменную. В Python допустим сокращенный способ записи подобных выражений: total -= n и i += 1.

Примечание 2. При использовании счетчика он не обязательно должен увеличиваться на единицу, а может изменяться в любую сторону на любое значение. Например, если надо вывести числа кратные пяти от 100 до 0, то изменение счетчика будет таким i = i - 5, или i -= 5.

Примечание 3. Для счетчика не обязательно использовать переменную с идентификатором *i*. Можно назвать переменную-счетчик как угодно. Однако так принято в программировании, что счетчики обозначают именами *i* и *j* (иногда одновременно требуются два счетчика).

Цикл for

Цикл for в языке программирования Python предназначен для перебора элементов структур данных и некоторых других объектов. Это не цикл со счетчиком, каковым является for во многих других языках.

Что значит перебор элементов? Например, у нас есть список, состоящий из ряда элементов. Сначала берем из него первый элемент, затем второй,

потом третий и так далее. С каждым элементом мы выполняем одни и те же действия в теле for. Нам не надо извлекать элементы по их индексам и заботится, на каком из них список заканчивается, и следующая итерация бессмысленна. Цикл for сам переберет и определит конец.

```
>>> spisok = [10, 40, 20, 30]
>>> for element in spisok:
...     print(element + 2)
...
12
42
22
32
```

После ключевого слова for используется переменная под именем element. Имя здесь может быть любым. Нередко используют i. На каждой итерации цикла for ей будет присвоен очередной элемент из списка spisok. Так при первой прокрутке цикла идентификатор element связан с числом 10, на второй – с числом 40, и так далее. Когда элементы в spisok заканчиваются, цикл for завершает свою работу.

С английского "for" переводится как "для", "in" как "в". Перевести конструкцию с языка программирования на человеческий можно так: для каждого элемента в списке делать следующее (то, что в теле цикла).

В примере мы увеличивали каждый элемент на 2 и выводили его на экран. При этом сам список конечно же не изменялся:

```
>>> spisok
[10, 40, 20, 30]
```

Нигде не шла речь о перезаписи его элементов, они просто извлекались и использовались. Однако бывает необходимо изменить сам список, например, изменить значение каждого элемента в нем или только определенных, удовлетворяющих определенному условию. И тут без

переменной, обозначающей индекс элемента, в большинстве случаев не обойтись:

```
>>> i = 0
>>> for element in spisok:
...     spisok[i] = element + 2
...     i += 1
...
>>> spisok
[12, 42, 22, 32]
```

Но если мы вынуждены использовать счетчик, то выгода от использования цикла for не очевидна. Если знать длину списка, то почему бы не воспользоваться while. Длину можно измерить с помощью встроенной в Python функции len().

```
>>> i = 0
>>> while i < len(spisok):
...     spisok[i] = spisok[i] + 2
...     i = i + 1 # или i += 1
...
>>> spisok
[14, 44, 24, 34]
```

Кроме того, с циклом while мы избавились от переменной *element*.

Функция range()

Теперь пришло время познакомиться со встроенной в Python функцией range(). "Range" переводится как "диапазон". Она может принимать один, два или три аргумента. Их назначение такое же как у функции randrange() из модуля random. Если задан только один, то генерируются числа от 0 до указанного числа, не включая его. Если заданы два, то числа генерируются от первого до второго, не включая его. Если заданы три, то третье число – это шаг.

Однако, в отличие от randrange(), функция range() генерирует не одно случайное число в указанном диапазоне. Она вообще не генерирует случайные числа. Она генерирует последовательность чисел в указанном диапазоне. Так, range(5, 11) сгенерирует последовательность 5, 6, 7, 8, 9, 10. Однако это будет не структура данных типа "список". Функция range() производит объекты своего класса – диапазоны:

```
>>> a = range(-10, 10)
>>> a
range(-10, 10)
>>> type(a)
<class 'range'>
```

Несмотря на то, что мы не видим последовательности чисел, она есть, и мы можем обращаться к ее элементам:

```
>>> a[0]
-10
>>> a[5]
-5
>>> a[15]
5
>>> a[-1]
9
```

Хотя изменять их нельзя, так как, в отличие от списков, объекты range() относятся к группе неизменяемых:

```
>>> a[10] = 100
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'range' object does not support
item assignment
```

Цикл for и range()

Итак, зачем нам понадобилась функций range() в теме про цикл for? Дело в том, что вместе они образуют неплохой тандем. For как цикл перебора элементов, в отличие от while, позволяет не следить за тем, достигнут ли конец структуры. Не надо вводить счетчик для этого, изменять его и проверять условие в заголовке. С другой стороны, range() дает последовательность целых чисел, которые можно использовать как индексы для элементов того же списка.

```
>>> range(len(spisok))
range(0, 4)
```

Здесь с помощью функции len() измеряется длина списка. В данном случае она равна четырем. После этого число 4 передается в функцию range(), и она генерирует последовательность чисел от 0 до 3 включительно. Это как раз индексы элементов нашего списка.

Теперь "соединим" for и range():

```
>>> for i in range(len(spisok)):
...     spisok[i] += 2
...
>>> spisok
[16, 46, 26, 36]
```

В заголовке цикла for берутся элементы вовсе не списка, а объекта range. Список, элементы которого планируется перезаписывать, тут по-сути не фигурирует. Если заранее знать длину списка, то заголовок может выглядеть так: for i in range(4). То, как используется *i* в теле цикла, вопрос второй. Примечание. Вместо идентификатора *i* может быть любой другой.

Вопросы для самоконтроля

1. Логические выражения и операторы
2. Сложные логические выражения
3. Ветвление. Условный оператор
4. Циклы в программировании.

5. Цикл while
6. Цикл for
7. Функция range()

4.4. Работа с функциями и модулями на языке программирования PYTHON.

Основные модули

- Функции в программировании
- Определение функции. Оператор def
- Вызов функции
- Возврат значений из функции. Оператор return
- Параметры и аргументы функции
- Произвольное количество аргументов
- Встроенные функции
- Модули

Функции в программировании

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции.

Функции можно сравнить с небольшими программками, которые сами по себе, то есть автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы.

Существует множество встроенных в язык программирования функций. С некоторыми такими в Python мы уже сталкивались.

Это print(), input(), int(), float(), str(), type(). Код их тела нам не виден, он где-то "спрятан внутри языка". Нам же предоставляется только интерфейс – имя функции.

С другой стороны, программист всегда может определять свои функции. Их называют пользовательскими. В данном случае под "пользователем" понимают программиста, а не того, кто пользует программу. Разберемся, зачем нам эти функции, и как их создавать.

Предположим, надо три раза подряд запрашивать на ввод пару чисел и складывать их. С этой целью можно использовать цикл:

```
i = 0
while i < 3:
    a = int(input())
    b = int(input())
    print(a+b)
    i += 1
```

Однако, что если перед каждым запросом чисел, надо выводить надпись, зачем они нужны, и каждый раз эта надпись разная. Мы не можем прервать цикл, а затем вернуться к тому же циклу обратно. Придется отказаться от него, и тогда получится длинный код, содержащий в разных местах одинаковые участки:

```
print("Сколько бананов и ананасов для обезьян?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")

print("Сколько жуков и червей для ежей?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
```

```
print("Сколько рыб и моллюсков для выдр?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
```

Пример исполнения программы:

```
Сколько бананов и ананасов для обезьян?
15
5
Всего 20 шт.
Сколько жуков и червей для ежей?
50
12
Всего 62 шт.
Сколько рыб и моллюсков для выдр?
16
8
Всего 24 шт.
```

Внедрение функций позволяет решить проблему дублирования кода в разных местах программы. Благодаря им можно исполнять один и тот же участок кода не сразу, а только тогда, когда он понадобится.

Определение функции. Оператор def

В языке программирования Python функции определяются с помощью оператора def. Рассмотрим код:

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Это пример определения функции. Как и другие сложные инструкции вроде условного оператора и циклов функция состоит из заголовка и тела.

Заголовок оканчивается двоеточием и переходом на новую строку. Тело имеет отступ.

Ключевое слово def сообщает интерпретатору, что перед ним определение функции. За def следует имя функции. Оно может быть любым, также как и всякий идентификатор, например, переменная. В программировании весьма желательно давать всему осмысленные имена. Так в данном случае функция названа "посчитать Еду" в переводе на русский.

После имени функции ставятся скобки. В приведенном примере они пустые. Это значит, что функция не принимает никакие данные из вызывающей ее программы. Однако она могла бы их принимать, и тогда в скобках были бы указаны так называемые параметры.

После двоеточия следует тело, содержащее инструкции, которые выполняются при вызове функции. Следует различать определение функции и ее вызов. В программном коде они не рядом и не вместе. Можно определить функцию, но ни разу ее не вызвать. Нельзя вызвать функцию, которая не была определена. Определив функцию, но ни разу не вызвав ее, вы никогда не выполните ее тела.

Вызов функции

Рассмотрим полную версию программы с функцией:

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")

print("Сколько бананов и ананасов для обезьян?")
countFood()

print("Сколько жуков и червей для ежей?")
countFood()
```

```
print("Сколько рыб и моллюсков для выдр?")
countFood()
```

После вывода на экран каждого информационного сообщения осуществляется вызов функции, который выглядит просто как упоминание ее имени со скобками. Поскольку в функцию мы ничего не передаем скобки опять же пустые. В приведенном коде функция вызывается три раза.

Когда функция вызывается, поток выполнения программы переходит к ее определению и начинает исполнять ее тело. После того, как тело функции исполнено, поток выполнения возвращается в основной код в то место, где функция вызывалась. Далее исполняется следующее за вызовом выражение.

В языке Python определение функции должно предшествовать ее вызовам. Это связано с тем, что интерпретатор читает код строка за строкой и о том, что находится ниже по течению, ему еще неизвестно. Поэтому если вызов функции предшествует ее определению, то возникает ошибка (выбрасывается исключение `NameError`):

```
print("Сколько бананов и ананасов для обезьян?")
countFood()

print("Сколько жуков и червей для ежей?")
countFood()

print("Сколько рыб и моллюсков для выдр?")
countFood()

def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Результат:

```
Сколько бананов и ананасов для обезьян?
Traceback (most recent call last):
  File "test.py", line 2, in <module>
    countFood()
NameError: name 'countFood' is not defined
```

Для многих компилируемых языков это не обязательное условие. Там можно определять и вызывать функцию в произвольных местах программы. Однако для удобочитаемости кода программисты даже в этом случае предпочитают соблюдать определенные правила.

Польза функций не только в возможности многократного вызова одного и того же кода из разных мест программы. Не менее важно, что благодаря им программа обретает истинную структуру. Функции как бы разделяют ее на обособленные части, каждая из которых выполняет свою конкретную задачу.

Пусть надо написать программу, вычисляющую площади разных фигур. Пользователь указывает, площадь какой фигуры он хочет вычислить. После этого вводит исходные данные. Например, длину и ширину в случае прямоугольника. Чтобы разделить поток выполнения на несколько ветвей, следует использовать оператор if-elif-else:

```
figure = input("1-прямоугольник,
2-треугольник, 3-круг: ")

if figure == '1':
  a = float(input("Ширина: "))
  b = float(input("Высота: "))
  print("Площадь: %.2f" % (a*b))
elif figure == '2':
  a = float(input("Основание: "))
  h = float(input("Высота: "))
  print("Площадь: %.2f" % (0.5 * a * h))
elif figure == '3':
```

```
  r = float(input("Радиус: "))
  print("Площадь: %.2f" % (3.14 * r**2))
else:
  print("Ошибка ввода")
```

Здесь нет никаких функций, и все прекрасно. Но напишем вариант с функциями:

```
def rectangle():
    a = float(input("Ширина: "))
    b = float(input("Высота: "))
    print("Площадь: %.2f" % (a*b))

def triangle():
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    print("Площадь: %.2f" % (0.5 * a * h))

def circle():
    r = float(input("Радиус: "))
    print("Площадь: %.2f" % (3.14 * r**2))

figure = input("1-прямоугольник,
2-треугольник, 3-круг: ")
if figure == '1':
  rectangle()
elif figure == '2':
  triangle()
elif figure == '3':
  circle()
else:
  print("Ошибка ввода")
```

Он кажется сложнее, а каждая из трех функций вызывается всего один раз. Однако из общей логики программы как бы убраны и обособлены инструкции для нахождения площадей. Программа теперь состоит из отдельных "кирпичиков Лего". В основной ветке мы можем комбинировать их как угодно. Она играет роль управляющего механизма.

Если нам когда-нибудь захочется вычислять площадь треугольника по формуле Герона, а не через высоту, то не придется искать код во всей программе (представьте, что она состоит из тысяч строк кода как реальные программы). Мы пойдем к месту определения функций и изменим тело одной из них.

Если понадобиться использовать эти функции в какой-нибудь другой программе, то мы сможем импортировать их туда, сославшись на данный файл с кодом (как это делается в Python, будет рассмотрено позже).

Возврат значений из функции. Оператор return

Функции могут передавать какие-либо данные из своих тел в основную ветку программы. Говорят, что функция возвращает значение. В большинстве языков программирования, в том числе Python, выход из функции и передача данных в то место, откуда она была вызвана, выполняется оператором return.

Если интерпретатор Питона, выполняя тело функции, встречает return, то он "забирает" значение, указанное после этой команды, и "уходит" из функции.

```
def cylinder():
    r = float(input())
    h = float(input())
    # площадь боковой поверхности цилиндра:
    side = 2 * 3.14 * r * h
    # площадь одного основания цилиндра:
    circle = 3.14 * r**2
    # полная площадь цилиндра:
```

```
    full = side + 2 * circle
    return full

square = cylinder()
print(square)
```

Пример выполнения:

```
3
7
188.4
```

В данной программе в основную ветку из функции возвращается значение локальной переменной *full*. Не сама переменная, а ее значение, в данном случае – какое-либо число, полученное в результате вычисления площади цилиндра.

В основной ветке программы это значение присваивается глобальной переменной *square*. То есть выражение square = cylinder() выполняется так:

Вызывается функция cylinder().

Из нее возвращается значение.

Это значение присваивается переменной *square*.

Не обязательно присваивать результат переменной, его можно сразу вывести на экран:

```
...
print(cylinder())
```

Здесь число, полученное из cylinder(), непосредственно передается функции print(). Если мы в программе просто напишем cylinder(), не присвоив полученные данные переменной или не передав их куда-либо дальше, то эти данные будут потеряны. Но синтаксической ошибки не будет.

В функции может быть несколько операторов return. Однако всегда выполняется только один из них. Тот, которого первым достигнет поток выполнения. Допустим, мы решили обработать исключение, возникающее на

некорректный ввод. Пусть тогда в ветке except обработчика исключений происходит выход из функции без всяких вычислений и передачи значения:

```
def cylinder():
    try:
        r = float(input())
        h = float(input())
    except ValueError:
        return
    side = 2 * 3.14 * r * h
    circle = 3.14 * r**2
    full = side + 2 * circle
    return full
print(cylinder())
```

Более того. Ранее мы рассматривали функции, которые вроде бы не возвращали никакого значения, потому что в них не было оператора return. На самом деле возвращали, просто мы не обращали на него внимание, не присваивали никакой переменной и не выводили на экран. В Python всякая функция что-либо возвращает. Если в ней нет оператора return, то она возвращает None. То же самое, как если в ней имеется "пустой" return.

Параметры и аргументы функции

В программировании функции могут не только возвращать данные, но также принимать их, что реализуется с помощью так называемых параметров, которые указываются в скобках в заголовке функции. Количество параметров может быть любым.

Параметры представляют собой локальные переменные, которым присваиваются значения в момент вызова функции. Конкретные значения, которые передаются в функцию при ее вызове, будем называть аргументами. Следует иметь в виду, что встречается иная терминология. Например, формальные параметры и фактические параметры. В Python же обычно все называют аргументами.

Рассмотрим схему и поясняющий ее пример (рис.4.4.1):

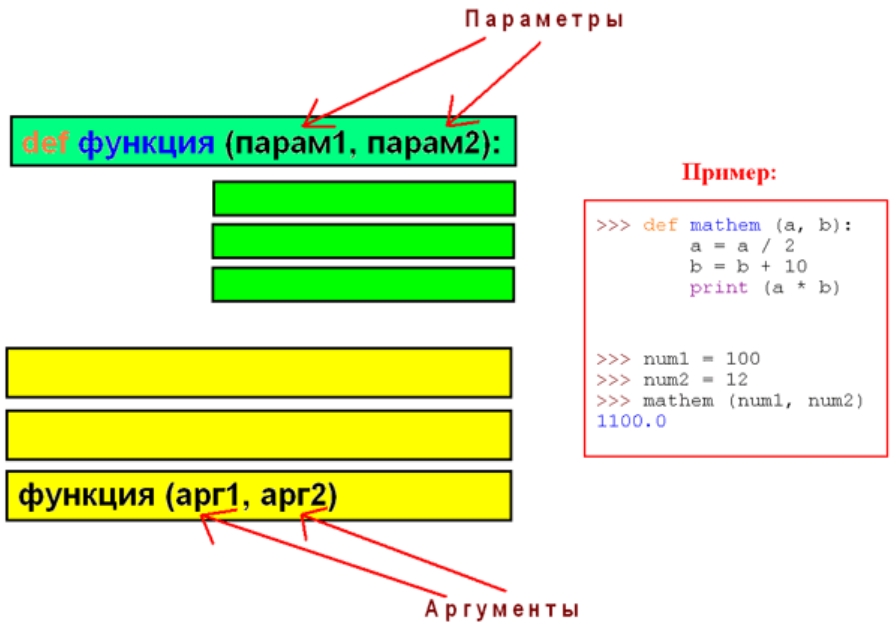


Рис.4.4.1. Параметры и аргументы функции

Когда функция вызывается, то ей передаются аргументы. В примере указаны глобальные переменные *num1* и *num2*. Однако на самом деле передаются не эти переменные, а их значения. В данном случае числа 100 и 12. Другими словами, мы могли бы писать mathem(100, 12). Разницы не было бы.

Когда интерпретатор переходит к функции, чтобы начать ее исполнение, он присваивает переменным-параметрам переданные в функцию значения-аргументы. В примере переменной *a* будет присвоено 100, *b* будет присвоено 12.

Изменение значений *a* и *b* в теле функции никак не скажется на значениях переменных *num1* и *num2*. Они останутся прежними. В Python такое поведение характерно для неизменяемых типов данных, к которым относятся, например, числа и строки. Говорят, что в функцию данные передаются по значению. Так, когда a присваивалось число 100, то это было уже другое число, не то, на которое ссылается переменная *num1*. Число 100 было скопировано и помещено в отдельную ячейку памяти для переменной *a*.

Существуют изменяемые типы данных. Для Питона, это, например, списки и словари. В этом случае данные передаются по ссылке. В функцию передается ссылка на них, а не сами данные. И эта ссылка связывается с локальной переменной. Изменения таких данных через локальную переменную обнаруживаются при обращении к ним через глобальную. Это есть следствие того, что несколько переменных ссылаются на одни и те же данные, на одну и ту же область памяти.

Необходимость передачи по ссылке связана в первую очередь с экономией памяти. Сложные типы данных, по сути представляющие собой структуры данных, обычно копировать не целесообразно. Однако, если надо, всегда можно сделать это принудительно.

Произвольное количество аргументов

Обратим внимание еще на один момент. Количество аргументов и параметров совпадает. Нельзя передать три аргумента, если функция принимает только два. Нельзя передать один аргумент, если функция требует два обязательных. В рассмотренном примере они обязательные.

Однако в Python у функций бывают параметры, которым уже присвоено значение по-умолчанию. В таком случае, при вызове можно не передавать соответствующие этим параметрам аргументы. Хотя можно и передать. Тогда значение по умолчанию заменится на переданное.

```
def cylinder(h, r = 1):
    side = 2 * 3.14 * r * h
    circle = 3.14 * r**2
    full = side + 2 * circle
    return full

figure1 = cylinder(4, 3)
figure2 = cylinder(5)
print(figure1)
print(figure2)
```

Вывод:

131.88

37.68

При втором вызове cylinder() мы указываем только один аргумент. Он будет присвоен переменной-параметру *h*. Переменная *r* будет равна 1.

Согласно правилам синтаксиса Python при определении функции параметры, которым присваивается значение по-умолчанию должны следовать (находиться сзади) за параметрами, не имеющими значений по умолчанию.

А вот при вызове функции, можно явно указывать, какое значение соответствует какому параметру. В этом случае их порядок не играет роли:

```
…
figure3 = cylinder(10, 2)
figure4 = cylinder(r=2, h=10)
print(figure3)
print(figure4)
```

В данном случае оба вызова – это вызовы с одними и теми же аргументами-значениями. Просто в первом случае сопоставление параметрам-переменным идет в порядке следования. Во-втором случае – по ключам, которыми выступают имена параметров.

В Python определения и вызовы функций имеют и другие нюансы, рассмотрение которых мы пока опустим, так как они требуют более глубоких знаний, чем у нас есть на данный момент. Скажем лишь, что функции может быть определена так, что в нее можно передать хоть ни одного аргумента, хоть множество:

```
def oneOrMany(*a):
    print(a)


oneOrMany(1)
oneOrMany('1',1, 2, 'abc')
```

oneOrMany()

Результат:

(1,)

('1', 1, 2, 'abc')

()

Опять же, судя по скобкам, здесь возникает упомянутый в прошлом уроке кортеж.

Встроенные функции

Язык Python включает много уже определенных, то есть встроенных в него, функций. Программист не видит их определений, они скрыты где-то в "недрах" языка. Достаточно знать, что эти функции принимают и что возвращают, то есть их интерфейс.

Ряд встроенных функций, касающихся ввода-вывода и типов данных, мы уже использовали. Это print(), input(), int(), float(), str(), bool(), type().

В этом уроке рассмотрим следующие встроенные функции, условно разбив их на группы:

функции для работы с символами – ord(), chr(), len() математические функции – abs(), round(), divmod(), pow(), max(), min(), sum()

Функция ord() позволяет получить номер символа по таблице Unicode. Соответственно, принимает она в качестве аргумента одиночный символ, заключенный в кавычки:

```
>>> ord('z')
122
>>> ord('ф')
1092
>>> ord('@')
64
```

Функция chr() выполняет обратное действие. Она позволяет получить символ по его номеру:

```
>>> chr(87)
```

```
'W'
>>> chr(1049)
'Й'
>>> chr(10045)
'❇'
```

Чтобы не путать ord() и chr(), помните, что функция – это действие. Ее имя как бы отвечает на вопрос "Что сделать?". Order – это порядок. Значит, мы хотим получить порядковый номер элемента в ряду. А чтобы получить номер, должны передать символ. Character – это символ. Значит, мы хотим получить символ. Поэтому должны передать порядковый номер.

Функция len() в качестве аргумента принимает объект, который состоит из более простых объектов, количество которых она подсчитывает. Числа – это простые объекты, их нельзя передавать в len(). Строки можно:

```
>>> len('abc')
3
>>> s1 = '------'
>>> s2 = '_______'
>>> len(s1) > len(s2)
False
>>> len(s1)
6
>>> len(s2)
7
```

Кроме строк в len() можно передавать другие, еще не изученные нами, структуры данных.

Функция abs() возвращает абсолютное значение числа:

```
>>> abs(-2.2)
2.2
>>> abs(9)
9
```

Если требуется округлить вещественное число до определенного знака после запятой, то следует воспользоваться функцией round():

```
>>> a = 10/3
>>> a
3.3333333333333335
>>> round(a,2)
3.33
>>> round(a)
3
```

Если второй аргумент не задан, то округление идет до целого числа. Есть одна специфическая особенность этой функции. Второй аргумент может быть отрицательным числом. В этом случае округляться начинают единицы, десятки, сотни и т. д., то есть целая часть:

```
>>> round(5321, -1)
5320
>>> round(5321, -3)
5000
>>> round(5321, -4)
10000
```

Функция именно округляет согласно правилу округления из математики, а не отбрасывает. Поэтому 5 тысяч неожиданно округляются до десяти.

```
>>> round(3.76, 1)
3.8
>>> round(3.72, 1)
3.7
>>> round(3.72)
4
>>> round(3.22)
3
```

Если нужно просто избавиться от дробной части без округления, следует воспользоваться функцией int():

```
>>> int(3.78)
3
```

Нередко функцию round() используют совместно с функцией print(), избегая форматирования вывода:

```
>>> a = 3.45673
>>> print("Number: %.2f" % a)
Number: 3.46
>>> print("Number:", round(a,2))
Number: 3.46
```

В последнем случае код выглядит более ясным.

Функция divmod() выполняет одновременно деление нацело и нахождение остатка от деления:

```
>>> divmod(10, 3)
(3, 1)
>>> divmod(20, 7)
(2, 6)
```

Возвращает она кортеж, извлечение данных из которого мы пока не изучали. В других языках нередко встречаются две отдельные функции: div() и mod(). Первая делит нацело, вторая находит остаток от целочисленного деления (деления по модулю). В Python и многих других языках для этого используются специальные символы-операнды:

```
>>> 10 // 3
3
>>> 10 % 3
1
```

Функция pow() возводит в степень. Первое число – основание, второе – показатель:

```
>>> pow(3, 2)
9
>>> pow(2, 4)
16
```

То же самое можно проделать так:

```
>>> 3**2
9
>>> 2**4
16
```

Однако pow() может принимать третий необязательный аргумент. Это число, на которое делится по модулю результат возведения в степень:

```
>>> pow(2, 4, 4)
0
>>> 2**4 % 4
0
```

Преимуществом первого способа является его более быстрое выполнение.

Функции max(), min() и sum() находят соответственно максимальный, минимальный элемент и сумму элементов аргумента:

```
>>> max(10, 12, 3)
12
>>> min(10, 12, 3, 9)
3
>>> a = (10, 12, 3, 10)
>>> sum(a)
35
```

В sum() нельзя передать перечень элементов, должна быть структура данных, например, кортеж. В min() и max() также чаще передают один так называемый итерируемый объект:

```
>>> max(a)
12
```

Модули

Встроенные в язык программирования функции доступны сразу. Чтобы их вызвать, не надо выполнять никаких дополнительных действий. Однако за время существования любого популярного языка на нем было написано столько функций и классов, которые оказались востребованными множеством программистов и в разных областях, что включить весь этот объем кода в сам язык если возможно, то нецелесообразно.

Чтобы разрешить проблему доступа к дополнительным возможностям языка, в программировании стало общепринятой практикой использовать так называемые модули, пакеты и библиотеки. Каждый модуль содержит коллекцию функций и классов, предназначенных для решения задач из определенной области. Так в модуле math языка Python содержатся математические функции, модуль random позволяет генерировать псевдослучайные числа, в модуле datetime содержатся классы для работы с датами и временем, модуль sys предоставляет доступ к системным переменным и т. д.

Количество модулей для языка Python огромно, что связано с популярностью языка. Часть модулей собрана в так называемую стандартную библиотеку. Стандартная она потому, что поставляется вместе с установочным пакетом. Однако существуют сторонние библиотеки. Они скачиваются и устанавливаются отдельно.

Для доступа к функционалу модуля, его надо импортировать в программу. После импорта интерпретатор "знает" о существовании дополнительных классов и функций и позволяет ими пользоваться.

В Питоне импорт осуществляется командой import. При этом существует несколько способов импорта. Рассмотрим работу с модулем на примере math. Итак,

```
>>> import math
```

Ничего не произошло. Однако в глобальной области видимости появилось имя math. Если до импорта вы упомянули бы имя math, то возникла бы ошибка NameError. Теперь же

```
>>> math
<module 'math' (built-in)>
```

В программе завелся объект math, относящийся к классу module.

Чтобы увидеть перечень функций, входящих в этот модуль, воспользуемся встроенной в Python функцией dir(), передав ей в качестве аргумента имя модуля:

```
>>> dir(math)
['__doc__', '__loader__', '__name__',
'__package__', '__spec__', 'acos', 'acosh',
'asin', 'asinh', 'atan', 'atan2', 'atanh',
'ceil', 'copysign', 'cos', 'cosh',
'degrees', 'e', 'erf', 'erfc', 'exp',
'expm1', 'fabs', 'factorial', 'floor',
'fmod', 'frexp', 'fsum', 'gamma', 'gcd',
 'hypot', 'inf', 'isclose', 'isfinite',
'isinf', 'isnan', 'ldexp', 'lgamma', 'log',
'log10', 'log1p', 'log2', 'modf',
'nan', 'pi', 'pow', 'radians', 'sin', 'sinh',
'sqrt', 'tan', 'tanh', 'trunc']
```

Проигнорируем имена с двойными подчеркиваниями. Все остальное – имена функций и констант (переменных, которые не меняют своих значений), включенных в модуль math. Чтобы вызвать функцию из модуля, надо впереди написать имя модуля, поставить точку, далее указать имя функции, после чего в скобках передать аргументы, если они требуются. Например, чтобы вызвать функцию pow из math, надо написать так:

```
>>> math.pow(2, 2)
4.0
```

Обратите внимание, эта другая функция pow(), не та, что встроена в сам язык. "Обычная" функция pow() возвращает целое, если аргументы целые числа:

```
>>> pow(2, 2)
4
```

Для обращения к константе скобки не нужны:

```
>>> math.pi
3.141592653589793
```

Если мы не знаем, что делает та или иная функция, то можем получить справочную информацию о ней с помощью встроенной в язык Python функции help():

```
>>> help(math.gcd)
Help on built-in function gcd in module math:

gcd(...)
    gcd(x, y) -> int
    greatest common divisor of x and y
```

Для выхода из интерактивной справки надо нажать клавишу q. В данном случае сообщается, что функция возвращает целое число, и это наибольший общий делитель для чисел *x* и *y*. Описание модулей и их содержания также можно посмотреть в официальной документации на сайте python.org.

Второй способ импорта – это когда импортируется не сам модуль, а только необходимые функции из него.

```
>>> from math import gcd, sqrt, hypot
```

Перевести можно как "из модуля math импортировать функции gcd, sqrt и hypot".

В таком случае при их вызове не надо перед именем функции указывать имя модуля:

```
>>> gcd(100, 150)
50
```

```
>>> sqrt(16)
4.0
>>> hypot(3, 4)
5.0
```

Чтобы импортировать сразу все функции из модуля:

```
>>> from math import *
```

Импорт через from не лишен недостатка. В программе уже может быть идентификатор с таким же именем, как имя одной из импортируемых функций или констант. Ошибки не будет, но одно из них окажется "затерто":

```
>>> pi = 3.14
>>> from math import pi
>>> pi
3.141592653589793
```

Здесь исчезает значение 3.14, присвоенное переменной *pi*. Это имя теперь указывает на число из модуля math. Если импорт сделать раньше, чем присвоение значения *pi*, то будет все наоборот:

```
>>> from math import pi
>>> pi = 3.14
>>> pi
3.14
```

В этой связи более опасен именно импорт всех функций. Так как в этом случае очень легко не заметить подмены значений идентификаторов.

Однако можно изменить имя идентификатора из модуля на какое угодно:

```
>>> from math import pi as P
>>> P
3.141592653589793
>>> pi
3.14
```

В данном случае константа pi из модуля импортируется под именем *P*. Смысл подобных импортов в сокращении имен, так как есть модули с длинными именами, а имена функций и классов в них еще длиннее. Если в программу импортируется всего пара сущностей, и они используются в ней часто, то имеет смысл переименовать их на более короткий вариант. Сравните:

```
>>> import calendar
>>> calendar.weekheader(2)
'Mo Tu We Th Fr Sa Su'
```

и

```
>>> from calendar import weekheader as week
>>> week(3)
'Mon Tue Wed Thu Fri Sat Sun'
```

Во всех остальных случаях лучше оставлять идентификаторы содержимого модуля в пространстве имен самого модуля и получать доступ к ним через имя модуля, то есть выполнять импорт командой import имя_модуля, а вызывать, например, функции через имя_модуля.имя_функции().

Вопросы для самоконтроля

1. Функции в программировании
2. Определение функции. Оператор def
3. Вызов функции
4. Возврат значений из функции. Оператор return
5. Параметры и аргументы функции
6. Произвольное количество аргументов
7. Встроенные функции
8. Модули

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Discovering Computers 2016. Tools, Apps, Devices, and the Impact of Texnology. 691 pg.
2. Richard L. Halterman Fundamentals of Ci++ Programming. Copyright © 2008–2016. All rights reserved. 634 pg.
3. Brian P. Hogan HTML5 and CSS3, Second Edition. Level Up with Today's Web Technologies. Copyright © 2013 The Pragmatic Programmers, LLC. All rights reserved. 290 pg.
4. Raavi O'Connor Autodesk 3ds Max® 2016 Modeling and Shading Essentials. Copyright © 2015 Raavi Design. 466 pg.

Дополнительная литература

5. Mirziyoev SH.M. Tashkiliy tahlil, qa`tiy tartib-intizom va shaxsiy javobgarlik-xar bir raxbar faoliyatining kundalik qoidasi bo'lishi kerak. T., "O'zbekiston". 2017 y. 102 bet.
6. Mirziyoev SH.M. Qonun ustuvorligi va inson manfaatlarini ta`minlash – yurt taraqqiyoti va xalq faravonligining garofi. T., "O'zbekiston". 2016 y. 47 bet.
7. Mirziyoev SH.M. Buyuk kelajagimizni mard va olijanob xalqimiz bilan birga quramiz. T., "O'zbekiston". 2016 y. 48 bet.
8. Randy H. Shih AutoCAD 2017 Tutorial - First Level: 2D Fundamentals Better Textbooks. Lower Prices.
9. .BarBaraZukinHeiman. PH.D. and others Practical Photoshop® CS6, Level 1 Copyright © 2009–2017 by. 53 pg.
10. М.Арипов. Информационные технологии. Учебное пособие Т.: "Noshir" 2009. 366-с.
11. М.М.Арипов. Информатика, Информационные технологии. Учебник Т.: TDYUI 2005. 278-с.

12.С.С.Косисов. Информационные технологии: Учебник для высших учебных заведений. Т.: Alokachi, 2006. – 360 b.

Internet сайты

13. www.uz – Национальная поисковая система

14. www.gov.uz – Правительственный портал Республики Узбекистан

15. www.ZiyoNET.uz – Информационно-образовательный портал Республики Узбекистан

16. www.e-darslik.uz

17. http://www.vse.uz/ – Энциклопедия поисковых систем

ОГЛАВЛЕНИЕ

Научное издание

Абдуллаева Озода Сафибуллаевна

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ТЕХНИЧЕСКИХ СИСТЕМАХ

Учебник

Редактор: Г.Зокирова.
Технический редактор: И.Юсупов.
Корректор: Д.Турабоева.

Подписано в печать: 18.02.2022 г.
Формат: 60x90 $^{1}/_{16}$. Усл. печ. л. 15,75.
Гарнитура Times New Roman. Бумага офисная.
Тираж 200 экз. Цена договроная.

Издательство «ARJUMAND MEDIA».
г. Наманган, улица Навоий, 36.
Изд. № AI – 007, 20.07.2018 г.

Отпечатана в типографии ООО "VODIY POLIGRAF".
Адрес: г. Наманган, 5 микрорайон, ул. Галаба, д. 19.