МИНИСТЕРСТВО СЕЛЬСКОГО ХОЗЯЙСТВА
И ПРОДОВОЛЬСТВИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ АГРАРНЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Н. Г. Серебрякова, О. Л. Сапун, Р. И. Фурунжиев

ОСНОВЫ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Рекомендовано
Учебно-методическим объединением
по аграрному техническому образованию
в качестве пособия для студентов
учреждений высшего образования
группы специальностей 74 80
Научная и педагогическая деятельность

Минск
БГАТУ
2015

УДК 004(075.8)
ББК 32.81я7
С32

Рецензенты:
заведующий кафедрой информационных технологий
Белорусской государственной сельскохозяйственной академии,
кандидат экономических наук, доцент
Н. К. Шуин;
заведующий кафедрой прикладной математики и информатики
Белорусского государственного педагогического университета имени
Максима Танка
С. И. Зенько

Серебрякова, Н. Г.

С32 Основы информационных технологий : пособие / Н. Г. Серебрякова, О. Л. Сапун, Р. И. Фурунжиев. – Минск : БГАТУ, 2015. – 400 с.
ISBN 978-985-519-700-4.

Содержит теоретический материал в рамках подготовки к сдаче кандидатского дифференцированного зачета. Дается обзор программных средств информационных технологий, рассматриваются основные понятия информационных технологий, сетевые технологии, системы управления базами данных, вопросы защиты информации, основы математического моделирования и численных методов, методы оптимизации.

Для магистрантов, аспирантов технических и экономических специальностей и соискателей ученой степени.

УДК 004(075.8)
ББК 32.81я7

ISBN 978-985-519-700-4

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ

Современный этап развития общества принято рассматривать в контексте широкой информатизации всех его сфер. Эффективное развитие сельского хозяйства невозможно без технического переоснащения производства, базирующегося на внедрении перспективных достижений науки и техники. Прогресс в развитии информационных технологий открывает дополнительные возможности в проведении научно-технических исследований в области агропромышленного производства.

В настоящее время информационные технологии – одна из самых динамично развивающихся областей. Совершенствуется элементная база и архитектура компьютеров, развиваются языки и технологии программирования, создаются новые пакеты прикладных программ на основе современных математических методов моделирования и оптимизации.

Одна из важнейших особенностей технического университета – фундаментальная подготовка инженера на основе расширенного цикла математических, естественнонаучных и общеинженерных дисциплин. Для этого необходимо современное учебно-методическое обеспечение, использующее передовые информационные технологии.

Целью дисциплины «Основы информационных технологий» является подготовка магистрантов и аспирантов к использованию современных информационных технологий как инструмента для решения на высоком уровне научных и практических задач в своей предметной области.

Задачами дисциплины являются:

– приобретение навыков использования технических устройств управления информацией и работы с компьютером;

– изучение современных средств телекоммуникаций;

– овладение современными технологиями информационного обеспечения научных исследований;

– обеспечение создания и ведения баз данных по технологической и эксплуатационной наследственности деталей машин и их соединений;

– овладение навыками работы с профессиональными базами данных;

– овладение информационными технологиями проектирования машин, сборочных единиц и технологических процессов.

Изучение дисциплины способствует формированию следующих компетенций:

1. Академических, включающих:

– способность самостоятельно приобретать новые знания и умения, в том числе в областях знаний, непосредственно не связанных со сферой деятельности;

– способность к самостоятельной научно-исследовательской деятельности, готовность генерировать и использовать новые идеи;

– методологические знания и исследовательские умения, обеспечивающие решение задач научно-исследовательской, научно-педагогической, управленческой и инновационной деятельности;

– способность использовать современные информационные технологии и базы данных;

– способность к самостоятельному обучению новым методам исследования, изменению научного профиля своей профессиональной деятельности;

2. Социально-личностных, включающих умения:

– совершенствовать и развивать свой интеллектуальный и общекультурный уровень;

– владеть навыками формирования и аргументации собственных суждений и профессиональной позиции;

– оказывать личным примером позитивное воздействие на окружающих и участников профессиональной деятельности с точки зрения соблюдения норм и правил здорового образа жизни, активной жизненной позиции;

– сотрудничать с коллективом, работать в команде, руководить и подчиняться;

– обобщать, анализировать, критически осмысливать, систематизировать, прогнозировать, определять цели и выбирать пути их достижения; владеть культурой мышления;

3. Профессиональных, включающих способность:

– подготавливать и проводить учебные занятия в учреждениях среднего специального и высшего образования;

– разрабатывать и использовать современное научно-методическое обеспечение;

– осуществлять мониторинг образовательного процесса, диагностику учебных и воспитательных результатов;

– руководить научно-исследовательской работой обучающихся;

– планировать и организовывать воспитательную работу с обучающимися;

– использовать современные средства и методы решения задач, связанных с реализацией образовательной и воспитательной деятельности;

– осваивать и внедрять в образовательный процесс инновационные образовательные технологии;

– анализировать работу основных аппаратных средств персонального компьютера, осуществлять поиск и изучение информации, применять персональный компьютер при решении прикладных задач;

– формулировать проблемы, решать задачи, разрабатывать планы и обеспечивать их выполнение в избранной сфере профессиональной деятельности.

В результате изучения дисциплины магистрант должен ***знать:***

– основные принципы создания, хранения, обработки и передачи информации;

– основы технологии обмена данными;

– методы и средства решения задач в своей предметной области на базе использования информационных технологий.

В результате изучения дисциплины магистрант должен ***уметь:***

– пользоваться основными программными продуктами информационных технологий: текстовыми, табличными и графическими процессорами, базами данных, средствами подготовки презентаций, сетевыми клиентскими программами, средствами поддержки математических вычислений;

– пользоваться глобальными информационными ресурсами;

– осуществлять поиск, систематизацию и анализ информации по перспективам развития отрасли, инновационным технологиям, проектам и решениям;

– проводить с использованием информационных технологий расчеты и оформление проектно-конструкторской документации;

– использовать современные достижения науки и передовые технологии в области систем технического обеспечения сельского хозяйства и автоматизации технологических процессов производства;

– выбирать методы расчета и анализа систем технического обеспечения сельского хозяйства, анализировать и представлять результаты научных исследований;

– намечать практические рекомендации по использованию научных исследований в области технического обеспечения сельского хозяйства;

– планировать и проводить аналитические, иммитационные и экспериментальные исследования, критически оценивать результаты и делать выводы.

– выбирать методы и проводить испытания для оценки физических, механических и эксплуатационных свойств средств механизации и систем технического обеспечения сельскохозяйственного производства;

– проводить патентный поиск и исследовать патентоспособность и показатели технического уровня разработок систем технического обеспечения сельского хозяйства;

– разрабатывать научно-техническую документацию, оформлять научно-технические отчеты, обзоры, публикации по результатам выполненных исследований;

– применять методы анализа вариантов, разработки и поиска компромиссных решений;

– использовать средства компьютерной техники при проектировании;

– оформлять проектную документацию;

– принимать оптимальные управленческие решения;

– осваивать и реализовывать управленческие инновации в профессиональной деятельности;

– разрабатывать планы и программы организации инновационной деятельности;

– применять инновационные методы решения профессиональных задач.

Освоив дисциплину студент должен ***иметь научное представление*:**

– о современных информационных технологиях в предметной области;

– о современных операционных системах и инструментальных пакетах программ;

– об основных программных продуктах информационных технологий: текстовых, графических, табличных процессорах, базах данных, средствах подготовки презентаций и средствах поддержки математических вычислений;

– сетевых технологиях и сервисах сети Интернет;

– проблемах защиты информации в компьютерах и компьютерных сетях.

Изучение дисциплины базируется на знаниях, полученных при изучении математики, информатики, информационных технологий, основ научных исследований и моделирования, оптимизации технологических процессов и принятия решений.

1. СОВРЕМЕННЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

1.1. История, современное состояние и перспективы развития вычислительной техники

Современные информационные технологии – это совокупность методов, производственных процессов и программно-технических средств, объединенных в технологическую цепочку, обеспечивающую сбор, регистрацию, обработку, накопление, хранение, отображение, поиск, анализ, защиту и распространение информации. В основе современных информационных технологий лежит вычислительная техника.

Вычислительная техника (ВТ) – совокупность технических и математических средств, используемых для механизации и автоматизации математических вычислений и обработки информации. В своем развитии вычислительная техника прошла за сравнительно короткий срок достаточно большой путь от замысла до воплощения в реальные машины. В развитии вычислительной техники принято выделять ряд этапов:

1. Ручной (до XVII в.);
2. Механический (с середины XVII в.);
3. Электромеханический (с 90-х гг. XIX в.);
4. Электронный (с 40-х гг. XX в.).

Впервые счетные устройства, называемые абак, появились, вероятно, в Древнем Вавилоне 3 тыс. лет до н. э. Первоначально это устройство представляло собой доску, разграфленную на полосы или со сделанными углублениями. Счетные метки (камешки, косточки) передвигались по линиям или углублениям. В V в. до н. э. в Египте вместо линий и углублений стали использовать палочки и проволоку с нанизанными камешками.

В 1623 г. Вильгельм Шиккард придумал «Считающие часы» – первый механический калькулятор, умевший выполнять четыре арифметических действия. Считающими часами устройство было названо потому, что как и в настоящих часах работа механизма была основана на использовании звездочек и шестеренок.

В 1642 г. Блезом Паскалем, французским ученым, в честь которого в наше время назван один из языков программирования, была

сконструирована счетная машина, которая могла выполнять операции сложения и вычитания. Она представляла собой механическую конструкцию с шестеренками и ручным приводом. Через тридцать лет, немецкий математик Готфрид Вильгельм Лейбниц построил другую механическую машину, которая помимо сложения и вычитания могла выполнять операции умножения и деления.

В 1822 г. Чарльз Бэббидж, профессор математики Кембриджского Университета, разработал и сконструировал аналитическую машину, которая, как и машина Паскаля, могла лишь складывать и вычитать. Поскольку аналитическая машина программировалась на элементарном ассемблере, ей было необходимо программное обеспечение. Созданием программного обеспечения занималась Ада Лавлейс. Таким образом, Ада Лавлейс стала первым в мире программистом. В ее честь назван современный язык программирования – Ada.

Работы по созданию отдельных элементов и узлов ЭВМ были начаты в 1937 г. в США Дж. Атанасовым. В 1942 г. им совместно с К. Берри была построена электронная машина АВС. Первая ЭВМ полностью на электронных лампах была названа ENIAC.

В 1944 г. немецкий инженер Конрад Цузе создал первую модель компьютера, которую сегодня многие считают первым реально действовавшим программируемым компьютером. В этом же году компьютер под названием «Mark I» разработал ученый из Гарварда – Говард Айкен. Его компьютер имел 72 слова по 23 десятичных разряда каждое и мог выполнить любую команду за 6 секунд. В устройствах ввода-вывода использовалась перфолента.

В 1947 г. под руководством С. А. Лебедева были начаты работы по созданию малой электронной счетной машины (МЭСМ). Эта ЭВМ была запущена в эксплуатацию в 1951 г. и стала первой ЭВМ в СССР и континентальной Европе.

Новые возможности по созданию вычислительной техники открылись с появлением электронных ламп и последующим бурным развитием электроники. Это новый период развития вычислительной техники делится на этапы, непосредственно связанные с уровнем развития элементной базы электронной техники, конструктивно-технологическим исполнением, логической организацией, математическим обеспечением, удобством общения человека машиной. Смена поколений электронно-вычислительных машин

(ЭВМ) происходила революционно, ей сопутствовало изменение технико-экономических показателей этих машин: быстродействие, надежность, потребляемая мощность, стоимость, габариты.

В настоящее время выделяют шесть этапов в развитии электронно-вычислительной техники, связанных с развитием элементной базы и промышленных технологий:

– ЭВМ на электронных лампах (1944–1956 гг.);

– ЭВМ на дискретных полупроводниковых и магнитных элементах (диоды, биполярные транзисторы, тороидальные ферритовые микротрансформаторы и ферромагнитные ячейки памяти) (1956–1964 гг.);

– ЭВМ на интегральных элементах малой плотности (1964–1971 гг.);

– ЭВМ на микропроцессорных элементах (1971–1990 гг.);

– ЭВМ на сверхбольших ИС и многопроцессорные системы (с 1990 по настоящее время);

– ЭВМ на новой элементной базе и новых принципах работы (настоящее и будущее).

Появлению первых ЭВМ предшествовали такие фундаментальные изобретения, как электронная лампа (1879 г.), триод (1913 г.). Триод, в отличие от двухэлектродной лампы, имеет еще один электрод – сетку. Благодаря наличию этого электрода появилась возможность управлять потоком электронов в лампе и создавать на их основе элементы памяти.

На этапах с первого по третий большой вклад в развитие вычислительной техники внесли советские ученые: С. А. Лебедев, И. С. Брук. Под их руководством были созданы ЭВМ первого поколения МЭСМ – 1951 г. (малая электронная счетная машина), БЭСМ – 1952–1953 гг. (большая электронная счетная машина). К машинам первого поколения можно отнести МЭСМ, БЭСМ, М-1, М-2, М-3, «Стрела», «Минск-1», «Урал-1», «Урал-2», «Урал-3», М-20, «Сетунь», БЭСМ-2, «Раздан».

В начале нового столетия наметился определенный сдвиг в развитии собственной элементной базы. В России в середине 2001 г. был введен в строй 768-процессорный суперкомпьютер МВС-1000М, обеспечивавший производительность в 1 Терафлоп. После этого Россия вышла на третье место в мире по мощности производимых суперкомпьютеров. В последующие годы совместными усилиями российских и белорусских

ученых создан ряд СуперЭВМ серии СКИФ, входящих в первую сотню мирового рейтинга компьютеров по производительности. Самый мощный в России, СНГ и Восточной Европе суперкомпьютер «СКИФ МГУ» занимает 22-е место в мировом рейтинге суперкомпьютеров TOP-500. Пиковая производительность суперкомпьютера «СКИФ МГУ» составляет 60 триллионов операций в секунду (60 Терафлоп).

ЭВМ пятого поколения не связаны с изменением элементной базы. В основу периодизации здесь для отличия их от ЭВМ четвертого поколения положены особенности архитектуры и организация вычислительного процесса. ЭВМ пятого поколения характеризуются наличием сверхсложных микропроцессоров с параллельно-векторной структурой, а также СуперЭВМ, содержащих в своей структуре сотни параллельно работающих процессоров, позволяющих строить системы обработки данных и знаний, эффективные сетевые компьютерные системы.

Современный этап развития ЭВМ можно охарактеризовать как этап развития машинного интеллекта. Вычислительные системы будущего будут ориентированы на обработку знаний и должны располагать развитыми возможностями логического вывода. Важнейшая черта их должна состоять в том, чтобы используемый интерфейс был непосредственно рассчитан на человека. Главными особенностями машин будущего будут речевой ввод-вывод информации и самообучаемость. Технический базис ее должна составить развивающаяся технология сверхбольших интегральных схем, создание памяти повышенного объема, возрастающие возможности высокоскоростных элементов.

Основу архитектуры должны составить системы с распределительными функциями, сетевая архитектура, машина базы данных, быстродействующая машина для численных расчетов, высокоуровневая система человеко-машинного общения. Основными системами программного обеспечения должны стать системы управления базами знаний, системы решения проблем и логического вывода, системы интеллектуального интерфейса.

Основными прикладными системами могут стать системы машинного перевода, вопросно-ответная система, прикладные системы понимания речи, изображений, рисунков, системы поддержки принятия решений.

1.2. Элементная база, архитектура, сетевая компоновка, производительность

Первый этап в развитии электронной вычислительной техники (ЭВМ) базировался на электронных лампах. Но электронные лампы обладали существенными недостатками: большие размеры, низкая надежность и др. Поэтому начала развиваться твердотельная электроника, а в качестве элементной базы стали применять диоды и транзисторы. Компьютеры, основанные на транзисторах, не устранили полностью эти недостатки. Для решения проблем начали применяться микросборки, а затем и микросхемы. Число элементов микросхем постепенно увеличивалось, стали появляться микропроцессоры. В настоящее время развитию электроники способствует появление сотовой связи, а также различных беспроводных устройств, навигаторов, коммуникаторов, планшетов и т. п.

Архитектура ЭВМ – концептуальная структура вычислительной машины, определяющая проведение обработки информации и включающая методы информации в данные и принципы взаимодействия технических средств и программмного обеспечения.

Архитектура ЭВМ определяется совокупностью общих принципов организации аппаратно-программных средств и их основных характеристик, обеспечивающих функциональные возможности вычислительной машины при решении соответствующих типов задач. Архитектура ЭВМ включает в себя как структуру, отражающую состав компьютера, так и программно-математическое обеспечение.

Структура ЭВМ – совокупность элементов и связей между ними. Основным принципом построения всех современных ЭВМ является программное управление.

Основы учения об архитектуре вычислительных машин были заложены Джон фон Нейманом. Совокупность этих принципов породила классическую архитектуру ЭВМ, представленную на рис. 1.1.

Сущность этих принципов заключалась в следующем:

– компьютер состоит из нескольких основных устройств (арифметико-логическое устройство, устройство управления, память, внешняя память, устройства ввода и вывода);

– арифметико-логическое устройство – выполняет арифметические и логические операции, необходимые для переработки информации, хранящейся в памяти;

– устройство управления – обеспечивает управление и кон-троль всех устройств компьютера (управляющие сигналы на рисунке указаны пунктирными стрелками);

– данные, которые хранятся в запоминающем устройстве, представляются в двоичной форме;

– программа, которая задает работу компьютера, и данные хранятся в одном и том же запоминающем устройстве;

– для ввода и вывода информации используются устройства ввода и вывода.

Рис. 1.1. Схема классической архитектуры ЭВМ

Принцип работы и структуру ЭВМ рассмотрим на примере персонального компьютера (ПК). На рис. 1.2 представлена схема структуры ПК. Основу ПК составляет системный блок, в котором размещены: микропроцессор (МП), блок оперативного запоминающего устройства (ОЗУ), постоянного запоминающего устройства (ПЗУ), долговременной памяти на жестком магнитном диске (винчестер), устройства для запуска компакт-дисков (CD) и дискет (НГМД). Там же находятся платы: сетевая, видеопамяти, обработки звука, модем (модулятор-демодулятор), интерфейсные

платы, обслуживающие устройства ввода-вывода – клавиатуры, дисплея, «мыши», принтеры и др.

Все функциональные узлы ПК связаны между собой через системную магистраль (шину), представляющую из себя более трех десятков упорядоченных микропроводников, сформированных на печатной плате.

Рис. 1.2. Схема структуры ПК

Микропроцессор служит для обработки информации: он выбирает команды из внутренней памяти (ОЗУ или ПЗУ), расшифровывает и затем исполняет их, производя арифметические и логические операции. Получает данные из устройства ввода и посылает результаты на устройства вывода. Он вырабатывает также сигналы управления и синхронизации для согласованной работы его внутренних узлов, контролирует работу системной магистрали и всех периферийных устройств.

С развитием вычислительной техники появились и приобрели широкое использование системы физического соединения двух или

более компьютеров – *компьютерные сети.* По территориально-организационным признакам (количеству машин и расстоянию между ними) компьютерные сети принято разделять на локальные и глобальные. Глобальные сети (например, Интернет) распространяют свое действие по всему миру и используют все каналы связи, включая спутниковые.

Архитектурный принцип построения большинства сетей называется «клиент-сервер». *Сервер* – компьютер сети, предоставляющий свои программные и аппаратные ресурсы пользователям сети для хранения данных, выполнения программ и других услуг. *Клиент* – компьютер сети, пользующийся услугами сервера; в его роли часто выступают программы, имеющие доступ к информационным ресурсам или устройствам сервера. Термины «клиент» и «сервер» используются для обозначения как программных, так и аппаратных средств.

Для передачи данных в сети используются специальные стандарты, обеспечивающие их совместимость – *сетевые протоколы.* Примером универсального протокола является семейство TCP/IP, широко применяющееся во всем мире для объединения компьютеров в сеть Интернет, которая состоит из множества сетей различной физической природы.

История его возникновения связана с задачей, поставленной после второй мировой войны правительством США. Требовалось создать единую сеть, которая могла бы своими средствами находить маршруты передачи данных, а также в случае повреждения некоторых каналов связи перенаправлять поток информации по другим каналам. При реализации этого проекта были созданы отдельные представители семейства протоколов TCP/IP.

Контрольные вопросы

1. Назовите основные этапы развития вычислительной техники.
2. Назовите ученых, которые внесли существенный вклад в развитие вычислительной техники.
3. Какие научные открытия предшествовали появлению первых вычислительных машин на электронных лампах?
4. Назовите особенности и направления развития вычислительной техники шестого поколения.

1.3. Понятие информации. Классификация и виды информационных технологий

Емкое понятие «информационные технологии» формируется двумя не менее емкими понятиями технология и информация. Термин *технология* (от греч. – «способ производства») – совокупность методов, процессов и материалов, используемыхз в какой-либо отрасли деятельности, а также описание типовых способов технического производства с применением современных методов и средств. Понятие *информация* (от лат. *informatio* – «разъяснение, осведомленность») – сведения о чем-либо, независимо от формы их представления.

В настоящее время не существует единого определения информации как научного термина. С точки зрения различных областей знания данное понятие описывается своим специфическим набором признаков. Понятие «информация» является базовым в курсе информатики, где невозможно дать его определение через другие, более «простые» понятия. В научном подходе выявлении наиболее характерных определений понятия информации привело к двум принципиальным изменениям в трактовке понятия информации.

Во-первых, оно было расширено и включило обмен сведениями не только между человеком и человеком, но и между человеком и автоматом, автоматом и автоматом; обмен сигналами в животном и растительном мире. Передачу признаков от клетки к клетке и от организма к организму тоже стали рассматривать как передачу информации.

Во-вторых, была предложена количественная мера информации в работах К. Шеннона и А. Н. Колмогорова, что привело к созданию теории информации.

Согласно концепции К. Шеннона, информация – это снятая неопределенность, т. е. сообщения, которые должны снять в той или иной степени существующую у потребителя до их получения неопределенность, расширить его понимание объекта полезными сведениями. В противном случае эти сообщения не представляют никакой ценности.

Американский инженер Р. Хартли в 1928 г. предложил формулу для определения количества информации I, содержащейся в сообщении, выбранном из множества N равновероятных сообщений:

$$I = \log_2 N.$$

Клод Шеннон развил идею Хартли для оценки количества информации при неодинаковой вероятности сообщений в наборе. Формула Шеннона имеет вид:

$$I = \sum_{i=1}^{n} \left(p_i \log_2 p_i \right),$$

где p_i – вероятность появления i-го события.

Понятие информации обязательно объединяет двух участников – ее источник и приемник. Для существования информации необходим также канал приема-передачи информации. При взаимодействии этих трех компонентов появляется и проявляется информация.

По способу передачи и приема различают информацию, передаваемую:

- изображениями – визуальную;
- звуками – аудиальную;
- ощущениями – тактильную;
- запахами и вкусами – органолептическую;
- информацию, передаваемую и воспринимаемую средствами вычислительной техники, – машинную.

Информацию, создаваемую и используемую человеком, по общественному назначению делят на три вида: личную, массовую, специальную.

В зависимости от области знаний различают научную, техническую производственную, правовую, патентную и иную информацию. Каждый вид информации имеет свои особые смысловые нагрузки и ценности, свои требования к ее точности и достоверности, преимущественные технологии обработки, формы представления и носители.

В роли источников и приемников информации могут выступать самые разнообразные объекты науки и техники, общества и природы. Разнообразие источников и потребителей информации привело к существованию различных форм ее представления.

Основные *формы представления информации*:

– символьная (основанная на использовании символов: букв, цифр, знаков);

– текстовая (использует тексты, т. е. символы, расположенные в определенном порядке);

– графическая (различные виды изображений);

– звуковая.

Символьная форма является наиболее простой, но практически она применяется только для передачи несложных сигналов о различных событиях. Примером может служить сигнал светофора, сообщающий о возможности начала или прекращения движения пешеходам или водителям автотранспорта.

Текстовая форма также использует различные символы: буквы, цифры, математические, но информация заложена не только в этих символах, но и в их сочетании, в порядке следования.

Графическая форма представления информации является наиболее сложной. Сюда относятся фотографии, схемы, рисунки, чертежи, имеющие большое значение в деятельности человека.

Звуковая – устная или в виде записи и передачи лексем языка аудиальным путем.

Качество информации – совокупность свойств, отражающих степень пригодности конкретной информации об объектах и их взаимосвязях для достижения ее пользователями целей деятельности.

Возможность и эффективность использования информации обусловливаются такими ее *потребительскими показателями качества* как репрезентативность, содержательность, достаточность, доступность, актуальность, своевременность, точность, достоверность, устойчивость.

Репрезентативность информации связана с правильностью ее отбора и формирования в целях адекватного отражения свойств объекта.

Содержательность информации отражает семантическую емкость, равную отношению количества семантической информации в сообщении к объему обрабатываемых данных.

Достаточность (полнота) информации означает, что информация содержит минимальный, но достаточный для принятия правильного решения объем знаний.

Доступность информации восприятию пользователя обеспечивается выполнением соответствующих процедур ее получения и преобразования.

Актуальность информации определяется степенью сохранения ее ценности для управления в момент использования и зависит

от динамики изменения ее характеристик и от интервала времени, прошедшего с момента возникновения данной информации.

Своевременность информации означает ее поступление не позже заранее назначенного момента времени, согласованного со временем решения поставленной задачи.

Точность информации определяется степенью ее близости к реальному состоянию объекта, процесса, явления.

Достоверность информации определяется ее свойством отражать реально существующие объекты с необходимой точностью. Измеряется достоверность информации доверительной вероятностью необходимой точности, т. е. вероятностью того, что отображаемое информацией значение параметра отличается от истинного значения этого параметра в пределах необходимой точности.

Устойчивость информации отражает ее способность реагировать на изменения исходных данных без нарушения необходимой точности.

Основные свойства информации рассматривают в трех аспектах:

– техническом – когда важны точность, надежность, скорость передачи сигналов;

– семантическом – на первом плане передача смысла текста с помощью кодов;

– прагматическом – оценивается эффективность влияния информации на поведение объекта.

В одном терминологическом ряду с понятием «информация» в информационных технологиях применяются понятия «данные» и «знания».

Данные с точки зрения информационных технологий это документированная информация, циркулирующая в процессе ее обработки на электронно-вычислительных машинах. Данные хранятся в базах данных.

База данных – совокупность структурированной и взаимосвязанной информации, организованной по определенным правилам на материальных носителях. Базы данных могут объединяться в банки данных.

Банк данных – организационно-техническая система, включающая одну или несколько баз данных и систему управления ими.

Знания – это информация, на основании которой путем логических рассуждений могут быть получены определенные выводы.

Для обобщения накопленных знаний и обеспечения возможности использования их в практической деятельности, они так же, как и данные, организуются в базы знаний.

База знаний – совокупность формализованных знаний об определенной предметной области, представленных в виде фактов и правил.

Информация передается от источника информации к приемнику информации в материально-энергетической форме (электрической, звуковой, световой и др.). В зависимости от физической природы, а также способа измерения и преобразования, информация может быть представлена в непрерывной – аналоговой форме или в виде дискретных сообщений. В настоящее время для управления технологическими процессами и техническими системами применяются преимущественно цифровые электронные вычислительные машины, воспринимающие и обрабатывающие информацию, представленную в форме дискретных сообщений.

Поэтому возникает объективная необходимость преобразования непрерывных параметров, характеризующих состояние моделируемого процесса (температуры, давления, влажности, процентного содержания отдельных компонентов, экономических показателей и т. д.), в дискретную форму. Устройства, осуществляющие такие преобразования, называются *аналого-цифровыми преобразователями* (АЦП). Цифровые ЭВМ выдают информацию также в дискретной форме, однако для управления некоторыми процессами необходимы непрерывные сигналы. Обратное преобразование сигнала из дискретной формы в цифровую форму осуществляется с помощью *цифро-аналоговых преобразователей* (ЦАП).

Для представления информации используется алфавитный способ, основой которого является использование фиксированного конечного набора символов любой природы, называемого алфавитом. Символы из набора алфавита называются буквами, а любая конечная последовательность букв этого алфавита – словом. Примеры слов: AAD, 10110110. Символы алфавита при вводе в ЭВМ должны быть преобразованы в код.

В цифровых ЭВМ преимущественное распространение получило двоичное кодирование, при котором символы вводимой в ЭВМ информации представляются средствами двоичного алфавита, состоящего из двух символов 0 и 1, которые моделируются двумя фиксированными состояниями среды: есть напряжение, нет напряжения; включено устройство или выключено и т. д. Каждой букве ставится в соответствие последовательность нескольких двоичных знаков или двоичное слово. Такие последовательности называются кодовыми комбинациями.

Процесс получения кодовых комбинаций для представления букв одного алфавита средствами другого алфавита называется *кодированием*. Процесс обратного преобразования информации относительно ранее выполненного кодирования называется *декодированием*. Например, представление буквы А русского алфавита с помощью двоичных символов 11100001 есть процесс кодирования, обратный процесс получения буквы А из двоичного кода 11100001 есть декодирование. Полный набор кодовых комбинаций, соответствующий представлению всех букв одного алфавита средствами другого алфавита, называется *кодом*. Число символов, составляющих кодовую комбинацию, называется длиной кода или разрядностью кода. Максимальное число кодовых комбинаций N при заданной разрядности n определяется выражением: $N = 2^n$.

В вычислительной технике обычно используется ASCII код (американский стандарт кодов для обмена информацией), позволяющий закодировать 256 символов. В настоящее время в приложениях операционной системы *Windows* применяется UNICOD, который позволяет закодировать 65 536 символов.

Для измерения объема передаваемой или хранимой информации используются следующие единицы измерения:

– *бит* – один двоичный символ. Он позволяет представить в двоичной форме одно из двух различимых состояний объекта – «0» или «1»;

– *байт* – восемь двоичных символов;

– *слово* – два байта для 16-разрядных ЭВМ или 4 байта для 32 разрядных ЭВМ, 8 байт для 64-разрядных ЭВМ;

– *килобайт* – 1024 бита (2^{10});

– *мегабайт* – 1 048 576 байт (2^{20});

– *гигабайт* – 2^{30} байт, *терабайт* – 2^{40} байт, *петабайт* – 2^{50} байт.

Функционирование информационных технологий связано с различными формами представления информации: числовой, буквенной, графической, звуковой. Для этого разработаны определенные

форматы. Форматы представления информации определяются разрядной сеткой ЭВМ. Под *разрядной сеткой* понимают совокупность двоичных разрядов, используемых для хранения и обработки машинных слов. В ЭВМ число этих разрядов фиксировано и кратно восьми: 8, 16, 32, 64, 128 и т. д.

Управление вычислительным процессом в ЭВМ осуществляется с помощью команд, хранящихся в памяти. Команда состоит из двух частей: кода операции (КОП) и адреса. Код операции определяет действие, которое должна выполнить ЭВМ, а адресная часть команды содержит адреса операндов (чисел, знаков, функций), участвующих в операции. Иначе говоря, адресная часть команды определяет, откуда необходимо выбрать информацию или куда следует поместить информацию. Различают одноадресные, двухадресные и трехадресные команды (табл. 1.1). Могут быть и безадресные команды. В этом случае команда содержит только код операции, а адреса заранее записываются в определенные регистры процессора.

Таблица 1.1

Представление команд ЭВМ

Код операции	Адресная часть команды		
КОП	A1		
КОП	A1	A2	
КОП	A1	A2	A3

Рассмотрим, как может быть записана операция выполнения сложения с помощью трехадресной команды. В этом случае код операции содержит указание процессору выполнить операцию сложения. Адрес первого слагаемого (операнда) содержится в ячейке с адресом A1, адрес второго слагаемого (операнда) хранится в ячейке с адресом A2, а в ячейке с адресом A3 хранится адрес, куда следует поместить результат вычисления. Для выполнения операции сложения с помощью одноадресной команды потребуется последовательность трех команд. То есть, чем больше адресов содержится в адресной части команды, тем быстрее будут выполняться вычислительные операции.

При обработке информации применяются различные системы счисления. Под *системой счисления* понимают совокупность приемов и правил для записи чисел цифровыми знаками. Различают позиционные и непозиционные системы счисления. Примером непозиционной

системы счисления является римская система, использующая набор символов I, V, X, C, L, D. В этой системе значение символов не меняется в зависимости от положения их в числе. Так, в числах VI и IV символ V имеет одно и то же значение – 5.

В позиционной системе счисления значение цифры определяется положением в числе: один и тот же знак принимает различные значения. Например, в вещественном числе 132,28 цифра 2 определяет различные значения в зависимости от расположения:

$$132{,}28 = 100 + 30 + 2 + 0{,}2 + 0{,}08 =$$
$$= 1 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0 + 2 \cdot 10^{-1} + 8 \cdot 10^{-2}.$$

Приведенный пример записи вещественного числа в десятичной системе счисления позволяет продемонстрировать общую форму записи чисел в позиционной системе счисления. Произвольное число A можно представить в виде полинома:

$$A(q) = a_{n-1}\, q^{n-1} + a_{n-2}\, q^{n-2} + \ldots +$$
$$+ a_i\, q^i + \ldots + a_1\, q^1 + a_0\, q^0 + a_{-1}\, q^{-1} + \ldots + a_{-m}\, q^{-m}, \qquad (1.3)$$

где a_i – символы алфавита системы счисления;

q – основание системы счисления;

n, m – число целых и дробных разрядов соответственно.

Пример 1.1. Определите значение числа, представленного двоичным кодом 101101.011_2.

Решение. Число содержит целую и дробную части $n = 6$, $m = 3$:

$$A(2) = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 +$$
$$+ 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1 \cdot 32 + 0 \cdot 16 + 1 \cdot 8 + 1 \cdot 4 + 0 \cdot 2 +$$
$$+ 1 \cdot 1 + 0 \cdot 1/2 + 1 \cdot 1/4 + 1 \cdot 1/8 = 32 + 8 + 4 + 1 + 1/4 + 1/8 =$$
$$= 45{,}375_{10}.$$

Любая позиционная система счисления характеризуется основанием (базисом). *Основание позиционной системы счисления* – это число знаков или символов для изображения цифр в данной системе. В настоящее время в вычислительной технике нашли наибольшее применение позиционные системы счисления, приведенные в табл. 1.2.

Таблица 1.2

Позиционные системы счисления

Наименование	Алфавит	Основание Системы счисления
Двоичная	0, 1	2
Восьмеричная	0, 1, 2, 3, 4, 5, 6, 7	8
Десятичная	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	10
Шестнадцатеричная	0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F	16

Если задано число $A_{\max}(q)$, то можно определить требуемое число разрядов для его представления – n:

$$n = \log_a(A_{\max}(q) + 1). \quad (1.3)$$

И наоборот, если известна длина разрядной сетки n, то можно определить максимальное число $A_{\max}(q)$, которое можно представить с использованием данной разрядной сетки:

$$A_{\max}(q) = q^n - 1. \quad (1.4)$$

Например, с помощью одного байта (восьми разрядов) можно представить число 255:

$$2^8 - 1 = 256 - 1 = 255,$$

то есть числа, которые можно представить с использованием данной разрядной сетки, заключены между некоторым минимальным значением и некоторым максимальным значением. Интервал числовой оси, заключенный между максимальным и минимальным числами, называют *динамическим диапазоном.*

Арифметические действия, выполняемые в двоичной системе, подчиняются тем же правилам, что и в десятичной. Перенос единиц в старший разряд возникает так же как в десятичной системе счисления (табл. 1.3).

Основные свойства информационных технологий

Информационная технология (ИТ) – комплекс взаимосвязанных научных, технологических, экономических, инженерных дисциплин, занимающихся изучением, созданием и применением методов,

средств, правил, используемых для получения новой информации (сведений, знаний), сбора, обработки, анализа, хранения, интерпре-

Таблица 1.3

Таблицы сложения и умножения

+	0	1
0	0	1
1	1	10

×	0	1
0	0	0
1	0	1

тации, выделения и применения данных и информации с целью удовлетворения информационных потребностей народного хозяйства и общества в требуемом объеме и заданного качества с применением современных программных и компьютерных средств. В настоящее время под информационными технологиями чаще всего подразумевают компьютерные информационные технологии, так как имеет место активное использование компьютерной техники и программного обеспечения для получения, хранения, обработки и передачи информации.

Согласно определению, принятому ЮНЕСКО, ИТ – это комплекс взаимосвязанных научных, технологических, инженерных дисциплин, изучающих методы эффективной организации труда людей, занятых обработкой и хранением информации; вычислительная техника и методы организации и взаимодействия с людьми и производственным оборудованием, их практические приложения, а также связанные со всем этим социальные, экономические и культурные проблемы.

Выделяют следующие основные характеристики информационной технологии:

– предметом процесса обработки являются данные;

– целью процесса является получение, хранение, обработка, передача информации;

– средствами осуществления процесса являются программные, аппаратные и программно-аппаратные вычислительные комплексы;

– процессы обработки данных разделяются на операции в соответствии с данной предметной областью;

– выбор управляющих воздействий на процессы осуществляется лицами, принимающими решения;

– критерием оптимизации процесса являются своевременность доставки информации пользователю, ее надежность, доступность, полнота.

Основные составляющие информационных технологий:

1. *Техническое обеспечение* – это аппаратные средства и средства коммуникации, обеспечивающие работу ИТ. Как правило, включают персональные компьютеры, периферийные устройства, линии связи, сетевое оборудование и др.

2. *Программное обеспечение* (ПО) непосредственно реализует функции получения, обработки, хранения, отображения, поиска и анализа данных, обеспечивает взаимодействие пользователя с ЭВМ посредством пользовательского интерфейса. ПО находится в прямой зависимости от технического обеспечения.

3. *Информационное обеспечение* представляет собой совокупность проектных решений по видам, объемам, способам размещения и формам организации информации.

4. *Методическое обеспечение* – это комплекс нормативно методических и инструктивных материалов по подготовке и оформлению документов по эксплуатации технических средств и компьютерной сети, организации работы специалистов – пользователей и технического персонала.

5. *Математическое обеспечение* – это совокупность алгоритмов, математических методов, моделей обработки информации, используемых при решении функциональных задач и в процессе автоматизации проектных работ. Математическое обеспечение включает средства моделирования процессов проектирования и управления, методы и средства решения типовых задач экономики и управления, методы оптимизации запасов материальных ресурсов и принятия оптимальных управленческих решений.

6. *Правовое обеспечение* – это совокупность правовых норм, регламентирующих правоотношения при создании, внедрении и использовании ИТ.

7. *Лингвистическое обеспечение* включает совокупность научно-технических терминов и других языковых средств, используемых в ИТ, а также правил формализации естественного языка, включающих методы сжатия и раскрытия текстовой информации с целью повышения эффективности автоматизированной обработки информации и облегчающих общение человека со средствами ИТ.

Информационные технологии характеризуются следующими *основными свойствами*:

1. *Целесообразность*. ИТ реализуется с целью повышения эффективности производства или иной деятельности путем автоматизации бизнес-процесс, в которых большую роль играет надежность, своевременность и полнота информации. Бизнес-процесс представляет собой систему последовательных, целенаправленных и регламентированных видов деятельности, достигающих значимых для организации результатов.

2. *Системная полнота* или *целостность* процесса. Процесс включает все элементы, обеспечивающие необходимую завершенность действий человека в достижении поставленной цели.

3. *Регулярность* процесса и *высокая степень расчлененности* его на однозначные этапы.

4. *Взаимодействие с внешней средой*.

5. *Реализация во времени*. Информационная технология должна соответствовать новым потребностям организации, реагировать на изменения бизнес-процессов и учитывать новые возможности технических средств и программного обеспечения. Поэтому после внедрения ИТ, как правило, происходит ее постоянное развитие: модификация, изменение структуры, включение новых компонентов.

Классификация информационных технологий зависит от критерия классификации. В качестве критерия может выступать показатель или совокупность признаков, влияющих на выбор той или иной информационной технологии.

По типу обрабатываемых данных ИТ разделяются на текстовые, табличные, графические, гипертекстовые. Информационные технологии, предназначенные для обработки одновременно нескольких видов информации (текстовой, графической, звуковой, видео и т.д.), называются *мультимедийными информационными технологиями*.

По степени распределенности выполняемых операций выделяют *локальные* и *сетевые* информационные технологии.

Наиболее наглядной классификацией информационных технологий (ИТ) является классификация *по степени автоматизации задач управления*. В соответствии с этим признаком выделяют пять основных видов ИТ:

1. Технологии обработки данных. Предназначены для решения структурированных задач, алгоритмы решения которых хо-

рошо известны и для решения которых имеются все необходимые входные данные.

2. Автоматизация функций управления. Цель такой ИТ удовлетворение информационных потребностей всех без исключюючения сотрудников организаций, имеющих дело с принятием решений, на основе различных видов отчетов.

3. Электронный офис. Обеспечивает организацию и поддержку коммуникационных процессов как внутри организации, так и с внешней средой на базе компьютерных сетей и других современных средств передачи и обработки информации. Технология автоматизации офиса строится на базе таких продуктов, как текстовые и табличные редакторы, электронная почта, электронный календарь, телеконференции, системы электронного документооборота и т. д.

4. Системы поддержки принятия решений. Обеспечивают переработку больших объемов информации и принятия решения. Особенность данной ИТ в том, что человек активно участвует в данном процессе на начальной и завершающей стадиях (вводит данные в компьютер и принимает окончательное решение на основе полученной информации), а компьютер под управлением человека создает новую информацию.

5. Экспертная поддержка. Дает возможность получать консультации экспертов по любым проблемам, о которых этими системами накоплены знания, и принимать обоснованные решения.

Контрольные вопросы

1. Опишите принцип работы системы передачи информации.
2. Что такое код? Какое количество кодовых комбинаций необходимо для кодирования всех символов русского алфавита, белорусского алфавита?
3. Что такое разрядная сетка ЭВМ? Чем она определяется?
4. Какие форматы используются для представления: а) целых; б) вещественных чисел?
5. Сколько кодовых комбинаций можно представить с помощью одного байта?
6. В чем состоит особенность представления графической информации?

7. Каким образом представляются команды в ЭВМ?
8. Что такое система счисления?
9. Что называется основанием (базисом) системы счисления?
10. Какая система счисления называется позиционной системой счисления?
11. Представьте число 132,35610 в виде полинома.
12. Какое максимальное число можно представить с помощью двух (трех, восьми, шестнадцати) разрядов?
13. Как перевести число из двоичной системы счисления в восьмеричную (шестнадцатеричную) систему счисления?

1.4. Операционные системы

Операционная система (ОС) – комплекс управляющих и обрабатывающих программ ЭВМ, которые, с одной стороны, выступают как интерфейс между устройствами вычислительной системы и прикладными программами, а с другой стороны – предназначены для управления устройствами, управления вычислительными процессами, эффективного распределения вычислительных ресурсов между вычислительными процессами и организации надежных вычислений.

В логической структуре типичной вычислительной системы операционная система занимает положение между устройствами с их микроархитектурой, машинным языком и, возможно, собственными (встроенными) микропрограммами – с одной стороны и прикладными программами – с другой. Разработчикам программного обеспечения операционная система позволяет абстрагироваться от деталей реализации и функционирования устройств, предоставляя необходимый набор функций. В большинстве вычислительных систем операционная система является основной, наиболее важной (а иногда и единственной) частью системного программного обеспечения.

С 1990-х годов наиболее распространенными операционными системами являются системы семейства *Windows* и системы класса *UNIX* (особенно Linux и Mac OS).

Операционная система выполняет следующие функции:

– управление работой каждого блока компьютера и их взаимодействием;

– управление выполнением программ;

– организацию хранения информации во внешней памяти;

– взаимодействие пользователя с компьютером, т.е. поддержку интерфейса пользователя.

Классификация операционных систем:

1. По особенностям алгоритмов управления ресурсами бывают:

– *локальные*, управляющие ресурсами отдельного компьютера;

– *сетевые*, участвующие в управлении ресурсами сети.

2. По количеству одновременно выполняемых задач бывают:

– *однозадачные*, выполняющие функцию предоставления пользователю виртуальной вычислительной машины, обеспечивая его простым и удобным интерфейсом взаимодействия с компьютером, средствами управления периферийными устройствами и файлами;

– *многозадачные*, кроме вышеперечисленных функций, управляют разделением совместно используемых ресурсов, таких как процессор, оперативная память, файлы и внешние устройства.

Многозадачные ОС в зависимости от способа управления распределением процессорного времени различаются двух видов: ОС с невытесняющей и вытесняющей многозадачностью. Невытесняющая многозадачность (Windows 3x, NetWare) характеризуется тем, что активный процесс выполняется до тех пор, пока он сам, по собственной инициативе, не передаст управление операционной системе. При вытесняющей многозадачности (Windows 9x, Windows NT, OS/2, UNIX) решение о переключении процессора с одного процесса на другой принимается операционной системой, а не самим процессом. Многозадачные ОС позволяют, например, редактировать текст и выводить на печать другую программу в фоновом режиме, включить для комфорта музыку и работать в графическом редакторе или в электронной таблице и т. п.

3. По количеству одновременно работающих пользователей ОС делятся на:

– однопользовательские;

– многопользовательские.

Основным отличием многопользовательских систем от однопользовательских является наличие средств защиты информации каждого пользователя от несанкционированного доступа других пользователей.

По способу распространения различают открытое и закрытое ПО.

Закрытое программное обеспечение – это такая модель программного обеспечения, при которой автор (или иной правообладатель) удерживает за собой определенные права (в частности, запрещение повторного распространения, изменения программ или очень жесткого их ограничения). Для большинства программ исходный код недоступен, что делает невозможной или по крайней мере нетривиальной задачу модификации программ.

Открытое программное обеспечение – это такая модель программного обеспечения, при которой дается гарантия свободно распространять копии программы вместе с исходным кодом, изменять программу или использовать ее части в новых открытых разработках.

Операционная система обеспечивают ввод и вывод информации на различные устройства. Такими устройствами являются, как известно, дисководы, монитор, клавиатура, порты параллельного и последовательного ввода/вывода информации. Операционная система обращается к этим устройствам по их логическим именам.

Дисководы как физические устройства имеют номера. Имя диска обозначается буквой латинского алфавита с двоеточием. Например: первый диск обозначается A:, второй – B:, жесткий диск – C:. Ввиду того, что емкость жестких дисков в настоящее время большая, их с помощью программных средств разбивают на логические диски. На жестком диске в этом случае могут быть расположены логические диски D:, E:, F:, G:, H: и т. д.

Для других устройств установлены, например, следующие логические имена:

PRN – печатающее устройство (принтер); CON – консоль. При вводе информации под консолью понимается клавиатура, при выводе информации – экран видеомонитора.

Имена устройств используются в командах ОС для обращения к ним. На современных компьютерах число разъемов значительно больше: имеются специальные разъемы для подключения звуковых колонок, устройств мультимедиа, игровых приставок, внешней сети, телефона, микрофона, наушников.

Файловая система – это часть операционной системы, которая обеспечивает пользователю удобный интерфейс при работе с даными, хранящимися на диске, совместное использование файлов несколькими пользователями и процессами. Основными задачами файловой

системы являются организация хранения, поиска, переименования, копирования, пересылки и удаления файлов и каталогов.

Информация на дисках или других машинных носителях, а также в памяти компьютера хранится в файлах. *Файл* – это организованный, поименованный набор данных на диске или другом носителе информации.

Файлы бывают разных типов: обычные файлы, специальные файлы, файлы-каталоги. В файлах могут храниться тексты программ, документы, данные, сведения о файлах и других каталогах. Обычные файлы в свою очередь подразделяются на текстовые файлы и двоичные файлы. Текстовые файлы состоят из строк символов, представленных в ASCII-коде. Это могут быть документы, исходные тексты программ, данные и т. п. Текстовые файлы можно прочитать на экране и распечатать на принтере. Двоичные файлы не используют ASCII-коды, они часто имеют сложную внутреннюю структуру, например, объектный код программы или архивный файл.

Специальные файлы – это файлы, предназначенные для управления устройствами ввода-вывода. Файлы-каталоги – хранят сведения о файлах и каталогах.

Каждый файл для возможности его идентификации на носителе информации имеет обозначение, состоящее из двух частей: имени и расширения. Имя файла в дисковой операционной системе может содержать от 1 до 8 символов. Имя должно начинаться с буквы и не должно содержать знаков пунктуации и пробелов, может содержать специальные символы: _ – $ # & @ ! % ( ) {} `~ ^. В операционной системе Windows для обозначения имен файлов можно использовать длинные имена до 255 символов. В имени файла допускается использование пробела и точки, при этом последняя точка считается началом расширения имени файла.

Расширение имени файла начинается с точки, за которой следует несколько символов. В дисковой операционной системе длина расширения имени файла фиксирована и не должна превышать трех символов. В операционной системе Windows расширение имени файла может содержать и большее число символов. Расширение имени файла не является обязательным, однако оно является очень полезным, так как характеризует вид хранимой в нем информации.

Примеры имен файлов: command.com, expert1.bas, autoexec.bat.

В имени фала autoexec.bat: autoexec – имя файла; .bat – расширение имени файла; autoexec.bat – полное имя файла.

Файлы, имеющие расширение *.exe, *.com или *.bat, считаются внешними командами ОС. При вызове внешней команды можно вводить только имя файла без расширения. Если используется несколько файлов, имеющих одинаковые имена, но разные расширения, то при вводе имени этой команды ОС выполнит только одну программу в соответствии с приоритетом: *.com, *.exe, *.bat. Файлы с указанными расширениями называются исполняемыми. Причем файлы типа *.com, *.exe хранятся в двоичных кодах, а файлы с расширением *.bat. и в символьном виде и содержат последовательность команд, которые должны выполняться так же, как и при вводе с клавиатуры.

Файл может иметь один из четырех атрибутов: R – только для чтения, A – не архивирован, S – системный, H – скрытый. Если файл имеет атрибут R, то в него нельзя внести изменения. Данный атрибут позволяет защитить файл от несанкционированного изменения пользователем. Атрибут «системный» предназначен для использования операционной системой. Такой файл имеет защиту от несанкционированного удаления. При попытке удаления системного файла операционная система выдает предупреждение о том, что это опасная операция, и требует подтверждения на удаление. Атрибут «скрытый» позволяет скрыть файл от посторонних глаз.

При выполнении некоторых операций с файлами: копировании, переименовании, удалении, поиске файлов – возникает необходимость выделить группу файлов, имеющих, например, одинаковые имена или расширения имен. В этом случае для выделения файлов применяются маски.

Маска – это символ, который заменяет все слово, его часть или один символ. В качестве маски используются символы * и ?. Символ * заменяет все имя файла, расширение имени файла или их часть, символ ? означает любой символ в месте расположения данного знака. Например:

. – все файлы на текущем диске;

*.com – все файлы с расширением *.com;

s*.sys – все файлы с расширением *.sys, имя которых начинается с символа «s»;

tabl?.txt – все файлы с именем tabl и расширением .txt, отличающиеся последним символом в имени файла (tabl1.txt, tabl2.txt и т. д.).

Имена файлов регистрируются на магнитных дисках в каталогах (или папках). В операционной системе Windows вместо понятия «каталог» введено понятие папка. В дальнейшем не будем делать различия между этими понятиями.

Каталог – это файл, в котором хранятся сведения о файлах и подчиненных каталогах. К сведениям, которые хранятся в каталогах, относятся: имя и расширение имени файла, размер файла в байтах, дата и время его создания или последней модификации, атрибуты файла, начальный адрес файла на диске. На каждом диске всегда имеется один каталог, который называется корневым и обозначается символом \ – обратный слэш. Все остальные каталоги размещаются в корневом каталоге. На диске может быть несколько каталогов. Их наличие позволяет сгруппировать файлы по назначению, теме или пользователю, что облегчает их поиск на диске. Структуру каталогов на диске принято называть деревом каталогов.

Имена каталогов (папок) образуются по тем же правилам, что и имена файлов. Каталог, с которым работает пользователь в настоящий момент, называется текущим. Если в команде указать имя файла, то он будет отыскиваться или создаваться в текущем каталоге. Путь (маршрут) – последовательность каталогов, разделенных символом \, ведущая к файлу.

Сведения о файлах и каталогах хранятся в специальной области диска. В операционной системе Windows 9x (общее обозначение семейства ОС Windows 95, Windows 98) применяются файловые системы FAT-16, FAT-32. В операционной системе Windows NT, Windows XP и старших версиях ОС Windows применяется файловая система NTFS. В ОС Windows 95 использовалась FAT-16, в Windows 98 – FAT-16 и FAT-32. В этих файловых системах сведения о каталогах хранятся в корневом каталоге, а сведения о файлах – в таблице размещения файлов – FAT. Для обеспечения надежности на диске создается копия таблицы размещения файлов.

После включения питания процессор под управлением базовой системы ввода-вывода (BIOS), хранящейся в ПЗУ, проверяет наличие установленных внешних устройств, памяти и проводит их инициализацию. Затем проверяются платы расширения, в том числе и видеоадаптера. После этого проверяется ОЗУ. При неисправности монитора выдается последовательность звуковых сигналов. Если проверка прошла успешно, то BIOS пытается загрузить ОС с первого диска.

Если на жестком диске не обнаружено блока начальной загрузки, то осуществляется попытка загрузить ОС с компакт-диска. Последовательность опроса дисков может быть изменена путем настройки BIOS. Если на первом диске обнаружена неисправность, то выдается сообщение об ошибке. При обнаружении в первом секторе нулевой дорожки блока начальной загрузки BIOS загружает его в оперативную память компьютера и передает ему управление. Блок начальной загрузки загружает остальную часть операционной системы в ОЗУ компьютера. После загрузки ОС Windows на экране появляется рабочий стол с элементами управления. В некоторых случаях, например, при «зависании» компьютера, требуется его перезагрузка. Для перезагрузки компьютера следует нажать клавишу RESET, установленную на лицевой панели системного блока.

После окончания работы необходимо закрыть все приложения. Выход из программы Windows следует осуществлять только через команду главного меню Завершение работы. В этом случае будет обеспечено сохранение всех настроек пользователя и сохранность данных. При выходе иным способом возможны нарушение целостности файловой системы и потеря данных.

После ввода команды Завершение работы на экране монитора появляется окно диалога, в котором необходимо отметить мышью флажок «Выключить компьютер» и щелкнуть по кнопке ДА (ОК).

Контрольные вопросы

1. Назовите классификационные признаки операционных систем.
2. Назовите наиболее известные операционные системы, применяемые на персональных компьютерах.
3. Что такое файловая система и для чего она предназначена?
4. Что такое файл?
5. Что такое имя, расширение и спецификация файла? Приведите примеры записи спецификации файла.
6. Назовите наиболее распространенные расширения имен файлов. Что они означают?
7. Поясните, что такое маска. Приведите примеры использования масок.
8. Что такое атрибут файла? Какие атрибуты имеет файл?
9. Что такое каталог? Какая информация в нем содержится?

1.5. Языки и технологии программирования

Классификация языков программирования

Язык программирования – искусственный формализованный язык, представляющий собой набор ключевых слов (словарь), и система правил (грамматических и синтаксических) для конструирования операторов, состоящих из групп или строк чисел, букв, знаков препинания и других символов, с помощью которых пользователи могут сообщать компьютеру набор команд.

Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы, и действия, которые выполнит компьютер под ее управлением. Со времени создания первых программируемых машин человечество придумало более двух с половиной тысяч языков программирования и каждый год их число увеличивается.

По сложности языки программирования разделяют на:

– языки низкого уровня (машинно-ориентированные);

– языки высокого уровня (машинно-независимые).

К машинно-ориентированным языкам относятся:

– *машинный язык* (язык машинных кодов) – совокупность команд, интерпретируемых и исполняемых компьютером; каждый оператор является машинной командой, а данные в ОЗУ размещены по абсолютным значениям адресов;

– *ассемблер* (макроассемблер) – язык символического кодирования, где операторами являются машинные команды, которым приписываются искусственные обозначения, а в качестве операндов используются символические имена адресов в ОЗУ.

Примеры команд ассемблера:

CLA – очистить один из регистров сумматора (аккумулятор);

ADD – сложение содержимого ячейки, номер которой написан после команды, с содержимым аккумулятора (результат остается в аккумуляторе);

MOV – перемещение содержимого аккумулятора в ячейку с указанным номером;

HLT – стоп.

Преобразование текста в последовательность машинных команд выполняет промежуточная программа – *компилятор*.

На этапе компиляции производится распределение данных в ОЗУ, при этом вместо имен переменных подставляются относительные адреса ячеек, в которых располагаются данные. Абсолютные данные присваивает операционная система при размещении программы в ОЗУ компьютера перед ее использованием.

По функциональному назначению языки программирования высокого уровня разделяют на:

1. Проблемно-ориентированные – предназначены для решения специфических задач из некоторой отрасли знаний:

– Fortran (formula translator) – язык решения сложных научных и инженерных задач, первый язык высокого уровня;

– COBOL (common business oriented language) – язык для решения экономических и коммерческих задач;

– Algol (algorithmic language) – языки решения научно-технических задач;

– LISP (list processing language) – язык для решения задач искусственного интеллекта;

2. Универсальные – позволяют решить любую задачу, хотя трудоемкость решения в разных языках будет отличаться:

– PASCAL (Philips automatic sequence calculator);

– BASIC (Beginner ALL-purpose symbolic instruction code);

– C/C++;

– Jawa;

– C#.

3. Современные среды визуального объектно-ориентированного программирования DELPHI, Visual Basic.

Языки программирования принято делить на пять поколений.

В первое поколение входят языки, созданные в начале 50-х годов, когда только появились первые компьютеры. Это был первый язык ассемблера, созданный по принципу «одна инструкция – одна строка».

Расцвет второго поколения языков программирования пришелся на конец 50-х – начало 60-х годов. Тогда был разработан символический ассемблер, в котором появилось понятие переменной. Он стал первым полноценным языком программирования.

Появление третьего поколения языков программирования принято относить к 60-м годам. Такие качества новых языков, как относительная простота, независимость от конкретного компьютера

и возможность использования мощных синтаксических конструкций, позволили резко повысить производительность труда программистов. Подавляющее большинство языков этого поколения успешно применяется и сегодня.

С начала 70-х гг. по настоящее время продолжается период языков четвертого поколения. Эти языки предназначены для реализации крупных проектов, повышения их надежности и скорости создания. Они ориентированы на специализированные области применения, где хороших результатов можно добиться, используя не универсальные, а проблемно-ориентированные языки, оперирующие конкретными понятиями узкой предметной области.

Рождение языков пятого поколения произошло в середине 90-х годов. К ним относятся также системы автоматического создания прикладных программ с помощью визуальных средств разработки, без знания программирования.

Сравнительные характеристики, назначение и возможности современных языков программирования

Fortran (Фортран). Это первый компилируемый язык, созданный в 50-е годы. В Фортране впервые был реализован ряд важнейших понятий программирования. Удобство создания программ было положено в основу возможностей языка. Фортран продолжает активно использоваться во многих организациях.

Cobol (Кобол). Это компилируемый язык для применения в экономической области и решения бизнес – задач, разработанный в начале 60-х годов. В Коболе были реализованы очень мощные средства работы с большими объемами данных, хранящимися на различных внешних носителях.

Algol (Алгол). Компилируемый язык, созданный в 1960 г. В 1968 г. была создана версия Алгол 68, по своим возможностям и сегодня опережающая многие языки программирования.

Pascal (Паскаль). Язык Паскаль, созданный в конце 70-х годов, во многом напоминает Алгол, но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов. Дальнейшим развитием этого языка явилась более эффективная версия – *Object Pascal,* который лег в основу современного объектно-ориентированного языка программирования Delphi.

Basic (Бейсик). Этот язык по популярности занимает первое место в мире. Для этого языка имеются и компиляторы, и интерпретаторы. Он создавался в 60-х годах в качестве учебного языка и очень прост в изучении. Дальнейшим развитием этого языка явился язык объектно-ориентированного программирования (ООП) VISUAL BASIC FOR APPLICATION.

C (Си). Данный язык был создан в лаборатории Bell (США). Он планировался для замены ассемблера, чтобы иметь возможность создавать столь же эффективные и компактные программы, и в то же время не зависеть от конкретного типа процессора. Язык С во многом похож на Паскаль и имеет дополнительные средства для прямой работы с памятью (указатели). На этом языке в 70-е годы написано множество прикладных и системных программ и ряд известных операционных систем (Unix).

C++ (Си++). Этот язык, являющийся объектно-ориентированным, – расширение языка С был разработан в 1980 г. В нем реализовано множество новых мощных возможностей, которые позволили резко повысить производительность труда программистов, однако создание сложных и надежных программ требует от разработчиков профессиональной подготовки высокого уровня.

Java (Ява). Этот язык был создан компанией Sun (США) в начале 90-х годов на основе C++. Он призван упростить разработку приложений на основе C++ путем исключения из него всех низкоуровневых возможностей. Главная особенность этого языка – компиляция не в машинный код, а в платформенно-независимый байт-код (каждая команда занимает один байт). Этот байт-код может выполняться с помощью интерпретатора – виртуальной Java-машины JVM (Java Virtual Machine), версии которой созданы сегодня для любых платформ. Особое внимание в развитии этого языка уделяется двум направлениям: поддержке всевозможных мобильных устройств и микрокомпьютеров, встраиваемых в бытовую технику; созданию платформенно – независимых программных модулей, способных работать на серверах в глобальных и локальных сетях с различными операционными системами.

C#. По технологическим показателям подобен языку Java и находится между компилируемыми и интерпретируемыми языками. Программа компилируется не в машинный язык, а независи-

мый от машины код низкого уровня – байт-код. Далее байт-код выполняется виртуальной машиной.

Области применения современных ЭВМ настолько обширны и разнообразны, что существует большое число специализированных языков в различных областях науки и техники. Например, язык программирования баз данных SQL, язык разметки гипертекста HTML, язык программирования задач компьютерного инженерного анализа APDL системы ANSYS и другие.

Языки программирования баз данных

Эта группа языков отличается от алгоритмических языков, прежде всего решаемыми задачами. *База данных* – это файл (или группа файлов), представляющий собой упорядоченный набор записей, имеющих единообразную структуру и организованных по единому шаблону (как правило, в табличном виде). База данных может состоять из нескольких таблиц. Удобно хранить в базах данных различные сведения из справочников, картотек, журналов бухгалтерского учета и т. д. Для этого был создан структурированный язык запросов SQL (Structured Query Language). Он основан на мощной математической теории и позволяет выполнять эффективную обработку баз данных, манипулируя не отдельными записями, а группами записей.

Для управления большими базами данных и их эффективной обработки разработаны СУБД (Системы Управления Базами Данных). Практически в каждой СУБД помимо поддержки языка SQL имеется свой уникальный язык, ориентированный на особенности этой СУБД и не переносимый на другие системы. Сегодня в мире насчитывается пять ведущих производителей СУБД: Microsoft (SQL Server), IBM (DB2), Oracle, Software AG (Adabas), Informix и Sybase. Их продукты нацелены на поддержку одновременной работы тысяч пользователей в сети, а базы данных могут храниться в распределенном виде на нескольких серверах.

С появлением персональных компьютеров были созданы так называемые настольные СУБД. Родоначальником современных языков программирования баз данных для ПК принято считать СУБД dBase II, язык которой был интерпретируемым. Затем для него были созданы компиляторы, появились СУБД FoxPro и Clipper, поддерживающие диалекты этого языка. Сегодня похожие, но несовместимые версии языков семейства dBase реали-

зованы в продуктах Visual FoxPro фирмы Microsoft и Visual dBase фирмы Inprise.

Языки программирования для Интернета

С активным развитием глобальной сети было создано немало популярных языков программирования, адаптированных специально для Интернета. Все они отличаются характерными особенностями: языки являются интерпретируемыми, интерпретаторы для них распространяются бесплатно, а сами программы – в исходных текстах. Такие языки называют *скрипт-языками*.

HTML. Общеизвестный язык для оформления документов. Он очень прост и содержит элементарные команды форматирования текста, добавления рисунков, задания шрифтов и цветов, организации ссылок и таблиц. Все Web-страницы написаны на языке HTML или используют его расширения.

Perl. Был разработан в 80-х годах Ларри Уоллом. По мощности Perl значительно превосходит языки типа Си. В него введено много часто используемых функций работы со строками, массивами, всевозможные средства преобразования данных, управления процессами, работы с системной информацией и др.

VRML. Был создан в1994 г. для организации виртуальных трехмерных интерфейсов в Интернете. Он позволяет описывать в текстовом виде различные трехмерные сцены, освещение и тени, текстуры (покрытия объектов), вращать в любых направлениях, масштабировать, регулировать освещенность и т. д.

Языки моделирования

При создании программ и формировании структур баз данных нередко применяются формальные способы их представления – формальные нотации, с помощью которых можно визуально представить таблицы баз данных, поля, объекты программы и взаимосвязи между ними в системе, имеющей специализированный редактор и генератор исходных текстов программ на основе созданной модели. Такие системы называются CASE-системами. В них активно применяются нотации IDEF, а в последнее время все большее распространение получает UML.

Языки визуального программирования интерфейса

Программирование вручную привычных пользователю окон, кнопок, меню, обработка событий мыши и клавиатуры, включение в программы изображений и звука требовало все больше и больше

времени программиста. Выход из этой ситуации обозначился благодаря двум подходам.

Первый – стандартизация многих функций интерфейса, благодаря чему появилась возможность использовать библиотеки, имеющиеся, например, в Windows. В итоге при смене стиля графического интерфейса приложения смогли автоматически приспосабливаться к новой системе без какого-либо перепрограммирования.

Вторым революционным шагом явилось появление визуального программирования, возникшего в Visual Basic и C++Builder фирмы Borland. Визуальное программирование позволило свести проектирование пользовательского интерфейса к простым и наглядным процедурам, которые дают возможность за минуты или часы сделать то, на что ранее уходили месяцы работы.

Из универсальных языков программирования сегодня наиболее популярны следующие: Бейсик (Basic), Паскаль (Pascal), Си++ (C++), Ява (Java), Дельфи (Delphi), C#. Для каждого из этих языков программирования сегодня имеется немало систем программирования, выпускаемых различными фирмами. Наиболее популярны следующие визуальные среды быстрого проектирования программ для Windows:

- Basic: Microsoft Visual Basic;
- Pascal: Borland Delphi;
- C++: Borland C++Bulider;
- Java: Symantec Cafe.

Для разработки серверных и распределенных приложений можно использовать систему программирования Microsoft Visual C++, продукты фирмы Borland, практически любые средства программирования на Java.

Основные этапы технологии программирования

Технологии программирования включают следующие основные этапы: постановка задачи, разработка математической модели, разработка алгоритма, программирование, отладка программы, передача программы в эксплуатацию и научно-техническое сопровождение (НТС) программы, завершение жизненного цикла.

Постановка задачи

На этом этапе определяются основные цели и функции, выполнение которых должна обеспечивать программа, исходные данные, требования к исходным данным, выходные данные. Математическая

постановка задачи сводится к точному описанию исходных данных, условий задачи и целей ее решения с использованием математических выражений в общем виде. При этом должен применяться системный подход, то есть предмет должен быть исследован всесторонне, учтены все внешние и внутренние связи и их влияние на конечные результаты. Задача представляется в виде «черного ящика», на вход которого поступают исходные данные, ограничения на входные параметры, требования к входным и выходным параметрам, а выходом являются значения результирующих параметров.

Разработка математической модели

На данном этапе производится декомпозиция задачи, формализация, разработка математической модели, выбор метода решения. Под декомпозицией понимается разделение задачи на простые блоки, каждый из которых может разрабатываться самостоятельно и связан с другими частями программы только входными и выходными данными. Для деления задачи на блоки чаще всего используется функциональный подход. Например, в каждой вычислительной задаче можно выделить такие блоки, как ввод данных, вычислительный блок, блоки сохранения результатов вычислений на дисках, анализа результатов вычислений, графического представления результатов вычислений, печати результатов.

Разработка алгоритма программы

На этом этапе разрабатывается алгоритм решения задачи. Разработка алгоритма предполагает определение состава функциональных модулей и формирование общей схемы алгоритма, разработку алгоритмов функциональных модулей. В зависимости от сложности задачи алгоритм представляют вначале в общем виде (укрупненном). Затем каждый из блоков алгоритма разбивается на более мелкие задачи таким образом, чтобы на конечном этапе получить базовые схемы алгоритмов. Такой метод проектирования называется нисходящей разработкой алгоритма (проектирования).

Основные подходы к разработке алгоритмов и программ: структурное проектирование; информационное моделирование предметной области и связанных с ней приложений; объектно-ориентированное проектирование.

В основе структурного проектирования лежит последовательная декомпозиция, целенаправленное структурирование на отдельные

составляющие. Типичными методами структурного проектирования являются: нисходящее проектирование, кодирование и тестирование программ; модульное программирование; структурное программирование.

Модульное программирование основано на понятии модуля. Модуль – логически взаимосвязанная совокупность функциональных элементов, оформленных в виде отдельных программных модулей, имеющих один вход и один выход. Структурное программирование основано на модульной структуре программного продукта и типовых управляющих структурах алгоритмов обработки данных различных программных модулей. Структурное программирование применяется в основном при программировании отдельных модулей и заключается в переводе алгоритма программы на алгоритмический язык с использованием определенных конструкций языка программирования.

Информационное моделирование предметной области и связанных с ней приложений предполагает определение состава и способа представления исходных данных и результатов вычислений.

Объектно-ориентированное проектирование основано на использовании при программировании объектов − функциональных программных модулей, которые на экране монитора представлены в виде элементов, например, кнопок, списков, переключателей и т. п., обладающих определенной совокупностью свойств, методов и событий.

Программирование

Программа – упорядоченная последовательность команд (предписаний) компьютера для решения задач. В общетеоретическом плане программирование – это теоретическая и практическая деятельность, связанная с созданием программы. В узком смысле под программированием понимается запись алгоритма с использованием команд и операторов одного из языков программирования – кодирование.

Отладка программы

Отладка программы заключается в проверке правильности функционирования алгоритма решения задачи с помощью контрольных примеров – тестов, результаты решения которых заранее известны; устранении обнаруженных синтаксических и логических ошибок.

Научно-техническое сопровождение

Научно-техническое сопровождение программы предусматривает контроль над работой программы и устранение ошибок, обнаруженных в процессе эксплуатации, доработку программы и ее совершенствование в соответствии с требованиями заказчика.

Свойства алгоритмов

Алгоритм это точно определенная (однозначная) последовательность простых (элементарных) действий, обеспечивающих решение любой задачи из некоторого класса. Алгоритмам характерны следующие общие свойства:

дискретность – алгоритм можно разделить на отдельные шаги (действия), выполнение каждого из которых возможно только после завершения всех операций на предыдущем шаге;

детерминированность – совокупность промежуточных величин на любом шаге однозначно определяется системой величин, имевшихся на предыдущем шаге;

элементарность шагов – закон получения последующей системы величин из предыдущей должен быть простым и локальным;

направленность – если способ получения последующих величин из каких-либо исходных не приводит к результату, то должно быть указано, что следует считать результатом алгоритма;

массовость – начальная система величин может выбираться из некоторого множества (т. е. один алгоритм может применяться для решения класса задач).

Формализация представления алгоритмов

Поскольку любой алгоритм является набором входных, промежуточных и выходных данных, то для его описания и системы правил преобразования служит определенный язык. Естественные языки являются изменчивыми, неоднозначными и избыточными и не подходят для записи алгоритмов, требующих однозначной определенности. Наиболее простой путь устранения этих недостатков – построение искусственных языков со строгим синтаксисом и полной смысловой определенностью. Такие языки получили название формальных.

В любом языке можно выделить две составляющие – синтаксис и семантику.Синтаксис (грамматика языка) – совокупность правил, согласно которым в данном языке строятся конструкции.

Семантика – смысловая сторона языка, соотносит единицы и конструкции языка с некоторым внешним миром, для описания которого язык используется.

Синтаксис формального языка задается некоторой системой правил, которая из небольшого набора исходных конструкций порождает все допустимые их комбинации, т.е. язык образуется как множество разрешенных правилами сочетаний исходных конструкций. Кроме того, синтаксис содержит формулировку условия, которое выполняется для законченных конструкций языка и не выполняется в противном случае. Наиболее наглядным способом описания формального языка является синтаксическая диаграмма.

Синтаксическая диаграмма – схема (графическое представление) описания какого-либо нетерминального символа языка-объекта. Схема всегда имеет один вход и один выход, а ее элементы соединяются между собой направленными линиями, указывающими порядок следования объектов в определенном нетерминальном символе.

В представлении алгоритмов можно выделить две основные формы: символьную (словесную) и графическую.

Строчная форма записи является основным способом представления алгоритмов последовательностью строк, каждая из которых содержит описание одного или нескольких элементарных действий. Логика алгоритма (порядок действий) задается в явном виде путем указания метки последующей строки (в виде порядковых чисел или букв), или в неявном – по умолчанию передается строке, следующей за выполненной. Данный способ позволяет записать алгоритмическую нотацию для любого исполнителя – как человека, так и технического устройства. Недостатком строчной формы является неудобство целостного восприятия его логической структуры.

Формами строчной записи алгоритмов являются:

– пошагово-словесная форма – пронумерованная последовательность строк, содержащих описания конкретных действий на естественном языке;

– формула – строчная запись действий, обеспечивающих обработку числовых, символьных или логических данных;

– псевдокод – частично формализованный язык, ориентированный на исполнителя «человек», позволяющий записывать алгоритмы в форме, близкой к англоподобным языкам программирования;

– язык программирования – искусственный формализованный язык, предназначенный для записи алгоритма для исполнителя «компьютер», метаязыком которого является естественный язык.

Графическая форма записи или *блок-схема* для представления отдельных блоков алгоритма использует набор геометрических фигур согласно требованиям ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения и правила выполнения. Достоинство данной формы записи заключается в наглядности: блок-схема позволяет охватить весь алгоритм сразу, отследить различные варианты его исполнения, позволяет сделать записи как на естественном, так и на формальном языках.

Процедурное, объектно-ориентированное и логическое программирование. Поскольку ассемблер – машинно-зависимый язык, то записанная на нем программа может выполняться только на той технике (тем типом процессора), ассемблер которого был использован. Этот недостаток отсутствует у языков высокого уровня, которые ориентированы не на систему команд той или иной машины, а на систему операторов, характерных для записи определенного класса алгоритмов (операторы присваивания, условные операторы, циклы, операторы ввода-вывода).

Таблица 1.4

Различия концепций программирования

Программирование	Представление программ и данных	Исполнение программы	Связь частей программы между собой
Процедурное	Программа и данные представляют собой не связанные друг с другом элементы	Последовательное выполнение операторов	Возможна только через совместно обрабатываемые данные
Объектно-ориентированное	Данные и методы их обработки инкапсулированы в рамках единого объекта	Последовательность событий и реакций объектов на эти события	Отдельные части программы могут наследовать методы и элементы данных друг у друга

Окончание табл. 1.4

Программирование	Представление программ и данных	Исполнение программы	Связь частей программы между собой
Логическое	Данные и правила их обработки объединены в рамках единого логического и структурного образования	Преобразование логического образования в соответствии с логическими правилами	Разбиение программы на отдельные независимые части затруднительно

Контрольные вопросы

1. Что включает в себя базовое программное обеспечение?
2. Какие программные средства относятся к прикладному программному обеспечению?
3. Назовите классификационные признаки операционных систем.
4. Назовите наиболее известные операционные системы, применяемые на персональных компьютерах.
5. Что такое файловая система? Для чего она предназначена?
6. Что такое файл?
7. Что такое имя, расширение и спецификация файла? Приведите примеры записи спецификации файла.
8. Назовите наиболее распространенные расширения имен файлов. Что они означают?
9. Поясните, что такое маска. Приведите примеры использования масок.
10. Что такое атрибут файла? Какие атрибуты имеет файл?
11. Что такое каталог? Какая информация в нем содержится?
12. Что такое спецификация файла? Приведите примеры.

2. ОСНОВНЫЕ ПРОГРАММНЫЕ СРЕДСТВА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

2.1. Программное обеспечение. Текстовые редакторы, их возможности и назначение

Программное обеспечение (ПО) – это совокупность компьютерных программ получения, поиска, передачи, хранения, обработки данных и необходимых для их эксплуатации документов. По назначению программное обеспечение разделяется на системное, прикладное, инструментальное обеспечение разработки программ.

Системное ПО представляет собой совокупность взаимосвязанных программ, которые обеспечивают функционирование средств вычислительной техники как таковых без выполнения операций по реализации функций офисных технологий. Системное ПО является продолжением аппаратного обеспечения компьютера. Оно подразделяется на *базовое* и *сервисное.*

Базовое ПО включает операционные системы, командно-файловые процессоры (операционные оболочки), системные утилиты.

Операционная система – совокупность программных средств, обеспечивающих управление аппаратной частью компьютера и прикладными программами, а также их взаимодействие между собой и пользователем.

Командно-файловые процессоры или операционные оболочки – это специальные программы, предназначенные для облегчения общения пользователя с командами операционной системы (например, Norton Commander, Total Commander и др.).

Системные утилиты (от латин. utilitas – «польза») – программы, служащие для вспомогательных операций обработки данных или обслуживания компьютера (диагностика, тестирование аппаратных или программных средств, оптимизация использования дискового пространства, восстановления разрушенной на магнитном диске информации и т.д.).

Сервисное ПО – программы и программные комплексы для выполнения вспомогательных сервисных услуг, например, антивирусные программы.

Прикладное ПО представляет собой совокупность программных комплексов, обеспечивающих решение конкретных задач при реализации тех или иных функций офисных технологий. Прикладное программное обеспечение работает только при наличии системного программного обеспечения. Прикладное ПО включает пакеты прикладных программ (ППП), которые называют также *приложениями*.

Пакеты прикладных программ (ППП) – это комплекс взаимосвязанных программ для решения задач определенного класса конкретной предметной области. ППП могут быть общего или специального назначения.

К пакетам ППП общего назначения относятся:

- текстовые процессоры;
- табличные процессоры;
- системы управления базами данных;
- графические редакторы;
- системы разработки презентаций;
- системы обработки финансово-экономической информации;
- системы управления проектами;
- экспертные системы и системы поддержки принятия решения;
- пакеты прикладных программ специального назначения.

Пакеты прикладных программ специального назначения ориентированы на решение задач в определенной предметной области. К ним относятся: пакеты компьютерной математики для научно-технических расчетов (например, *Matematica, Mathcad, Matlab*), пакеты моделирования и компьютерного инженерного анализа (например, *Ansys.Inc, Nastran* и др.); пакеты статистической обработки данных; системы поддержки принятия решений (*Assistant Choice, Multi e*xpert и др.), корпоративные информационные системы (ERWin, BPWin и др.), обучающие программы, бухгалтерские и экономические пакеты и др. Прикладные программы пользователей служат для решения практических задач в различных предметных областях. Они автоматизируют практически все виды человеческой деятельности.

Инструментальное обеспечение разработки программ выполняется комплексом программного обеспечения, с помощью которого могут разрабатываться и адаптироваться к конкретным условиям применения те или иные функциональные программы, например

для офисных технологий. Для разработки программ разработаны и применяются мощные *системы программирования* – это совокупность программ для разработки, отладки и внедрения новых программных продуктов. Системы программирования обычно содержат: трансляторы; среду разработки программ; библиотеки справочных программ; отладчики; редакторы связей и др.

Текстовые редакторы. В настоящее время широкое распространение получил текстовый редактор *Microsoft Word 2010*, представляющий собой интегрированную программную среду для создания и редактирования документов произвольной структуры. Он обеспечивает ввод, редактирование и форматирование текста, вставку диаграмм, таблиц и рисунков, обмен данными с другими приложениями Windows, работу с гипертекстовыми документами, просмотр web-страниц и размещение документов на web-страницах, подготовку писем и их рассылку по электронной почте. Текстовый процессор содержит большой набор шаблонов, облегчающих создание стандартных документов, позволяет создавать записи в блоге.

Аналогичная версия текстового процессора представлена в офисном пакете *Open Office* – одном из ведущих систем для обработки текстов, электронных таблиц, презентаций, графиков, баз данных и многого другого. Этот мультиязычный и мультиплатформенный офисный пакет с открытым исходным кодом доступен на многих языках и работает на всех персональных компьютерах. Совместим с основными офисными пакетами. Его можно загрузить и использовать совершенно свободно для любых целей бесплатно.

Для верстки и дизайна бизнес-публикаций часто применяется программа *PageMaker*. Тесная интеграция с программами Adobe Photoshop, Illustrator, Acrobat вплоть до уровня drag and drop и совместимость с текстовым редактором Word 2010 обеспечивает удовлетворение большинства требований по верстке и оформлению.

Запуск программы осуществляется командой Пуск – Программы Microsoft Office 2010 – Microsoft Office Word 2010 или щелчком мыши по пиктограмме ***W*** на рабочем столе или в строке состояния. После запуска программы на выполнение на экране отображается рабочее окно Microsoft Word 2010 (рис. 2.1.). Выход из программы осуществляется командой Файл – Выход главного меню или комбинацией клавиш *Alt + F5*.

В верхней части расположена строка заголовка – *1*, в которой выводится имя редактируемого документа и название программы. В левой части строки заголовка размещается кнопка системного меню *W*, а в правой – кнопки свертывания, развертывания/восстановления и закрытия окна – *6*. Ниже строки заголовка расположены меню Файл и Лента – *2*, панель быстрого доступа – *3*, рабочее окно и строка состояния *11*.

В рабочем окне расположены: окно документа – *5*; горизонтальная и вертикальная линейки прокрутки – *9*. Горизонтальная линейка прокрутки появляется в том случае, когда ширина окна программы становится уже окна документа. Выше вертикальной линейки прокрутки расположены кнопка управления делением окна документа разделительной линией на две части по горизонтали – *7* и кнопка управления линейками – *8*, позволяющая показать или убрать линейки, не обращаясь к командам ленты Вид. На вертикальной линейке прокрутки расположена кнопка *Выбор объекта перехода – 10*.

В окне документа расположен курсор ввода (точка вставки), который отмечает место вставки текста или других объектов в документ – *13*; горизонтальная и вертикальная линейки прокрутки – *4*, *12*. Над вертикальной линейкой прокрутки расположена кнопка маркера табуляторов – *14*.

Меню Файл. Кнопка меню *Файл* (кнопка Office) расположена в верхнем левом углу экрана. Это меню содержит команды работы с файлами: создания, открытия, закрытия, сохранения, печати и др. Для выхода из меню щелкните по ярлычку ленты *Главная* или ярлычку другой ленты.

Панель инструментов. В программах офисного пакета Microsoft Office 2010 самым главным нововведением последних лет было использование ленточного интерфейса, заменившего меню и панели инструментов предыдущих версий. Лента, с расположенными на ней инструментами, размещается вверху окна. Лента состоит из трех элементов: вкладок, групп и команд.

Вкладки. На вкладках собраны команды по их функциональному назначению. Доступ к вкладкам осуществляется с помощью ярлычков. Имеется возможность добавлять на ленту новые кнопки, а также добавлять новую вкладку, новую группу на существующую вкладку с любым набором команд.

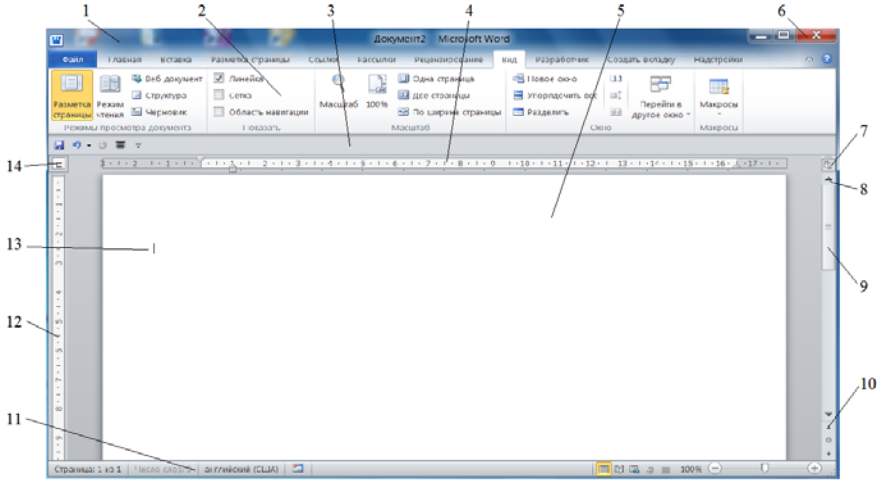


Рис. 2.1. Окно программы *MS Word 2010*

1 – строка заголовка; *2* – панель инструментов; *3* – панель быстрого доступа;
4 – горизонтальная линейка; *5* – окно документа;
6 – кнопки свертывания, развертывания и закрытия окна;
7 – кнопка деления окна документа; *8* – кнопка управления линейками;
9 – вертикальная линейка прокрутки; *10* – кнопка перехода к объекту;
11 – строка состояния; *12* – вертикальная линейка; *13* – курсор (точка вставки);
14 – установка табуляторов

Группы. Каждая вкладка содержит несколько групп команд, объединенных по функциональному назначению. Например, на вкладке *Главная* имеется пять групп: Буфер обмена, Шрифт, Абзац, Стили, Редактирование.

Команды – это кнопка, открывающая поле для ввода информации или меню. Все кнопки снабжены всплывающими подсказками.

Ленту можно изменять и дополнять. Для настройки ленты необходимо вызвать контекстное меню ленты щелчком правой кнопки мыши по свободному участку ленты и выбрать пункт меню *Настройка ленты* или выберите команду *Параметры – Настройка ленты* в меню *Файл*.

Панель быстрого доступа. Панель быстрого доступа позволяет разместить на ней те кнопки, которые должны быть всегда под рукой и которых нет на вкладке Главная, например, кнопки Сохранить,

Откат, Создать новый документ. Справа от установленных кнопок расположена кнопка раскрывающегося списка *Настройка панели быстрого доступа*. Эта кнопка открывает всплывающее меню, в котором содержится набор команд. Команду можно поместить на панель, щелкнув по ней мышкой. Для удаления команды с Панели быстрого доступа выделите команду в списке панели и щелкните по кнопке *Удалить*.

Строка состояния. Строка состояния расположена в нижней части окна программы. В левой части строки состояния выводится информация о редактируемом документе: номер текущей страницы, общее число страниц в документе, число слов в документе, используемый язык ввода. В правой части строки состояния расположены кнопки управления режимами просмотра документа, а также управление масштабом его представления. Аналогичные команды размещены в группах *Режимы просмотра документа* и *Масштаб* вкладки *Вид*.

Линейки прокрутки. Линейки прокрутки служат для просмотра документа. На вертикальной линейке прокрутки имеется несколько кнопок. Кнопки с одиночной стрелкой служат для перемещения текста на одну строку в соответствующем направлении, двойные стрелки перемещают текст на страницу. Быстрое перемещение по тексту удобно с помощью ползунка: зацепите ползунок мышью и перемещайте в требуемом направлении. При этом слева от ползунка появляется всплывающее окно сообщения, в котором отображаются номер страницы и наименование раздела. На вертикальной линейке прокрутки имеется кнопка – *Выбор объекта перехода*. При щелчке мышью по этой кнопке открывается меню, в котором можно выбрать объект для быстрого перехода: страница, разделы, примечания, сноски, концевые сноски, поля, таблицы, рисунки, заголовки, исправления, а также команды *Найти, Переход и Отмена.*

Создание и открытие документов

Начало работы с документом начинается с открытия меню *Фай*л. Создание нового документа можно выполнить так: ввести в окне программы Microsoft Word команду *Файл*, *Создать*. Открывается окно диалога, в котором следует выбрать *Новый документ* и щелкните по кнопке *Создать*. Все документы создаются на основе шаблонов. Документ, который создается командой *Новый*

документ, *Создать*, основан на шаблоне *Нормальный* по умолчанию. Microsoft Word 2010 предлагает пользователю большое число шаблонов разного назначения. Имеется возможность создать свой пользовательский шаблон, который будет сохранен в папке Мои шаблоны.

Сохранение документа

Рекомендуется регулярно сохранять работу на диске: командой *Файл*, *Сохранить* или *Файл*, *Сохранить как*. По умолчанию предлагается тип файла «Документ *Word*», расширение имени файла .docx. Можно установить режим автосохранения: выберите в меню *Файл* команду *Параметры, Сохранение* и установите флажок *Автосохранение каждые*….

Печать документов

Для вывода документа на печать необходимо воспользоваться командой *Печать* меню *Файл* или соответствующей кнопкой панели быстрого доступа.

Ввод и редактирование текста

Ввод текста

Текст документа вводится после точки ввода, которая отображается мигающей вертикальной чертой (*курсор*). При вводе длинного текста автоматически осуществляется переход на другую строку. При нажатии клавиши Enter курсор ввода переходит на другую строку в положение абзацного отступа. Для вставки пустой строки нажмите клавишу Enter. Чтобы разделить строку на две строки, установите курсор в точку раздела и нажмите клавишу Enter.

Выделение текста

Все операции в редакторе: копирование, перемещение, удаление – выполняются над выделенным текстом. Для выделения текста можно использовать несколько способов:

– отдельное слово выделяется двойным щелчком мыши;

– абзац выделяется тройным щелчком мыши;

– другой способ выделения абзаца – установите указатель мыши на поле выделения и протяните мышью по этому полю до последней строки выделяемого текста;

– для выделения произвольного фрагмента текста установите указатель мыши в начало выделяемого текста и протяните мышью до конца выделяемого текста.

Перемещение или копирование текста

Для копирования текста можно пользоваться командами *Копирование* и *Вставка* группы *Буфер обмена* вкладки Главная или использовать прием перетаскивания выделенного текста мышью при нажатой клавише Ctrl. Эти операции удобно выполнять также с помощью контекстного меню.

Перемещение отличается от копирования тем, что исходный текст удаляется. В этом случае для помещения текста (объекта) в буфер целесообразно использовать команду *Вырезать* группы *Буфер обмена*. При перемещении объекта мышью клавиша Ctrl не нажимается. Команда *Вставить* группы *Буфер обмена* имеет подменю, позволяющее выбрать режим вставки объекта. Полезно также запомнить комбинации клавиш: [Ctrl+C] – копировать выделенный текст в буфер обмена и [Ctrl + V] – вставить текст из буфера обмена.

Поля

Документ, подготовленный пользователем, содержит ряд элементов: поля, заголовки, форматированный текст, абзацы, колонтитулы, таблицы, рисунки, графики, формулы и т. п. Поля ограничивают текст документа. Левое поле используется обычно для подшивки и принимается равным 20–25 мм, правое поле – не менее 10 мм. Верхнее и нижнее поля используются для размещения колонтитулов. Размеры полей можно настраивать. Настройка параметров страницы осуществляется с помощью команд группы *Параметры страницы* на вкладке *Разметка страницы* или линейками.

Для удобства размещения таблиц, рисунков и других объектов, вставляемых в документ, полезно установить на страницы границы текста. Граница текста отображается пунктирной линией только в электронном документе. Для ее установки введите команду *Файл, Параметры, Дополнительно* и в группе *Показывать содержимое документа установите* флажок *Показывать границы текста*.

Заголовки

Заголовки оформляются с помощью стилей заголовков из списка Стили, расположенного в группе *Стили* на вкладке *Главная* ленты. Здесь же имеются два списка для изменения стилей документа. Выделенные стили можно настраивать с помощью контекстного меню. Стили, используемые в заголовках, не должны

использоваться в тексте документа для выделения других фрагментов текста. При оформлении заголовков с помощью стилей открывается возможность автоматизировать процедуру составления оглавления документа.

Оформление шрифтов

Чтобы сделать текст более читабельным, удобным для чтения и восприятия пользователем, применяют различное оформление шрифта. К параметрам шрифта, подлежащим настройке, относятся: тип шрифта, размер, начертание (полужирный, курсив, подчеркнутый), цвет, межсимвольный интервал и ряд дополнительных эффектов. Настройку параметров шрифтов осуществляют с помощью команд группы *Шрифт* вкладки *Главная* ленты или окна диалога *Шрифт*, которое вызывается кнопкой, расположенной в правом нижнем углу группы *Шрифт*. Окно диалога имеет две вкладки: *Шрифт* и *Дополнительно*. На вкладке *Шрифт* настраиваются практически все параметры шрифтов. Для печатного текста наиболее подходящими являются шрифты Times New Roman – шрифт с засечками и Arial – прямоугольный шрифт.

Оформление абзацев

Абзац – это фрагмент текста, который содержит, как правило, законченную мысль. Для текстового процессора абзац – это текст, заключенный между двумя нажатиями клавиши Enter. При нажатии клавиши Enter автоматически вставляется символ пробела. Точка вставки (курсор) переводится на новую строку в положение отступа первой строки. Параметры абзацев настраиваются с помощью команд группы *Абзац* вкладки *Главная* ленты или окна диалога *Абзац*, вызываемого кнопкой, расположенной в правом нижнем углу группы *Абзац*. Выравнивание может принимать четыре значения: по левому краю, по центру, по правому краю и по ширине. Отступы слева и справа определяют расстояние текста от границы текста, но не от края страницы. Первая строка может иметь три значения: без выступа, выступ вправо – красная строка или выступ влево – висячая строка. Часть параметров абзацев можно регулировать с помощью горизонтальной линейки. На ней имеется три маркера: два в левой части и один в правой части линейки. Эти маркеры позволяют устанавливать отступы и выступ первой строки.

Табуляторы

Выше вертикальной линейки прокрутки находится кнопка для установки табуляторов. Табуляторы позволяют управлять размещением текста. По умолчанию текст выравнивается по левому краю, этому состоянию соответствует маркер табуляции └, маркер ┴ означает выравнивание по центру, а маркер ┘ – выравнивание по правому краю, четвертый тип маркера – выравнивание по десятичному разделителю и пятый тип маркера – с чертой |. Маркер применяется к выделенному абзацу или абзацам. Установить позиции табуляции можно также с помощью команды *Табуляция* окна диалога *Абзац*.

Границы и заливка

Границы и заливка применяются для выделения текста, оформления страниц и абзацев. Для этой цели можно воспользоваться кнопкой *Границы* в группе *Абзац* вкладки *Главная* ленты или командой *Границы страниц* в группе *Фон страницы* вкладки *Разметка страницы* ленты. Выделите текст, который требуется заключить в рамку, и введите команду *Границы страниц* из группы *Фон страницы* – открывается окно диалога *Границы и заливка*. Окно диалога имеет три вкладки. Вкладка *Граница* позволяет установить границы на выделенный фрагмент текста, а вкладка *Страница* – установить границы страницы для всего документа или раздела, первой страницы или всех страниц, кроме первой.

Списки

Списки могут быть нумерованные, маркированные и многоуровневые (иерархические). Для оформления списков необходимо воспользоваться кнопками в группе *Абзац* вкладки *Главная*. Набранный текст следует предварительно выделить, затем выбрать в раскрывающемся списке вида списка нужный образец маркера (номера) и щелкнуть по нему мышью. При необходимости можно изменить значок или стиль оформления номера.

Группа Редактирование

В группе *Редактирование* вкладки *Главная* расположены кнопки Найти, Заменить и Выделить для выполнения редактирования. Первые две кнопки предназначены для поиска и замены текста в документе. А кнопка Выделить содержит

подменю: *Выделить все, Выбор объектов, Выделить текст, имеющий такой же формат, Область выделения.*

Оформление документа

Вставка объектов групп. Колонтитулы, Текст и Символы вкладки Вставка

Документ – материальный носитель с зафиксированной на нем в любой форме информацией должен быть определенным образом оформлен. В него могут вставляться различные объекты: номера страниц, сноски, рисунки, формулы, таблицы, диаграммы, оглавление и т. п. Рассмотрим некоторые из возможностей текстового процессора по оформлению документов. Вставка различных объектов в документ осуществляется с помощью вкладок ленты *Вставка, Ссылки, Рецензирование*.

Колонтитулы

Колонтитул – это короткий текст, расположенный в верхнем или нижнем поле страницы. Это может быть краткое сообщение, например, фамилия автора документа, наименование раздела, номер страницы, дата и время разработки документа, рисунки и т. п. По умолчанию все колонтитулы одинаковые, однако, можно изменить эту настройку. В группе *Текст* вкладки *Вставка* собран ряд команд для вставки текста и объектов: Буквица, Надпись, Строка подписи, Экспресс-блоки, WordArt, а также команды вставки даты и времени и объектов.

Вставка номера страниц

Вставка номеров страниц осуществляется командой *Номер страницы* вкладки *Вставка*.

Надпись

Данный объект позволяет вставлять предварительно отформатированный текст. Его вставка осуществляется командой *Надпись* вкладки *Вставка*. Это текстовый документ, заключенный в рамку, к которому можно применять все элементы форматирования текста, а также вставлять другие объекты: рисунки, картинки, формулы и т. п.

Буквица

Позволяет создать большую заглавную букву в начале документа. Для создания буквицы выделите заглавную букву в первом слове предложения и введите команду *Вставка, Текст, Буквица*. В окне диалога выберите способ размещения буквицы: в тексте или на поле.

Дата и время

Дата и время вставляются командой *Дата* и *Время* из группы *Текст*. После ввода команды открывается одноименное окно диалога, которое позволяет вставлять в документ поле даты и времени.

WordArt

Данная команда позволяет вставлять броские, красочные заголовки, выбираемые из коллекции шрифтов. При выборе команды на экране появляется шаблон, в который необходимо ввести свой текст. К данному тексту можно добавлять некоторые элементы форматирования, например, курсив.

Команда Объект

Данная команда позволяет вставлять текст из файлов, а также загружать в документ объекты, подготовленные другими приложениями: Word, Excel, Power Point, MathCad, редактор формул Microsoft Equation 3.0 и др.

Вставка спецсимволов

Вставка символов, отсутствующих на клавиатуре, осуществляется с помощью команды *Символы* вкладки *Вставка*.

Вставка формул

Для записи в текст документа формул применяется специальная программа *Microsoft Equation 3.0*, которая вызывается командой *Объект*, *Вставка объекта* из группы *Текст* вкладки *Вставка*. При загрузке программы на экране появляется панель инструментов, которая содержит набор элементов для вставки в формулы. При выборе любого значка открывается всплывающая панель с набором значков, соответствующих выбранной теме. Каждый значок имеет знакоместа, куда вписываются символы. Выбор знакоместа осуществляется мышью.

Вставка рисунков

Вставка рисунков осуществляется с помощью команд из группы *Иллюстрации* вкладки *Вставка*. Технология вставки рисунков следующая: щелкните мышкой по значку *Картинка* – откроется окно диалога Картинка. Откройте список *Искать объекты* и установите флажки у той группы объектов, которые хотите найти, щелкните по кнопке *Начать*. Открывается библиотека рисунков. Выберите нужный рисунок и щелкните по нему мышкой – рисунок вставляется в документ в точке вставки. Команда *Рисунок* позволяет вставить рисунок из файла.

Сетка

Для удобства рисования можно установить на лист сетку командой *Сетка* из группы *Показать* вкладки *Вид*. Границы сетки действуют как магнит, притягивая рисуемые объекты к узлам сетки.

Вставка объектов вкладки Ссылки

Вкладка *Ссылки* содержит шесть групп команд: *Оглавление*, *Сноски*, *Ссылки и списки литературы*, *Названия, Предметный указатель, Таблица ссылок*. Указанные группы команд позволяют подготовить к изданию брошюру, дипломный проект, диссертацию или учебное пособие в соответствии с требованиями, предъявляемыми к печатной продукции.

Оглавление

Оглавление – это список заголовков разделов, подразделов с указанием номеров страниц. В редакторе Word вставка оглавления предельно упрощена. Для вставки оглавления выполните следующее:

– напишите заголовки и подзаголовки к тексту документа и оформите их с помощью стилей заголовков группы *Стили* вкладки *Главная* ленты. Заголовок первого уровня должен быть самым крупным. Заголовки следующих уровней имеют меньшую высоту шрифта. Редактор позволяет создавать до 9 уровней заголовков;

– укажите место вставки списка;

– введите команду *Оглавление* на вкладке ***Ссылки***. Откроется список стандартных вариантов оформления оглавления. Если эти варианты не устраивают, то щелкните по кнопке *Оглавление* в конце списка. В этом случае открывается окно диалога для выбора другого формата и числа уровней оглавления.

Сноски

Сноска – это краткий комментарий к тексту, пояснение какого-либо термина, описание событий, связанных с какой-нибудь датой, и тому подобное. Различают обычную и концевую сноски. Обычная сноска помещается в конце текущей страницы, концевая сноска – в конце документа. Для вставки сноски установите курсор после текста, к которому будет относиться сноска, введите команду *Вставить сноску* из группы *Сноска*.

Ссылки и списки литературы

Ссылки облегчают создание списка литературных источников, использованных при разработке документа. В группе *Ссылки*

и списки литературы размещены кнопки *Управление источниками*, *Стиль*, *Список литературы* и *Вставить ссылку*. Список *Стиль* позволяет выбрать стиль оформления списка литературы. Список *Вставить ссылку* добавляет в текст документа ссылку на источник фрагмента текста, вставленного в документ, в соответствии с выбранным стилем. Ссылка выбирается из списка.

Названия

Команда *Вставить название* из группы *Названия* позволяет автоматизировать нумерацию рисунков, таблиц, формул и облегчает составление списка иллюстраций документа. При вводе команды открывается окно диалога *Название*. После номера рисунка следует ввести название этого рисунка. Номера рисунков программа формирует автоматически.

Перекрестная ссылка

Для создания перекрестной ссылки выполните следующее: в месте перекрестной ссылки напишите нужный текст; выделите слово *Сноски* (на этом месте может быть любой текст, так как он будет заменен на выбранный) и введите команду Вставка, Перекрестная ссылка – открывается окно диалога Перекрестная ссылка; выберите в списке «Тип ссылки» требуемый тип, например, Заголовок; в списке «Вставить ссылку на:» выберите объект, на который будете ссылаться (текст заголовка, номер страницы и др.); в окне «Для какого заголовка:» выделите заголовок Сноски; нажмите кнопку *Вставить*. Если предполагается создавать гиперссылку, то установите флажок «Вставить как гиперссылку» в окне диалога. Для завершения работы по созданию перекрестной ссылки щелкните по кнопке *Вставить*.

Закладка

Закладка предназначена для присвоения имени определенной позиции в документе. На закладки можно ссылаться при создании гиперссылок и ряда других полей Word. Для вставки закладки выделите объект (текст, формулу, заголовок, таблицу и др.) и введите команду *Вставка, Ссылки, Закладка*. Укажите имя закладки. Чтобы просмотреть закладки, введите команду *Файл, Параметры, Дополнительно* и в группе *Показывать содержимое документа установить/снять* флажок *Показывать закладки*. Закладки выделяются квадратными скобками. Для удаления закладки введите команду *Вставка*, *Ссылки, Закладка*, выделите нужное имя закладки и щелкните по кнопке Удалить в окне диалога.

Создание таблиц

Команды для работы с таблицами в текстовом процессоре Word 2010 размещаются в группе команд *Таблица* вкладки *Вставка* и на вкладке *Работа с таблицами* ленты. Вкладка Работа с таблицами открывается при создании или выделении таблицы. Она имеет две вкладки: *Конструктор* и *Макет*.

Создание таблицы осуществляется командой *Таблица* на вкладке *Вставка*. После ввода команды открывается меню команды *Таблица*. Оно позволяет создать новую таблицу несколькими способами:

1. Выделить мышью нужное число строк и столбцов в макете таблицы.
2. Вставить таблицу командой ***Вставить таблицу***. После ввода команды откроется окно диалога, в котором необходимо указать требуемое число строк и столбцов.
3. Нарисовать таблицу.
4. Вставить таблицу *Excel*.
5. Вставить таблицу из набора шаблонов – Экспресс-таблицу.

Таблица состоит из строк и столбцов. По умолчанию все столбцы нумеруются латинскими буквами от A до Z, а строки цифрами. Поэтому при написании формул к ячейке следует обращаться по ее адресу, например, S1. Каждая ячейка таблицы представляет собой самостоятельный документ, в который можно помещать числа, текст, рисунки, даты, таблицы. Редактируется текст в ячейке по тем же правилам, что и в основном документе.

Шаблоны

Шаблон – это набор параметров форматирования текста, абзацев, списков, элементов автотекста, макросов. Редактор Word имеет стандартный шаблон Normal, который загружается при открытии нового документа. Кроме того, он имеет большое количество других шаблонов для создания записок, факсов, писем, публикаций и т. п. Чтобы найти требуемый шаблон, введите команду ***Файл***, ***Создать***, затем выберите подходящий по содержанию шаблон и щелкните по кнопке ***Создать***. Шаблон может содержать поля ввода с текстом, который нужно заменить. В документ можно вставлять поля различного вида с помощью команды ***Вставка, Экспресс-блоки, Поля***.

Word 2010 позволяет создавать и собственные шаблоны – пользовательские. Для создания пользовательского шаблона введите команду ***Файл, Создать*** и выберите в окне диалога папку *Мои шаблоны*:

откроется окно диалога Создать. Активизируйте значок *Новый документ*, установите в группе ***Создать*** переключатель ***Шаблон*** и щелкните по кнопке ***ОК***. При сохранении документа шаблон сохраняется с расширением *.dotx (.dot) в папке Мои шаблоны.

В шаблон документа введите постоянный текст и вставьте необходимые поля с помощью команды ***Вставка, Экспресс-блоки*** (*Автотекст*, *Поле* и *Организатор стандартных блоков*). В шаблон можно вставлять такие объекты, как текст (A), поле ввода (ab|), флажки, списки. Элементы управления для вставки в шаблоны размещены в группе ***Элементы управления*** вкладки ***Разработчик*** ленты. В режиме ограниченной функциональности эти кнопки недоступны.

Создание шаблона пользователя

В группе *Элементы управления* вкладки *Разработчик* имеются следующие элементы для создания полей:

– *форматированный текст* (Aa) – ввод форматированного текста;
– *обычный текст* (Aa) – ввод обычного текста;
– *рисунок* – вставка элемента управления содержимым «рисунок»;
– *коллекция стандартных блоков* – служит для вставки стандартных блоков;
– *поле со списком* – содержит текстовое поле с раскрывающимся списком, позволяет вносить данные во время работы;
– *раскрывающийся список* – содержит список данных;
– *выбор даты* – служит для вставки даты;
– *флажок* – может иметь два состояния: установлен или снят.

Установка элементов управления осуществляется в режиме конструктора (режим установлен по умолчанию). Настройка параметров элементов управления также осуществляется в режиме конструктора, вызов свойств объектов осуществляется кнопкой *Свойства* в группе *Элементы управления* или через контекстное меню.

Слияние документов. Подготовка документов к рассылке

Часто приходится рассылать один и тот же документ по разным адресам, например, приглашение на юбилей учебного заведения, симпозиум и др. Текстовый редактор Word позволяет автоматизировать эту работу. Для подготовки к рассылке документа, оформления конвертов и наклеек имеется Мастер слияния документов. Для выполнения операции слияния необходимо подготовить шаблон документа с полями слияния и источник данных для заполнения этих

полей. Шаблон документа может быть подготовлен заранее и сохранен на диске как документ Word или создан в процессе выполнения работы. Источник данных представляет собой обычную таблицу. Таблица может быть подготовлена средствами Word или другими приложениями, например, с помощью электронной таблицы, системы управления базой данных и др. Каждая строка таблицы является записью, относящейся к одному объекту, а каждая ячейка – элементом данных. Выполнение операций по подготовке документов к рассылке осуществляется с помощью вкладки *Рассылка*.

Макрокоманды

Макрокоманды, или макросы, представляют собой набор команд, с помощью которых можно автоматизировать выполнение повторяющейся задачи. Макросы позволяют легко и быстро выполнять часто повторяющиеся действия и тем самым повышать производительность труда пользователя. Прикладные приложения Макрокоманды хранятся в шаблонах. Выбором шаблона для сохранения ограничивается круг документов, в которых можно использовать ту или иную макрокоманду. Место хранения макроса указывается при его создании.

Макрокоманда может быть написана с использованием встроенного языка программирования Visual Basic или записана с помощью средств записи программы Word. Макрокоманда имеет имя. Оно может содержать до 36 символов, в имени не допускается использование пробелов. Макрокоманде можно назначить комбинацию клавиш или кнопку на панели инструментов. Запись макроса средствами записи программы Word. Проще всего создать макрос с помощью встроенных средств записи. Продумайте последовательность действий, необходимых для выполнения нужной операции, а затем повторите их в режиме записи макроса. Это необходимо делать всегда, так как в макрос будут записаны все команды, в том числе и ошибочные.

Запись макроса с помощью встроенного языка программирования Visual Basic

Для записи макроса с помощью встроенного языка программирования Visual Basic введите команду Visual Basic в группе *Код* вкладки *Разработчик* – открывается окно разработки проекта. Если нет вкладки *Разработчик*, то выполните следующие действия: введите команду *Файл, Параметры, Настройка* ленты и в списке

Основные вкладки установите флажок Разработчик. Введите команду *Insert, Module* – открывается окно разработки программы *Module1*. Введите команду Insert, *Procedur*e – открывается окно добавления процедуры *Add Procedure*. Введите имя процедуры, например, *Codirovk*a, установите переключатели *Sub* – ключевое слово заголовка процедуры и Public – общая (процедура будет доступна всем формам проекта) и щелкните по кнопке ОК – программа возвращается в окно проекта Module1. В этом окне появится шаблон процедуры, две строки команд, между которыми необходимо записать текст программы:

Public Sub Codirovka()

‘Текст программы

End Sub

Между командами Public Sub и End Sub записывается или копируется туда текст необходимой программы макроса.

Контрольные вопросы

1. Назовите наиболее распространенные текстовые редакторы и перечислите их особенности.
2. Назовите основные элементы окна текстового редактора Microsoft Word.
3. Назовите основные вкладки ленты.
4. Как выполнить настройку ленты?
5. Как добавить кнопки на Панель быстрого доступа?
6. Поясните назначение кнопок в строке состояния.
7. Как сохранить документ на диске?
8. Как вывести документ на печать?
9. Перечислите основные элементы форматирования текста.
10. Поясните назначение кнопок в группе Шрифт ленты.
11. Поясните назначение кнопок в группе Абзац ленты.
12. Какие параметры абзацев Вы знаете, как их установить?
13. Как создать маркированный, нумерованный, иерархический список?
14. Как вставить в документ картинку, дату, сноску, номера страниц?
15. Как вставить в документ формулу? Как используется мастер формул?
16. Какие виды колонтитулов можно установить в документе?
17. Расскажите алгоритм вставки оглавления.

18. Как нарисовать схему алгоритма в документе Word?
19. Как вставить в документ перекрестную ссылку, гиперссылку? Перечислите команды создания таблиц.
20. Что такое шаблон? Для какой цели он создается?
21. Опишите алгоритм создания шаблона.
22. Какие поля можно использовать в шаблоне и как они вставляются?
23. Как настроить свойства (параметры) полей формы?
24. Какие элементы используются для создания полей в шаблоне?
25. Что такое макрос? Как его создать?
26. Как используются макросы?
27. Как назначить макрос кнопке панели инструментов?
28. Как назначить макросу комбинацию клавиш?
29. Как написать макрос с помощью встроенного языка программирования?

2.2. Графические редакторы

С развитием информационных технологий пользователи получили возможность создавать работы требующие внедрения графических объектов. Для этих целей разработаны специальные программные средства – графические редакторы, позволяющие удовлетворить потребности работы с графикой как опытных пользователей, так и непрофессионалов.

В настоящее время активно применяются программы Adobe PhotoShop (цветоделение и обработка изображений), Quark Press (верстка периодики), CorelDraw (графический редактор), *PowerPoint* (разработка сценария и стиля презентаций, слайд-фильмы), FaxLine (факсовая связь), Machaon (факсимильная и почтовая связь и безбумажный документообмен), AutoCAD (черчение и конструирование), Adobe Illustrator (дизайнерство), Corel ArtShow (библиотека иллюстраций, созданных художниками всего мира), всемирно известные браузеры Интернет Explorer и Netscape Navigator, в которых используются графические редакторы, Microsoft Paint – многофункциональный, но в то же время простой в использовании растровый графический редактор компании Microsoft, входящий в состав всех операционных систем Windows, начиная с первых версий.

Существует специальная область информатики, изучающая методы и средства создания и обработки изображений с помощью программных и аппаратных вычислительных комплексов – компьютерная графика. Она охватывает все виды и формы представления изображений, доступных для восприятия человеком либо на экране монитора, либо в виде копии из внешнего носителя.

В зависимости от способа формирования изображений компьютерную графику принято подразделять на растровую, векторную и фрактальную.

Растровая графика применяется при разработке электронных (мультемедийных) и полиграфических изданий. Образует изображение множеством точек (пикселей), с каждой из которых можно работать отдельно (в том числе и закрашивать в определенный цвет). Большинство графических редакторов предназначены для работы с растровыми изображениями, ориентированы не столько на создание изображений, сколько на их обработку (*MS Paint*). В Интернете применяются только растровые иллюстрации. Одним из недостатков растровой графики является снижение качества изображений при их увеличении. Раз в оригинале предусмотрено определенное количество точек, то при большом масштабе увеличивается их размер, становятся заметны элементы растра, что искажает саму иллюстрацию.

Векторная графика образует изображение системой отдельных объектов, которыми могут быть различные геометрические фигуры, составленные из прямых, дуг, окружностей. Если в растровой графике базовым элементом изображения является точка, то в векторной графике – линия (вектор). Линия описывается математически как единый объект, и потому объем данных для отображения объекта средствами векторной графики существенно меньше, чем в растровой графике. Из простейших объектов создаются более сложные. Программные средства для работы с векторной графикой предназначены в первую очередь для создания иллюстраций и в меньшей степени для их обработки. Такие средства широко используются в рекламных средствах и издательствах, при создании чертежей и карт. К редакторам векторного типа относятся графический редактор в приложении Microsoft Office (Word, Excel). Некоторые программы, например Adobe PhotoShop, позволяют комбинировать растровые и векторные технологии.

Фрактальная графика очень похожа на векторный способ отображения графической информации. Фрактальная графика как и векторная базируется на математических вычислениях. Однако базовым элементом фрактальной графики является сама математическая формула, т. е. изображение строится исключительно по уравнениям. Таким способом строят как простейшие регулярные изображения так и сложные иллюстрации, имитирующие, например, природный ландшафт.

Отдельным предметом считается ***трехмерная (3D) графика*,** реализующая приемы и методы построения объемных моделей объектов в пространстве. Как правило, в ней сочетаются векторный и растровые методы формирования изображений. Популярным пакетом обработки трехмерной графики является 3D Studio Max.

Основные форматы графических файлов: BMP – стандартный формат растровых изображений воспринимается всеми графическими редакторами; GIF – распространенный формат, получил популярность благодаря высокой степени сжатия; форматы JPEG и PCX.

Растровый графический редактор компании Microsoft Paint

Microsoft Paint – многофункциональный, но в то же время простой в использовании растровый, графический редактор компании Microsoft, входящий в состав всех операционных систем Windows, начиная с первых версий.

Первая версия Paint появилась в Windows 1.0. В Windows 3.0 был переименован в PaintBrush, но позже был опять переименован в Paint.

В Windows 7 Paint впервые был полностью переработан, получил ленточный интерфейс, дополнительные кисти и фигуры, схожие с библиотекой Microsoft Office. Краткий обзор нововведений:
9 разновидностей кисти (Brush).

Обновилась библиотека фигур: к стандартным эллипсу, прямоугольнику, вектору, кривой, многограннику и скругленному прямоугольнику добавилось еще 17 типовых фигур, среди которых: треугольник равнобедренный, треугольник прямоугольный, ромб, пяти- и шестиугольник, стрелки вправо, влево, вверх и вниз; звезды: четырех-, пяти- и шестиугольная; прямоугольный, круглый и «думающие» пузыри для комиксов, сердце и молния.

Нарисовав фигуру, можно еще настроить ее параметры: повернуть, растянуть, изменить цвет и фактуру.

7 разновидностей заливки/контура.

Также в меню «Вид» добавлены: новая линейка, режим предпросмотра печати.

Возможность получения материала для редактирования со сканера.

Теперь возможно использовать разные стили для каждого фрагмента текста внутри одной рамки.

Многократное увеличение или уменьшение инструмента

Выберите один из инструментов «Кисть», «Ластик», «Линия» или «Распылитель» и нажмите клавиши *Ctrl* и *NumPad* +. Чем дольше держать нажатой эту комбинацию, тем больше будет увеличиваться инструмент. Соответственно, если зажать *Ctrl* и *NumPad* −, то инструмент будет уменьшаться.

Пипетка

Инструмент «одноразового» действия – после применения автоматически возвращает тот инструмент, который был активен до ее включения. Левой кнопкой берет основной цвет, правой – фоновый. С нажатым *Ctrl* берет «третий» цвет.

Заливка

В связи с тем, что в Paint не используется полупрозрачность, заливка аккуратно и четко заполняет области, обведенные любыми кривыми линиями. Пользуясь заливкой и пипеткой, можно быстро стереть множество деталей одного цвета на фоне другого – достаточно залить этот фон цветом деталей, а затем вернуть ему его цвет.

Замена цвета

Инструмент «Ластик» работает, фактически рисуя вторым – «фоновым» – цветом там, где им проведут при нажатой левой кнопке мыши. Однако если им водить при нажатой правой кнопке, то он будет «стирать» фоновым цветом только то, что нарисовано первым – «основным» – цветом.

Выделение

Выделенный фрагмент оказывается «плавающим» (он может быть перенесен в любое место рабочей области без изменения самой картинки), а его место заполняется фоновым цветом. При этом, если в момент начала перетаскивания нажата клавиша Ctrl, в начальной позиции остается «штамп» – туда впечатывается копия плавающего

выделения (при первоначальном перемещении получается так, как будто унесена копия выделенного, а на исходном месте ничего не изменилось). Если нажата клавиша Shift – то подобный штамп делается и во всех промежуточных точках перемещения.

Прозрачное выделение

Прозрачным считается цвет, который в момент выделения назначен фоновым (назначается правой кнопкой мыши как с палитры, так и с рабочей области инструментом взятия цвета (пипеткой)).

Пользовательская кисть

Сделайте сбоку от своего изображения свободное поле, отодвинув в сторону правый или нижний край полотна за имеющийся на нем маркер (добавленная область будет закрашена «фоновым» цветом). Нарисуйте там, на фоновом цвете, картинку, которую хотите использовать как кисть. Выделите эту часть изображения в режиме прозрачного фона и, зажав Ctrl чтобы оставить ее копию для следующих применений, перенесите выделение в то место, где должен начаться штрих такой кистью. Теперь зажмите клавишу Shift и перемещайте выделенный фрагмент. Он будет оставлять след так же, как оставляет след инструмент кисть (можно сказать, что шлейф от изображения примет вид карточной колоды).

Вставка

При вставке размер рабочей области увеличивается так, чтобы вмещать в себя вставляемое изображение. Пользуясь этим, можно измерять размеры изображений: достаточно перед вставкой уменьшить за нижний правый угол размеры рабочей области до минимальных, а после вставки посмотреть на «атрибуты» изображения.

Графический редактор Adobe Photoshop

Adobe Photoshop – многофункциональный графический редактор, разработанный фирмой *Adobe Systems*. В основном работает с растровыми изображениями, однако имеет некоторые векторные инструменты. Часто эту программу называют просто Photoshop. В настоящее время Photoshop доступен на платформах Windows, OS X в мобильных системах iOS и Android. Для версий 8.0 и CS6 возможен запуск под Linux с помощью альтернативы Windows API – Wine.

Несмотря на то, что изначально программа была разработана как редактор изображений для полиграфии, в данное время она широко используется и в web-дизайне. Photoshop тесно связан с другими программами для обработки медиафайлов, анимации и другого творчества. Photoshop CS3 в версии Extended поддерживает также работу с трехмерными слоями.

Первая версия Photoshop появилась в 1987 г. Ее создал студент Мичиганского университета Томас Нолл (англ. Thomas Knoll) для платформы Macintosh.

Дальнейшее развитие программа получила в коммерческой версии Photoshop CS3, которая была разработана в апреле 2007 г. Список нововведений включает в себя новый интерфейс, увеличенную скорость работы, новые фильтры и инструменты, а также приложение, позволяющее осуществлять предварительный просмотр работы в шаблонах популярных устройств, например мобильных телефонов.

В программе Adobe Photoshop Extended можно открывать и работать с 3D-файлами, создаваемыми другими программами. Возможно использовать трехмерные файлы для внедрения в двумерное фото. Доступны некоторые операции для обработки 3D-модели, такие как работа с каркасами, выбор материалов из текстурных карт, настройка света.

Также можно создавать надписи на 3D-объекте, вращать модели, изменять их размер и положение в пространстве. Программа включает в себя также команды по преобразованию плоских фотографий в трехмерные объекты определенной формы, например, пирамида, цилиндр, сфера, конус и др.

Для имитации движения в Photoshop можно создавать кадры мультипликации, используя слои изображения. Можно создавать видеоизображения, основанные на одной из многих заданных пиксельных пропорций. После редактирования можно сохранить работу в виде файла GIF-анимации или PSD, который впоследствии можно проиграть в других видеопрограммах, таких как Adobe Premiere Pro или Adobe After Effects. Доступно открытие или импортирование видеофайлов и последовательности изображений для редактирования и ретуширования, создание видеоряда мультипликации и экспорт работ в файл формата QuickTime, GIF-анимацию или последовательность изображений. Видеокадры можно отдельно редактировать, трансформировать, клонировать, применять

к ним маски, фильтры, разные способы наложения пикселов, на них можно рисовать, используя различные инструменты.

С помощью программы Photoshop Extended можно рассматривать MatLab-изображения, обрабатывать их в программе PhotoShop, комбинировать команды MatLab с технологиями обработки изображений PhotoShop. Как только устанавливается соединение с программой PhotoShop из программы MatLab и осуществляется ввод команд в командную строку MatLab, эти управляющие воздействия незамедлительно выполняются в *PhotoShop*. Файлы, подготовленные в программе MatLab, имеют расширение *.m, *.fig, *.rpt, *.mat, *.mdl. Коммуникация между PhotoShop и MatLab использует интерфейс PhotoShop JavaScript и библиотечный интерфейс MatLab.

Графический редактор CorelDRAW

Программа *CorelDRAW*, разработанная фирмой *Corel*, была выпущена в 2002 г. Эта программа обладает удивительной универсальностью и мощностью, будучи в равной степени полезной и в промышленном дизайне, и в разработке рекламной продукции, и в подготовке публикаций, и в создании изображений для web-страниц. Это делает программу весьма эффективной в качестве первого программного средства для приступающих к изучению машинной графики в целом или векторной графики в частности. Пользовательский интерфейс *CorelDRAW* построен очень рационально, с высокой степенью унификации. Все изображения, с которыми работает программа, разделяются на два класса: точечные и векторные.

Объектно-ориентированный подход

CorelDRAW представляет собой интегрированный объектно-ориентированный пакет программ для работы с *иллюстративной графикой*. Интегрированностъ пакета следует понимать в том смысле, что входящие в него программы могут легко обмениваться данными или последовательно выполнять различные действия над одними и теми же данными.

Иллюстративная графика – это прикладная ветвь машинной графики, сравнительно недавно выделившаяся в отдельное направление

наряду с графикой деловой, научной и инженерной. К области иллюстративной графики относятся в первую очередь рисунки, коллажи, рекламные объявления, заставки, постеры – все, что принято называть художественной продукцией. Объекты иллюстративной графики отличаются от объектов других прикладных областей своей первичностью: они не могут быть построены автоматически по некоторым исходным данным, без участия художника или дизайнера. В отличие от них такие графические изображения, как диаграммы (деловая графика), чертежи и схемы (инженерная графика), графики функций (научная графика), представляют собой лишь графический способ представления первичных исходных данных – как правило, таблицы (или аналитической модели, представленной в другой форме). В этом состоит их вторичность, производность.

Особенность объектной ориентации пакета состоит в том, что все операции, выполняющиеся в процессе создания и изменения изображений, пользователь проводит не с изображением в целом и не с его мельчайшими, атомарными частицами (пикселами точечного изображения), а с объектами – семантически нагруженными элементами изображения. Начиная со стандартных объектов (кругов, прямоугольников, текстов и т. д.), пользователь может строить составные объекты (например, значок в рассмотренном выше примере) и манипулировать ими как единым целым. Таким образом, изображение становится иерархической структурой, на самом верху которой находится иллюстрация в целом, а в самом низу – стандартные объекты.

Вторая особенность объектной ориентации пакета состоит в том, что каждому стандартному классу объектов ставится в соответствие уникальная совокупность управляющих параметров или атрибутов класса.

Третья особенность объектной ориентации пакета состоит в том, что для каждого стандартного класса объектов определен перечень стандартных операций. Например, описанный выше прямоугольник можно развернуть, масштабировать, закруглить ему углы, преобразовать его в объект другого класса – замкнутую кривую.

Контрольные вопросы

1. Какой растровый графический редактор входит в состав всех операционных систем Windows, начиная с первых версий?

2. Можно ли применять графический редактор Adobe Photoshop для создания изображений web-страниц?
3. В чем отличие объектов иллюстративной графики от объектов других прикладных областей?
4. На какие виды графики подразделяется компьютерная графика в зависимости от способа формирования изображений?
5. Позволяет ли графический редактор Adobe Photoshop комбинировать растровые и векторные технологии?
6. На какие два класса разделяются все изображения, с которыми работает программа CorelDRAW?
7. В чем заключается объектная ориентация пакета CorelDraw?

2.3. Электронные таблицы

Электронная таблица Microsoft Excel

Электронная таблица – интерактивная система обработки информации, упорядоченной в виде таблицы с поименованными строками и столбцами. *Табличный процессор* − категория прикладного программного обеспечения, предназначенного для работы с электронными таблицами. Инструментарий электронных таблиц включает мощные математические функции, позволяющие вести сложные инженерные, финансовые, статистические и прочие расчеты. Одними из самых популярных табличных процессоров сегодня являются Microsoft Excel, входящий в состав пакета Microsoft Office, и Open Office Calc, входящий в состав открытого для доступа пакета Open Office.

Для загрузки программы Microsoft Excel следует щелкнить мышкой по значку приложения Microsoft Excel на рабочем столе или панели задач операционной системы Microsoft Windows 7. Для выхода из программы введите команду ***Файл, Выход***.

Рабочее окно Microsoft Excel представляет собой стандартное окно Microsoft Office со специфическими элементами. В верхней части окна расположена строка заголовка, в которой располагаются кнопка системного меню, название приложения и имя файла, загруженного в окно (в начале работы по умолчанию выводится имя файла *Книга1*), кнопки свертывания и развертывания окна программы. Ниже распо-

ложены: кнопка меню **Файл**, лента, панель быстрого доступа, строка ввода данных, рабочее окно, строка состояния.

Лента имеет ту же структуру, что и лента *Microsoft Word 2010*. Она имеет восемь постоянно открытых вкладок, ярлычки которых видны на экране: ***Главная, Вставка, Разметка страницы, Формулы, Данные, Рецензирование, Вид и Надстройки.*** Каждая вкладка состоит из групп команд, в которых расположены кнопки управления. У большинства групп в правом нижнем углу имеется кнопка, которая вызывает окно диалога настройки параметров. Для увеличения рабочей области ленту можно свернуть соответствующей командой контекстного меню или комбинацией клавиш ***Ctrl + F1***. По умолчанию на экран не выводится одна важная вкладка ленты – ***Разработчик***. Чтобы добавить ее на ленту, выполните следующее: введите команду ***Файл, Параметры, Настройка*** ленты, в окне Основные вкладки установите флажок *Разработчик* и щелкните по кнопке ОК.

Панель быстрого доступа содержит команды, используемые наиболее часто, состав этих команд может настраиваться пользователем. Положение панели быстрого доступа может быть определено пользователем: над лентой или под лентой. Для изменения положения панели быстрого доступа воспользуйтесь контекстным меню.

Строка формул имеет три поля: поле адреса ячейки (Имя), поле управляющих клавиш и поле ввода данных (Строка формул). Поле *Имя* представляет собой раскрывающийся список, в нем указан адрес текущей ячейки или ее имя. Если щелкнуть мышкой по этому полю, ввести адрес ячейки и нажать клавишу Enter, то курсор электронной таблицы перейдет в указанную ячейку. Этот прием перехода к нужной ячейке называется непосредственной адресацией.

В поле управляющих кнопок выводятся три кнопки: × – отмена редактирования строки ввода, ν – окончание редактирования, f_x – ввод функций. Названные кнопки появляются при активизации строки ввода. Поле *Ввод данных* предназначено для отображения вводимой информации или содержания выделенной ячейки. Для активизации строки ввода необходимо щелкнуть по ней мышью.

Строка состояния расположена в нижней части рабочего окна. В левой части строки состояния выводится информация о текущем состоянии электронной таблицы. В правой части расположены кнопки управления режимами просмотра таблицы: обычный,

разметка страницы и страничный, а также элементы управления масштабом представления информации. Строка состояния имеет контекстное меню, которое позволяет управлять представлением информации в строке состояния. Над строкой состояния размещены кнопки навигации, ярлычки листов, специальная кнопка ***Вставить лист*** и горизонтальная линейка прокрутки.

Рабочая книга, рабочий лист

Информация в электронной таблице сохраняется в виде рабочих книг. Имя книги выводится в строке заголовка. Рабочая книга состоит из листов различного типа. Максимально возможное число листов в рабочей книге – 256. Рабочий лист состоит из пронумерованных строк и столбцов. Столбцы рабочих листов озаглавлены латинскими буквами от A до Z и их комбинациями по два и три символа, например, AA, AB, IU, … , XFD. Строки пронумерованы цифрами.

Рабочий лист содержит 16 385 столбца и 1 058 576 строк. На пересечении строк и столбцов образованы ячейки. В одной из ячеек расположен контур выделения – курсор электронной таблицы. Рабочий лист имеет номер, который указан на ярлыке. Если щелкнуть правой кнопкой мыши по ярлыку, то откроется контекстное меню с перечнем команд для управления рабочим листом. Рабочие листы можно добавлять, удалять, копировать, переименовывать, перемещать, группировать, разгруппировывать.

Ячейка и ее свойства

Ячейка является основным элементом таблицы. В качестве содержания ячейки выступают числовые и текстовые константы, а также выражения (формулы).

Ячейка – область, образованная пересечением строки и столбца. Она обозначается номером столбца и строки, на пересечении которых находится. Например, C1, AV9999.

Диапазон (группа) – непрерывная область ячеек, обозначенная номерами начальной и конечной ячеек, разделенными двоеточием или точкой, например, A1:C10, D8.H12. Ячейке или диапазону может быть присвоено уникальное имя. Ячейка характеризуется следующими параметрами: адрес, содержание, значение, формат, статус.

Адрес ячейки

Адрес ячейки может быть задан абсолютным, относительным и смешанным.

Относительный адрес: A1, E7. Относительный адрес в операциях копирования автоматически настраивается.

Абсолютный адрес: A1, E7. Абсолютный адрес ячейки не меняется в операциях копирования, вставки или удаления ячеек, строк и столбцов.

Смешанный адрес: $A1, A$1. Если ячейке присвоен смешанный адрес, то при копировании будет меняться только тот параметр, перед которым не стоит знак $. Например: $D6 – при копировании ячейки будет меняться только номер строки; D$6 – при копировании будет меняться только адрес столбца.

Присвоение имени ячейке

Ячейке или диапазону ячеек может быть присвоено имя. В Microsoft Excel 2010 для работы с именами создана отдельная группа – ***Определенные имена*** на вкладке ***Формулы***. Присвоение имени осуществляется командой ***Присвоить имя***.

Для присвоения имени ячейке или диапазону ячеек необходимо:

1. Выделить ячейку (диапазон ячеек).
2. Ввести команду ***Формула***, ***Присвоить имя***.
3. Выбрать область действия имен из списка Область: книга или лист. Ввести в строке ввода Имя диалогового окна Создание имени имя ячейки и щелкнуть по кнопке **ОК**. Область, которой присваивается имя, отображается в строке *Диапазон*.

Содержание ячейки

Содержание ячейки – это то, что вводится в нее через строку ввода. Поэтому ячейка может либо быть пустой, либо содержать данные: текст, текстовую константу, формулу, дату, время.

Значение ячейки

Значением ячейки могут быть число, текстовая константа, дата, время, сообщения об ошибках. Значением пустой ячейки и ячейки, содержащей текст, является ноль.

Текстовая константа – это строка символов, используемая в выражениях как операнд, при вводе текстовой константы она заключается в скобки и в кавычки.

Дата – значение функции дата.

Время – значение функции время.

Сообщения об ошибках:

#ДЕЛ/0! – деление на ноль;

#ИМЯ? – не определено имя переменной в формуле;

#Н/Д! – нет допустимых значений, аргумент функции не может быть определен;

#ЗНАЧ! – неправильный тип аргумента; например, использование текста там, где необходимо число и др.

Формат ячейки

К формату ячейки относятся такие параметры ячейки, как ширина, режим отображения формул, формат отображения числовых величин, размещение содержимого ячейки, шрифт, цвет, границы, статус ячейки. Все основные параметры настройки свойств ячейки в Excel 2010 вынесены на вкладку *Главная* ленты.

Статус ячейки

Ячейка может иметь два статуса: защищена или не защищена. В защищенную ячейку нельзя внести информацию или изменить ее содержание. Установка режима защиты осуществляется командами вкладки *Защита* окна диалога *Формат ячеек*. Эта команда позволяет установить защиту на ячейку или скрыть формулу. По умолчанию для всех ячеек установлен режим защиты. Режим защиты ячейки вступает в силу только после защиты листа командами *Защитить лист* из группы *Изменения* вкладки *Рецензирование*. Для отмены защиты ячейки достаточно отменить защиту листа командой *Снять защиту листа*. Можно защитить также структуру книги.

Ввод данных

Данные вводятся в *Строку ввода данных* или непосредственно в ячейку. В первом случае выделите ячейку, в которую вводятся данные, и щелкните по *Строке ввода данных*. Введите нужную информацию. Для окончания ввода нажмите клавишу Enter или щелкните кнопку ν Строки ввода данных. Во втором случае выделите ячейку и вводите данные прямо в ячейку. По окончании ввода данных нажмите клавишу *Enter*. Для очистки ячейки выделите ее и нажмите клавишу Delete или Space и Enter.

Курсор таблицы, или контур выделения, представляет собой рамку, окаймляющую всю ячейку. В правом нижнем углу рамки на пересечении сторон располагается маленький квадрат – *маркер заполнения*. Этот маркер используется для заполнения ячеек рядом данных с постоянным шагом, а также для копирования формул. Для перемещения курсора используются клавиши управления перемещением курсора, а также клавиши Home – перейти в первую ячейку строки, [Ctrl + Home] – перейти в ячейку A1, [End + клавиши

управления перемещением курсора] – последняя занятая ячейка в соответствующем направлении, используются также клавиши прокрутки и ряд других комбинаций клавиш. Непосредственная адресация осуществляется вводом адреса ячейки в поле «Адрес ячейки» Строки ввода данных.

Ввод текста. Признаком текста при вводе данных является апостроф ('), например, 'Исходные данные. По умолчанию вводимые данные в этом случае воспринимаются как текст.

Ввод даты. Дата вводится в формате ДД.ММ.ГГ или ДД.ММ.ГГГГ: день, месяц, год (17.05.14). В качестве разделителя используется точка. Электронная таблица позволяет выводить дату на экран в различных форматах.

Ввод текстовых констант. Для ввода текстовых констант необходимо ввести символ = и текст в кавычках: = «Текст». Для преобразования чисел или числовых значений выражений в текстовые константы следует воспользоваться функцией ТЕКСТ(). Тип данных в ячейке определяется при первом вводе.

Ввод формул. Признаком формулы является знак =. Если при вводе формулы допущена ошибка, то программа выдает сообщение об ошибке. При вводе формулы без знака «равно» программа воспринимает вводимые данные как текст. Адреса ячеек вводятся только латинскими символами. При вводе вещественных чисел, используется десятичная запятая, а не точка. Для ввода формул или ознакомления с функциями Excel можно использовать *Мастера функций*. Для этого необходимо щелкнуть кнопку f_x в стандартной панели инструментов или воспользоваться вкладкой *Формулы* ленты.

Примеры записи формул:

=A2+2 – сложение;

=ЕСЛИ(A2<B2;C3;D2*E17) – условное выражение.

Если значение в ячейке A2 меньше значения в ячейке B2, то результат будет равен значению ячейки C3, иначе – произведению значений ячеек D2 и E17.

При записи формул, для указания адреса ячеек, значения которых не должны изменяться при копировании формул, следует обязательно использовать абсолютный адрес.

Формула может содержать ссылки на ячейки таблицы, расположенные в том числе и на другом рабочем листе.

Методика работы с электронной таблицей:

– Выбор ячейки. Установите курсор на ячейку или щелкните по ней мышью.

– Выбор группы ячеек. Установите указатель мыши на первую ячейку группы, нажмите левую клавишу и протащите указатель по всем ячейкам группы. В конце области выделения отпустите клавишу мыши.

– Для выбора нескольких несвязанных областей необходимо нажать и удерживать клавишу *Ctrl*, а затем выделить требуемые области.

– Для отмены выделения ячеек щелкните мышью по чистому полю в любом месте экрана.

– Выбор строк и столбцов. Для выбора одной строки (столбца) щелкните мышью по номеру строки (столбца). Для выбора группы строк (столбцов) установите указатель мыши на номер первой строки (столбца), нажмите клавишу мыши и протащите ее указатель по всем строкам (столбцам) выделяемой группы. Отпустите клавишу мыши.

Копирование ячеек

При копировании ячеек можно использовать команды ***Копировать*** и ***Вставить*** из группы ***Буфер обмена*** вкладки ***Главная*** ленты, команды контекстного меню.

Копирование с использованием маркера автозаполнения. Если копирование осуществляется в соседние ячейки, то его удобно выполнить с использованием *маркера автозаполнения*: выделите копируемую ячейку; зацепите мышью за маркер автозаполнения (подведите курсор к черному квадратику в правом нижнем углу курсора таблицы так, чтобы указатель мыши превратился в черный крестик) и протащите указатель мыши по всем ячейкам назначения.

Копирование с помощью команд группы Буфер обмена или контекстного меню. Для копирования с помощью команд группы ***Буфер обмена*** необходимо: выделить ячейку или блок ячеек, подлежащих копированию; выбрать команду *Копировать*. Вокруг выделенной ячейки появляется мерцающий контур; выделить начальную ячейку, куда будут копироваться данные; выбрать команду *Вставить*; выбрать подходящую команду из отобразившегося меню.

Копирование с помощью мыши. Для копирования содержимого ячеек, содержащих текстовую информацию или формулы с абсолютными адресами данных, выделите с помощью мыши ячейку или блок ячеек, подлежащих копированию, нажмите и удерживайте

клавишу Ctrl, перетащите выделенные ячейки на новое место, отпустите кнопку мыши, а затем отпустите кнопку Ctrl.

Для копирования формул, содержащих ссылки на ячейки с относительными или смешанными адресами, следует воспользоваться командой ***Вставить***, ***Специальная вставка***: выделите копируемые ячейки; укажите место вставки; введите команду ***Вставить***, ***Специальная вставка*** из группы ***Буфер обмена*** и щелкните по кнопке ***Вставить связь***. В этом случае в формуле сохраняются ссылки на ячейки, содержащие данные. При изменении данных в ячейках автоматически будут меняться значения в ячейках источника данных и в ячейках назначения.

Оформление таблицы

Для оформления таблицы: обрамления, заливки цветом – можно воспользоваться вкладками ***Граница*** и ***Заливка*** окна диалога ***Формат ячеек*** или одноименными кнопками в группе ***Шрифт*** вкладки ***Главная***. Можно воспользоваться также командами ***Форматировать как таблицу*** и ***Стили ячеек*** из группы ***Стили*** вкладки ***Главная***.

Предварительный просмотр. Настройка параметров страниц. Для просмотра таблицы введите команду ***Разметка страницы*** в группе ***Режимы просмотра книги*** вкладки ***Вид***. В этом режиме лист представляется в том виде, как будет выглядеть при печати, можно просмотреть начало и конец страниц, верхний и нижний колонтитулы, на экран выводятся горизонтальная и вертикальная линейки. Для изменения параметров страницы откройте вкладку ***Разметка страницы*** и обратитесь к группе ***Параметры страницы***.

Сохранение, открытие и печать таблицы. Перед печатью целесообразно сохранить документ на диске, для этого необходимо выполнить следующее: введите команду ***Сохранить*** или ***Сохранить как***... из меню ***Файл***, укажите в строке ввода «Имя Файла» имя файла, при необходимости измените тип файла .xlsx на .xls для совместимости с предыдущими версиями программы Excel. Выберите диск и папку, щелкните по кнопке ***ОК***. Открытие сохраненного ранее документа осуществляется командой ***Файл, Открыть***. Выберите соответствующий диск, папку.

Настройка параметров таблицы. Приступая к работе с электронной таблицей, полезно ознакомиться с некоторыми настройками. Настройка параметров электронной таблицы осуществляется командой ***Файл, Параметры***. После ввода команды открывается окно диалога ***Параметры Excel.*** Команды собраны в отдельные группы: язык, интервал времени, через который рабочая книга будет сохраняться и др.

Разработка электронных таблиц

Категории функций электронной таблицы

Для удобства вычисления в табличных процессорах имеются встроенные функции (11 категорий): математические/тригонометрические, инженерные, текстовые, статистические, финансовые, даты и времени, логические, функции для работы с базами данных/списками, информационные и функции категории ссылки/массивы; функции проверки свойств и значений и др. Кроме того, Excel содержит большое число надстроечных функций, которые используются для создания компьютерных программ в Excel, а также имеется возможность создания пользовательских функций и программ на Visual Basic for Applications. Можно написать программы на других языках программирования высокого уровня, например, C, FORTRAN, и потом вызвать их в Excel. Вызов функций осуществляется с помощью кнопки f_x строки ввода данных или команд вкладки ***Формулы***.

В группе **Библиотека функций** вкладки ***Формулы*** размещены списки категорий функций, что позволяет легко находить нужные функции. Список Автосумма содержит функции Сумма, Среднее, Максимум, Минимум, Число. Команда Число вызывает функцию Счет, которая подсчитывает число непустых ячеек в выделенной области.

Каждый список содержит команду ***Вставить функцию***, аналогичная команда имеется в группе ***Библиотека функций***. Эта команда вызывает окно диалога ***Мастер функций***. Мастер функций содержит окно для поиска функции по ее краткому описанию, список категорий функций и окно выбора функции. При создании функций пользователя в списке категорий появляется группа ***Пользовательские***.

Генерирование данных

Часто бывает необходимо сгенерировать последовательность чисел, дат. Для этой цели можно использовать механизм автозаполнения. Чтобы заполнить несколько ячеек прогрессией, необходимо

записать в смежные ячейки данные, отличающиеся на величину шага, выделить эти ячейки и перетащить маркер автозаполнения выделенного диапазона ячеек. Можно также воспользоваться командой *Заполнить* из группы ***Редактирование*** вкладки ***Главная***. Данная команда имеет подменю: *влево, вправо, вверх, вниз, прогрессия, выровнять*.

Команды *Влево, Вправо, Вверх, Вниз* позволяют заполнить выделенные ячейки одинаковыми данными или узором.

Команда ***Выровнять*** позволяет заполнить ячейку узором по ширине.

Команда ***Прогрессия*** позволяет заполнять ячейки рядами чисел и дат.

Если ввести в ячейки выделенного диапазона начальное и конечное значения ряда чисел, то шаг прогрессии определится автоматически.

Графические возможности электронных таблиц

Виды иллюстраций деловой графики

Табличные процессоры предлагают различные виды *иллюстраций деловой графики* (диаграмм), причем их построение облегчено за счет использования «*Мастера диаграмм*» − встроенных автоматизированных пошаговых процедур, позволяющих выбрать тип диаграммы и для него выполнить все необходимые операции, в том числе оформления различными компонентами.

Гистограмма показывает изменение данных за определенный период времени и иллюстрирует соотношение отдельных их значений. Категории располагаются по горизонтали, а значения − по вертикали. Гистограмма с накоплением демонстрирует вклад отдельных элементов в общую сумму.

Линейчатая диаграмма отражает соотношение отдельных компонентов. Категории расположены по горизонтали, а значения по вертикали. Она ориентирована на сопоставление значений. Линейчатая диаграмма с накоплением демонстрирует вклад отдельных элементов в общую сумму.

График представляет варианты отображений изменений данных за равные промежутки времени.

Круговая диаграмма отражает как абсолютную величину каждого элемента ряда данных, так и его вклад в общую сумму. На круговой диаграмме может быть представлен только один ряд данных. Такую диаграмму рекомендуется использовать, когда необходимо подчеркнуть какой-либо значительный элемент.

Точечная диаграмма показывает взаимосвязь между числовыми значениями в нескольких рядах и представляет две группы чисел в виде одного ряда точек в координатах *x* и *y*. Она отображает нечетные интервалы данных и часто используется для предоставления данных научного характера. При подготовке данных следует расположить в одной строке или столбце все значения переменной *x*, а соответствующие значения *y* – в смежных строках или столбцах.

Поверхностная диаграмма используется для поиска наилучшего сочетания двух наборов данных. Так, на топографической карте области с одним значением выделяются одинаковым узором и шрифтом.

Кольцевая диаграмма, как и круговая диаграмма, показывает вклад каждого элемента в общую сумму, но в отличие от круговой диаграммы может содержать несколько рядов данных.

Биржевая диаграмма часто используется для демонстрации цен на акции. Этот тип диаграммы применяется для отображения научных данных, например, изменения температуры. Существуют и другие виды диаграмм.

Построение графиков и диаграмм

Для построения графиков и диаграмм в электронной таблице используется группа ***Диаграммы*** вкладки ***Вставка***, а также вкладки ***Конструктор, Макет*** и ***Формат*** вкладки ***Работа с диаграммами***. Непосредственно в группе Диаграммы можно выбрать тип диаграммы и ее вид. Мастер диаграмм позволяет использовать 11 стандартных типов диаграмм. Для построения графиков функций необходимо использовать ***Точечную*** диаграмму Тип ***График*** используется только для построения линейных диаграмм, так как он не позволяет связать функцию с аргументом. Выбранный вид диаграммы отображается на экране.

После построения графика активизируется вкладка ***Работа с диаграммами.*** Вкладка ***Конструктор*** позволяет изменить тип диаграммы и сохранить его как шаблон, поменять местами строки и столбцы при построении графиков функций и диаграмм.

Вкладка ***Макет*** позволяет управлять оформлением диаграммы. Другие типы позволяют также эффективно компоновать диаграммы и выбрать стиль их оформления (цвет линий, стиль линий, тип диаграммы, шрифт и так далее). Многообразны и доступны возможности оформления диаграмм, например, вставка и оформление легенд, меток данных, оформление осей, возможность вставки линий сеток и т. д.

Работа с матрицами

Операции с матрицами

Электронная таблица позволяет выполнять линейные преобразования матриц: умножение, деление матриц на число, прибавление или вычитание чисел, а также операции над матрицами: сложение, умножение матриц, транспонирование, вычисление обратной матрицы и определителей. Средствами Excel можно решать системы линейных алгебраических уравнений, задачи многомерной оптимизации и др. Для этой цели электронная таблица имеет ряд функций для работы с матрицами, например:

МОБР(массив) – вычисление обратной матрицы;

МОПРЕД(массив) – вычисление определителя матрицы;

МУМНОЖ(массив; массив) – умножение матриц;

ТРАНСП(массив) – транспонирование матриц и др.

Решение систем линейных алгебраических уравнений

С помощью встроенных функций МОБР, МУМНОЖ и МОПРЕД операции решения систем линейных алгебраических уравнений выполняются достаточно эффективно. Например, можно воспользоваться формулой вычисления вектора неизвестных через обратную матрицу A^{-1} и вектор свободных членов B:

$$X = A^{-1} \times B.$$

Пример. Решить систему линейных алгебраических уравнений матричным методом:

$$\begin{cases} 65{,}18x + 36{,}31y + 23{,}76z = 86{,}56; \\ -17{,}98x + 23{,}89y + 27{,}55z = -38{,}07; \\ 23{,}75x + 13{,}95y + 58{,}12z = 53{,}97; \end{cases}$$

Решение:

1. Внесите в ячейки B6–D8 значения коэффициентов при неизвестных.

2. Внесите в ячейки F6–F8 значения свободных членов системы уравнений.

3. Выделите диапазон ячеек B12:D14 и введите формулу МОБР(B6:D8), для завершения операции ввода нажмите комбинацию клавиш Ctrl + Shift + Enter.

4. Выделите диапазон ячеек F12:F14 и введите формулу МУМНОЖ(B12:D14; F6:F8). Для завершения ввода формулы нажмите комбинацию клавиш *Ctrl* + *Shift* + *Enter*. В ячейках F12–F14 появятся значения решения системы уравнений: 1,652508; –0,88768; 0,466381.

Контрольные вопросы

1. Как изменить ширину строки (столбца)?
2. Как вставить строку, группу строк (столбцов)?
3. Каким образом устанавливается формат ячейки?
4. Что такое Строка формул, какие поля она имеет?
5. Где можно увидеть адрес текущей ячейки?
6. Что такое абсолютный, смешанный и относительный адреса ячейки, как они записываются?
7. Как выполняется копирование ячеек?
8. Как ввести в ячейку текст, формулу?
9. Как сохранить таблицу на диске?
10. Как загрузить таблицу с диска?
11. Как вывести таблицу на печать?
12. Каким образом осуществляется фиксация шапки таблицы?
13. Назовите основные элементы диаграммы.
14. Опишите порядок построения графиков функций.
15. Как добавить график функции на диаграмму?
16. Как изменить заголовок диаграммы или ее осей?
17. Как изменить стиль линий сетки?
18. Каким образом перемещается диаграмма по рабочему листу?
19. Как изменить размеры диаграммы?
20. Как установить вспомогательную ось на диаграмму?
21. Для чего необходима проверка данных?
22. Как создать список для выбора значений из списка?
23. Как вычисляется значение обратной матрицы для заданной матрицы?

2.4. Сервисные инструментальные программные средства

Сервисная программная система – это программный продукт, изменяющий и дополняющий пользовательский и программный

интерфейсы операционной системы. Сервисные системы используются для обеспечения эффективного взаимодействия пользователя, компьютера и сетевых систем. Их характеризует то, что, с одной стороны, они не входят в состав операционных систем, а с другой – предназначены для реализации служебных функций по управлению компьютером. Иными словами, сервисные программы – это вспомогательные инструменты, расширяющие и дополняющие функциональность операционных систем. Среди множества сервисных систем особое место занимают файловые менеджеры, например, программа Проводник Windows (англ. *Windows Explorer*), встроенная в операционную систему Windows.

Файловый менеджер

Файловый менеджер (англ. *file*) – компьютерная программа, предоставляющая интерфейс пользователя для работы с файловой системой и файлами. Файловый менеджер позволяет выполнять наиболее частые операции над файлами: создание, открытие/проигрывание /просмотр, редактирование, перемещение, переименование, копирование, удаление, изменение атрибутов и свойств, поиск файлов и назначение прав. Помимо основных функций, многие файловые менеджеры включают ряд дополнительных возможностей, например таких, как работа с сетью (через FTP, NFS и т. п.), резервное копирование, управление принтерами и др. Применяются различные типы файловых менеджеров.

Электронные словари и переводчики

Электронный словарь и *электронный переводчик* – это устройства, предназначенные для перевода отдельных слов (основная масса переводит только по одному слову) или фраз с иностранного языка на русский или в обратную сторону. Некоторые модели имеют возможность озвучить переведенное слово или фразу. Принцип работы и словаря, и переводчика практически один и тот же. Это устройства, которые позволяют получить для любого иностранного слова множество вариантов перевода, и ориентированы на различные предметные области изучаемого иностранного языка. Каждый переводчик или словарь имеет свой словарный запас, чем он больше, тем больше у вас шансов найти нужное слово.

Электронный словарь – узконаправленное устройство с определенным спектром выполняемых функций и предлагаемых опций с вариантами переводов иностранный слов или фраз. Такой словарь, зачастую, только позволяет быстро найти нужное слово с учетом морфологии и возможностью поиска словосочетаний (примеров употребления). Наиболее популярные виды словарей: Free On-line Dictionary of Computing, FreeDict, Jargon file, WordNet. Часто используемые программы: AtomicDic, GoldenDict, StarDict, программы, сайты и др.: ABBYY Lingvo, DICT – сетевой протокол, Мультитран, Polyglossum, МультиЛекс (русско-английские, немецкие, французские, испанские, итальянские, португальские и многоязычные словари, включающие толковые словари и тематические словарные базы для перевода специализированной лексики).

Электронный переводчик в отличие от словаря, как правило, имеет разговорник, разбитый по темам и включающий в себя наиболее часто употребляемые фразы и выражения, которые можно также прослушать при наличии функции произношения. Во многих переводчиках есть дополнительные технические возможности, помогающие в изучении языка: транскрипция, обучающие программы в виде электронных учебников по грамматике, функция орфографического корректора, обучающие игры, экзамены TOEFL и др. Также имеются дополнительные функции, не относящиеся к функциям перевода слов: мировое время, перевод мер и весов, соответствие размеров одежды и обуви, будильник, калькулятор, метрические преобразования, записная книжка, которые будут очень кстати, если отправляться в путешествие или рабочую командировку по разным странам.

Программа ABBYY Lingvo. Российская компания ABBYY – ведущий мировой разработчик программного обеспечения и поставщик услуг в области распознавания документов, лингвистики и перевода. Запустите словарь *ABBYY Lingvo*, выберите нужное направление перевода (англо-русское или русско-английское) и введите незнакомое слово в строку ввода. Нажмите кнопку «Перевод» или клавишу Enter. В карточке словаря вы можете посмотреть переводы слова, примеры употребления, ознакомиться с транскрипцией и послушать произношение слова.

Система перевода PROMT Professional. Программа ***PROMT Professional*** – профессиональная система перевода текстов

с английского, французского, немецкого, испанского, итальянского языков на русский язык и обратно, с расширенным набором настроек для точного перевода документов по различным тематикам (деловых контрактов, технических описаний, финансовых и других документов).

Основные возможности переводчика *PROMT Professional*:

Пользовательские словари и онлайновые словарные базы дополняют основной словарный запас программы.

Перевод файлов *.pdf, *.doc, *.xls, *.ppt, *.msg, *.html, *.xml, Open Office.org Writer.

Тексты *pdf*-файлов можно переводить непосредственно в редакторе PROMT с сохранением исходного форматирования, а также в интерфейсе программ *Adobe Acrobat* и *Adobe Professional*.

Использование базы переведенных текстов позволяет экономить время при работе с типовыми документами и поддерживать единую корпоративную терминологию.

Перевод без ошибок и опечаток.

Возможность переводить несколько файлов одновременно.

Удобный, простой и понятный интерфейс. Удобные, полные и понятные справочные материалы

Личный кабинет для доступа к пользовательским ресурсам и сервисам системы *PROMT*.

Программы архивации данных

Для уменьшения места, необходимого для содержания копий, и удобства эксплуатации создано достаточно много программ, позволяющих работать с архивными файлами. Архивный файл может содержать один и более файлов в сжатом виде. Можно как извлекать из этого файла исходные файлы в их первоначальном виде, так и добавлять в него новые. Среди популярных программ архивации, работающих в среде Windows, можно назвать такие, как WinZip, WinRar, WinArj, NetZip и др.

Архиватор WinZip. После установки этой программы создается не только меню *WinZip*, но и ярлыки на рабочем столе и в кнопке Пуск, в контекстное меню для папок и файлов добавляются пункты, связанные с архивацией. Программа может быть запущена последовательностью *Пуск, Программы, WinZip* или с рабочего стола.

Создание и изменение архива. Работа с архивом начинается с создания файла архива при помощи последовательности команд

File, New Archive (новый архив). Как в любых программах *Windows*, любая команда или операция может быть осуществлена разными способами, в частности, клавишами Ctrl+N или кнопкой *New* на панели инструментов. Файл архива будет создан после задания его имени. При установке флажка *Add dialog* будет открыто дополнительное диалоговое окно *Add*, в котором можно указать имена и параметры файлов, добавляемых в архив. В противном случае (если флажок *Add dialog* не установлен) будет создан пустой архивный файл.

При добавлении файлов в уже существующий архив необходимо открыть выбранный файл архива. Для этого используется команда открытия *File, Open Archive*. Для включения файла в архив используется команда добавления *Actions, Add*. В окне *Add* кроме указания файла (файлов) для добавления в архив задаются параметры включения в архив: например, активизация параметра *Update* (*and add*) *files* – обновляются файлы, которые уже есть в архиве и добавляются новые. При добавлении файлов можно указать пароль для недопущения несанкционированного доступа к файлам архива. Команда добавления запускается нажатием кнопки Add.

Извлечение файлов из архива. При извлечении файла из архива извлекаемый файл из архива не удаляется. Для извлечения файлов из архива необходимо открыть файл архива и запустить команду извлечения. Команда извлечения файла из архива может быть вызвана последовательностью команд *Actions, Extract*. В появившемся окне необходимо в поле *Extract to*: ввести имя папки, в которую будут помещаться извлекаемые из архива файлы, а в поле *Files* – имена извлекаемых файлов.

Программы распознавания текстов

Наиболее распространенное программное средство оптического распознавания текста – ***ABBYY FineReader***, которая поддерживает такие функции, как автоматическая сетевая установка, работа с сетевыми сканерами, удобные инструменты администрирования, распределенная обработка документов, поддержка многопроцессорных систем, распознавание изображений, полученных с помощью цифровой камеры. Если под рукой нет сканера, можно сделать снимок документа с помощью фотоаппарата. Программа адаптируется к этим изображениям, учитывая такие их особенности, как неравномерная освещенность, нечеткая фокусировка и искажение строк.

Программа управления данными

Программа ***SQL Server*** представляет безопасную, надежную и высокопроизводительную платформу управления данными для корпоративных пользователей. Новая версия СУБД обеспечивает комплексное управление данными и предоставляет платформу для анализа, что помогает организациям уверенно управлять критически важной информацией и использовать мощные бизнес-приложения. С помощью встроенных в SQL Server функций отчетности и инструментов анализа данных сотрудники компаний, обрабатывая бизнес-информацию, смогут получать более полные сведения, принимать правильные решения и быстрее достигать результатов. Различные выпуски новой версии обладают широкими функциональными возможностями от высокой доступности и надежности при масштабировании до усовершенствованных средств анализа бизнес-процессов.

Контрольные вопросы

1. Перечислите основные особенности сервисных инструментальных программных средств.
2. Какие файловые менеджеры используются в Вашей практике?
3. Какие средства представляются файловыми менеджерами для работы с файловой системой и файлами.
4. Какая первая операция должна быть выполнена при добавлении файлов в уже существующий архив?

2.5. Системы математических вычислений *MatLab*

Графический интерфейс пользователя и простейшие вычисления

MatLab (англ. *Matrix Laboratory* – матричная лаборатория), разработанная фирмой *MathWorks, Inc*. (США), является интерактивной системой для выполнения инженерных, экономических, научных и других расчетов. В MatLab интегрирован мощный математический аппарат, позволяющий решать сложные задачи анализа динамических систем, а также систем управления с предоставлением пользователю эффективных средств графического отображения информации.

Графический интерфейс пользователя MatLab состоит из 4 независимых окон, имеющих следующие названия:

– *Command Window* (Командное окно) – область ввода команд и вывода получаемых результатов;

– *Workspace* (Рабочая область) – область отображения всех переменных, используемых в текущем сеансе работы;

– *Command History* (Окно истории команд) – перечень команд, выполнявшихся ранее;

– *Current Directory* (Список файлов) – один из элементов панели инструментов (в верхней части экрана), на который указывается папка, которая будет по умолчанию использоваться для всех операций с файлами.

Вид окна можно изменять с помощью команд из меню ***Desktop***, а также используя стандартные возможности управления окнами.

В рабочей области окна ***Command Window*** находится строка ввода команд, отмеченная знаком курсора >>, в которой можно вводить числа, имена переменных и знаки операций, составляющих в совокупности выражения. Имена переменных должны начинаться с буквы и состоять из букв, цифр и знаков припинания.

Вид команд, вводимых в командном окне, очень нагляден. Например, если ввести $x = 4 \times 3$ и нажать клавишу Enter, то на экран выводится результат. После отображения результатов вычисления в командном окне создается новая строка ввода команд, отмеченная знаком >>. При этом выполненная команда отображается в окне истории команд (***Command History)***, а в рабочей области (***Workspace***) создается переменная *x*.

Если команда завершается точкой с запятой, то ее результат не выводится на экран. Если при выполнении команды не указана переменная, которой должен быть присвоен результат, то он по умолчанию присваивается переменной **ans**.

Новую команду в командном окне можно вводить только в строке ввода, в которой находится курсор. В эту строку можно копировать команды, выполненные ранее, используя обычные способы копирования. Кроме того, команды, выполненные ранее, можно вызывать в строку ввода из окна истории команд (двойным щелчком или перетаскиванием с помощью мыши), а также нажатиями клавиши «Стрелка вверх».

С переменными можно работать в рабочей области. Двойной щелчок по имени переменной в рабочей области вызывает редактор в виде электронной таблицы, в котором можно изменять значение переменной, строить графики (если переменная представляет собой матрицу) и выполнять ряд других операций. Кроме того, для переменных в рабочей области можно использовать команды контекстного меню, вызываемого правой кнопкой мыши.

Заглавные и строчные буквы в именах переменных различаются. Имеется возможность просмотра краткой подсказки по любой функции. Для этого используется команда **help**. Например, для получения подсказки о функции sin следует в окне команд ввести **help sin**. Для очистки командного окна используется команда **clc**.

Представление и отображение данных в MatLab

Основной тип данных, используемый в *MatLab* – вещественные числа (тип *double*). Точность представления – 15 значащих цифр. Для управления форматом представления данных на экране используется команда **format**. Основные форматы вывода данных на экран следующие: **format short** – вывод с точностью до четырех значащих цифр после запятой (используется по умолчанию); **format long** – полное представление числа. Выбранный формат применяется до тех пор, пока не будет введена команда **format** с другим форматом.

Значения переменных, вычисленных в течение текущего сеанса работы, сохраняются в специально зарезервированной области оперативной памяти компьютера, называемой рабочей областью. Значения переменных, необходимость хранения которых нецелесообразна, удаляются командой **clear name1 name2 …**. Здесь **name1, name2, … –** имена удаляемых переменных.

Для удаления всех переменных используется команда **clear.**

Содержимое рабочей области (т. е. значения всех переменных) можно сохранить на диске, используя команду меню ***File, Save Workspace As...,*** или команду ***save*** в командном окне. Файл с содержимым рабочей области сохраняется с расширением *.mat. При выходе из MatLab содержимое рабочей области теряется, поэтому, прежде чем завершать сеанс работы в *MatLab*, необходимо сохранить рабочую область.

Для загрузки сохраненного содержимого рабочей области используется команда меню ***File, Open*** или команда ***load*** в командном окне. Имя файла в командах **save** и **load** задается по обычным правилам. Если при сохранении или загрузке файла не указывается путь, то используется папка, указанная в поле окна ***Current Directory***.

Все команды, вводимые в *MatLab*, а также результаты их выполнения, выводимые в командном окне, можно сохранять в текстовом файле. Для этого требуется ввести команду ***diary***. Для прекращения вывода команд и результатов в файл используется команда ***diary off***.

Операции с матрицами

Все данные в *MatLab* рассматриваются как матрицы. Даже обычная переменная представляет собой матрицу размером 1×1.

Применяются следующие простейшие способы задания матриц-перечислением:

- в виде диапазона значений (с помощью операции «двоеточие»);
- с помощью специальных функций.

Например, требуется задать для дальнейшего применения матрицы:

$$a = (3\ 6\ 5),\ b = \begin{pmatrix} 5 & 7 & 9 \\ 6 & 2 & 1 \end{pmatrix}.$$

Для этого ввести:

a = [3 6 5];
b = [5 7 9; 6 2 1].

Как видно, матрицы вводятся в квадратных скобках. Если матрица двумерная (многомерная), то она вводится по строкам; при вводе в одной строке конец строки обозначается точкой с запятой. Элементы матрицы разделяются пробелами. Вместо пробела в качестве разделителя элементов матриц можно использовать запятую, например:

a = [3,6,5];
b = [5,7,9; 6,2,1].

Вычисления с матрицами

Основные приемы работы с матрицами рассмотрим на следующих примерах.

1. Задать матрицу-строку:

```
>> s1=[1 3 2]
s1 =
1 3 2
```

2. Задать матрицу-столбец:

```
>> s2=[2;1;–1]
s2 =
2
1
–1
```

3. Задать одномерную матрицу-строку, содержащую числа от 0 до 10 с шагом 0,1.

В конце команды вставить точку с запятой, чтобы созданная матрица (в рассматриваемом примере из 101 элемента) не выводилась на экран.

```
>> dialp=0:0.1:10;
```

4. Вычислить скалярное произведение векторов

```
>> a=[1 2 3];
>> b=[3 2 1];
>> a* b'
ans = 10
```

В соответствии с правилами умножения матриц, принятыми в линейной алгебре, можно умножать вектор-строку на вектор-столбец, поэтому, для вычисления скалярного произведения необходимо, как видно, предварительно транспонировать вектор *b*: «'» – символ транспонирования.

5. Поэлементное умножение векторов

```
>> a=[1 2 3];
>> b=[3 2 1];
>> a.*b
ans =
3 4 3
```

6. Создать матрицу

```
>> A=[–1 1 2;3 –1 1; –1 3 4]
```



```
A =
 –1   1   2
  3  –1   1
 –1   3   4
```

7. Выделить заданный столбец матрицы

```
>>A(:,1)
ans =
–1
 3
–1
```

8. Выделить заданную строку матрицы

```
>>A(2, :)
ans =
3  –1  1
>>
```

9. Выделить определитель матрицы

```
>> det(A)
ans =
```

10. Вычислить обратную матрицу

```
>>  inv(A)
ans =
```

–0.7000	0.2000	0.3000
–1.3000	–0.200	0.7000
0.80000	0.20000	–0.200

Пример. Решить систему линейных алгебраических уравнений:

$$A \times X = B.$$

Для решения системы уравнений следует вычислить столбец значений переменных **X** следующим образом: $X = A^{-1} B$, где A – матрица коэффициентов уравнений (A^{-1} – обратная матрица), B – столбец правых частей уравнений. Для проверки полученного решения необходимо перемножить матрицы A и X: в результате получиться вектор правых частей уравнений B**.**

Построение графиков

Для построения простых графиков вида $y = f(x)$ применяется функция **plot(x,y,'строка')**, где ***x*** и ***y*** – матрицы (обычно одномер-

ные), задающие координаты точек, по которым строится график; 'строка' – набор управляющих символов, задающих вид линии графика (необязателен). Кроме того, в окне графика имеется собственная система меню для настройки его внешнего вида. Можно также построить в одном окне графики несколько графиков.

Пример. Построить график функции

$y = 0{,}25 + \sin(x) - 1$

для значений аргумента от 0 до 10.

Решение.

Получить массив значений переменной 0 от 0 до 10 с шагом 0,1. Для этого ввести: ***x* = 0:0.1:10;** (точка с запятой требуется, чтобы на экран не выводились все полученные величины).

Получить массив соответствующих значений переменной y:

$y = 0{,}25 + \sin(x) - 1$.

Для построения графика ввести: **plot(*x,y*).**

Построение трехмерных графиков

В качестве примера построения трехмерных графиков рассмотрим построение графиков функций ***z* = *f*(*x,y*)**. Такие графики строятся следующим образом:

- задаются диапазоны значений переменных *x* и *y*;
- строятся две матрицы значений переменных *x* и *y*, составляющие координатную сетку для последующего вычисления функции $z = f(x,y)$ и построения ее графика. Для этого используется функция meshgrid: [*x,y*] = meshgrid(диапазон_*x*, диапазон_*y*); (точка с запятой в конце строки желательна, так как матрицы координатной сетки *x* и *y*, получаемые в результате применения функции meshgrid, обычно достаточно велики);
- вычисляются значения функции $z = f(x,y)$ (матрица *z*);
- строится график функции $z = f(x,y)$: plot3(*x,y,z*).

Основы программирования в MatLab

Система компьютерной математики *MatLab* имеет собственный язык программирования высокого уровня, включающий как стандартные возможности традиционных языков программирования, так и собственно математические возможности MatLab.

Файлы с текстами программ сохраняются в MatLab с расширением *.m и, как правило, называются М-файлами. Для подготовки текста нового М-файла следует использовать команду File, New, M-file, для загрузки существующего М-файла и его последующего редактирования – команду File, Open. В MatLab имеются два основных вида М-файлов: файлы-сценарии и файлы-функции.

Файлы-сценарии

Файл-сценарий представляет собой набор команд MatLab, сохраненный в файле. После того, как файл-сценарий подготовлен и сохранен, для его выполнения требуется указать имя данного файла-сценария в командном окне или в другом М-файле. Файл-сценарий не имеет входных или выходных параметров. При выполнении файла-сценария используются и изменяются переменные рабочей области, как если бы команды, составляющие файл-сценарий, просто вводились в командном окне.

Пример. Разработать файл-сценарий для выделения последнего столбца произвольной матрицы в отдельную матрицу.

Для подготовки текста М-файла требуется выбрать команду **File, New M-file.** Вызывается редактор для записи текста М-файла. В данном примере текст М-файла может быть следующим:

```
% Выделение последней строки в отдельную матрицу
[m,n] = size(a);
b = a(:,n).
```

Символ **%** в М-файле является признаком комментария. Функция **size(имя_матрицы)** определяет размеры заданной матрицы. Точнее, результатом выполнения функции **size** является матрица (строка) из двух элементов, первый из которых – количество строк заданной матрицы, второй количество столбцов. В результате переменная ***m*** получит значение, равное количеству строк матрицы ***a***, а переменная ***n*** – количеству ее столбцов. Последняя команда в файле-сценарии выделяет из матрицы с именем ***a*** последний столбец.

Для сохранения М-файла выбирается команда **File, Save**, в которой указывается имя. После сохранения М-файла можно закрыть окно редактора и вернуться в командное окно.

Прежде чем использовать созданную программу-сценарий, необходимо создать матрицу с именем ***a***, которая будет обрабатываться с помощью этого сценария. Например, введем в командном окне следующую матрицу из трех строк и четырех столбцов:

a = [5,8,5,7; 3,1,9,5; 5,9,6,1]. Матрица, которую предполагается обрабатывать с помощью созданного файла-сценария, должна иметь имя **a**, так как это имя указано в файле-сценарии. Чтобы выделить из введенной матрицы последний столбец в отдельную матрицу, требуется ввести в командном окне имя файла-сценария, т. е. слово **stolbec**. В результате выполнения файла-сценария в рабочей области создаются три новые переменные: *m* = 3, *n* = 5, *b* = [7; 5; 1]. Если до выполнения файла-сценария переменные с такими именами уже имелись в рабочей области, то их прежние значения теряются.

Файлы-функции

Файл-функция представляет собой программу, обычно имеющую входные и выходные параметры. Файл-функция обрабатывает величины, переданные ему в качестве входных параметров, и возвращает переменные, указанные как выходные параметры.

При выполнении файла-функции переменные рабочей области не изменяются и не используются: все переменные файла-функции локальны. Например, если в рабочей области имеется переменная с именем ***x***, и в файле-функции имеется переменная с тем же именем, то любые операции с переменной ***x*** в файле-функции никак не влияют на ее значение в рабочей области (конечно, если при вызове файла-функции переменная x не была указана в качестве выходного параметра).

Основные управляющие структуры для программ в MatLab

Основные конструкции, используемые для управления выполнением программы в MatLab, подобны аналогичным конструкциям в алгоритмических языках программирования Fortran, Pascal и др.

Условный оператор:

```
if условие_1
  команды_1
elseif условие_2
  команды_2
else
  команды_3
end
```

Здесь команды_1, команды_2 и команды_3 – произвольные наборы команд MatLab, выполняемые при соответствующих условиях.

Оператор цикла "до":

```
for переменная=начальное_значение : шаг : конечное_значение
  команды
end
```


Если шаг равен единице, то его можно не указывать.

Оператор цикла "пока":

```
while условие
  команды
end
```

Для прерывания цикла используется команда break.

Переключатель:

```
switch выражение
case значение_1
  команды_1
case значение_2
  команды_2
otherwise
  команды
end
```

Если выражение равно значению_1, то выполняются команды_1; если выражение равно значению_2, то выполняются команды_2 и т. д. Если выражение не равно ни одному из указанных значений, то выполняются команды, указанные после слова otherwise.

Команда ввода (ввод значения переменной *x*):

```
x=input('Введите переменную:');
```

Команда вывода (вывод значения переменной *y*):

```
disp('Значение Y равно ') ; disp(y)
```

Решение алгебраических и дифференциальных уравнений

Функция для решения алгебраических уравнений

Основная функция для решения алгебраических уравнений вида f(*x*)=0 функция **zero**. Она может применяться в двух формах:

fzero('уравнение', начальная_точка)

или

fzero('уравнение', [*a, b*]),

где 'уравнение' – левая часть решаемого уравнения *f*(*x*)=0 или имя М-файла, реализующего функцию *f*(*x*); начальная_точка – значение переменной, в окрестности которого ищется решение; *a, b* – границы

отрезка, на котором ищется решение, при этом величины $f(a)$и $f(b)$ должны иметь разные знаки. Если уравнение имеет несколько решений, то функция fzero находит лишь одно из них. Другие решения требуется определять, изменяя значения начальной точки или границы отрезка (a или b).

Решение систем алгебраических уравнений

Основная функция для решения систем алгебраических уравнений – функция:

fsolve('система_уравнений', начальная_точка),

где 'система уравнений' – имя М-файла (функции), реализующего левые части уравнений системы; начальная_точка – массив, задающий начальную точку для поиска решения. Для использования функции **fsolve** решаемую систему уравнений необходимо преобразовать к виду, где правые части уравнений представляют собой нули.

Решение дифференциальных уравнений

Для решения дифференциальных уравнений в MATLAB имеется ряд специальных функций, называемых решателями. Решатель **ode55** вызывается следующим образом:

[*x*,*y*] = ode55('ду',[x_{min}, x_{max}], y_0),

где 'ду' – имя М-файла (функции), реализующего правую часть системы дифференциальных уравнений; *x*min**,** *x*max – границы диапазона значений независимой переменной; y_0 – массив начальных условий для функций $y_1, y_2, \ldots, y_n$.

Выходные параметры решателя **ode55** имеют следующий смысл:

x – массив-столбец значений независимой переменной;

y – матрица из n столбцов, представляющих собой наборы значений переменных $y_1, y_2, \ldots, y_n$, полученные по результатам решения системы дифференциальных уравнений. Каждый столбец содержит набор значений одной переменной.

Поиск экстремумов функций.
Решение задач линейного и нелинейного программирования

Поиск экстремумов функций

Для поиска минимумов функций одной переменной $y = f(x)$ используются функция MATLAB:

fminbnd('функция', *a, b*),

где 'функция ' – функция $f(x)$, для которой требуется найти минимум, или имя М-файла, реализующего эту функцию;

a, b – границы отрезка, на котором ищется минимум.

Если функция $y = f(x)$ имеет несколько минимумов, то функция **fminbnd** находят лишь один из них. Другие минимумы требуется определять, изменяя значения *a* или *b*. Если требуется найти максимум функции $y = f(x)$, то необходимо использовать функции *MatLab* **fminbnd**, указав в них функцию $f(x)$, умноженную на –1.

Для поиска минимума функций нескольких переменных применяется функция **fminsearch:**

fminsearch('функция',x_0),

где 'функция' – функция $f(x)$, для которой требуется найти минимум, или имя М-файла, реализующего эту функцию; x_0 – вектор аргументов, с которого начинается поиск экстремума.

Решение задач линейного программирования

Для решения задач линейного программирования в MATLAB используется функция:

linprog(f, a, b, ar, br, x_{min}, x_{max}),

где *f* – массив-столбец коэффициентов целевой функции, подлежащей минимизации;

a – матрица коэффициентов ограничений-неравенств, имеющих вид «меньше или равно» (коэффициенты каждого ограничения указываются в отдельной строке матрицы);

b – массив-столбец правых частей ограничений-неравенств;

ar – матрица коэффициентов ограничений-равенств (коэффициенты каждого ограничения указываются в отдельной строке матрицы);

br – массив-столбец правых частей ограничений-равенств;

x_{min} – массив-столбец ограничений на минимальные значения переменных;

x_{max} – массив-столбец ограничений на максимальные значения переменных.

Большой класс практических задач оптимизации может быть сведен к решению задачи линейного программирования.

Пример. Молочный завод выпускает два вида молочных продуктов 1 и 2, для чего используются два исходных вида сырья – A и B. Суточные запасы этих видов сырья составляют соответственно 6 и 8 т. Расходы сырья A и B на 1 т соответствующих продуктов приведены в табл. 2.1.

Таблица 2.1

Исходный вид сырья	Расход исходных видов сырья на 1 т продуктов, т		Запас сырья, т
	Продукт 1	Продукт 2	
A	1	2	6
B	2	1	8

Изучение рынка сбыта показало, что суточный спрос на продукт 2 не превышает спроса на продукт 1 более чем на 1 т и, кроме того, не превышает 2 т. Прибыль от продукта 1 составляет 3000 ден.ед./т, от продукта 2 – 2000 ден.ед./т. Требуется определить, какое количество продуктов каждого вида следует производить, чтобы получить максимальную прибыль.

Построение математической модели задачи

Обозначим через x_1 и x_2 (т) суточный обьем производства продуктов 1 и 2 соответственно.

Целевой функцией S является прибыль:

$$S = (3x_1 + 2x_2)\,1000 \rightarrow \max. \qquad (2.1)$$

Запишем условие задачи в виде системы ограничений:

1) ограничение по сырью A:

$$x_1 + 2x_2 \le 6, \qquad (2.2)$$

2) ограничение по сырью B:

$$2x_1 + x_2 \le 8, \qquad (2.3)$$

3) ограничения по спросу:

$$x_2 - x_1 \le 1, \qquad (2.4)$$

$$x_2 \le 2. \qquad (2.5)$$

Очевидно, должно также выполняться условие неотрицательности параметров:

$$x_1 \ge 0\ ; \quad x_2 \ge 0. \qquad (2.6)$$

Как видно, рассматриваемая математическая модель записывается в форме общей задачи линейного программирования: найти значения оптимизируемых параметров x_1 и x_2, обеспечивающие максимум целевой функции 2.1 при ограничениях 2.2–2.5, а также при ограничениях на не отрицательность параметров 2.6.

Решение задачи в системе MatLab

Для решения задач линейного программирования применим процедуру **linprog** прикладного пакета **Optimization Toolbox** системы MatLab. Процедура имеет много вариантов обращения и может использовать различные алгоритмы.

Процедура решает задачу:

$$\min_x S(x)\ ,$$

при ограничениях:

$$A \cdot x \le b;$$

$$Aeq \cdot x = \mathrm{beq};$$

$$lb \le x \le ub,$$

где S, x, b, beq, lb и ub – векторы, а A и Aeq – матрицы.

Наиболее простая форма обращения:

$$x = \mathrm{linprog}(S,A,b,Aeq,beq),$$

а наиболее полная:

$$[x,Sval,exitflag,output,lambda] = \\ = \mathrm{linprog}(S,A,b,Aeq,beq,lb,ub,x0,options).$$

Для рассматриваемой задачи достаточно:

$$[x,Sval] = \text{linprog}(S,A,b,Aeq,beq,lb).$$

Здесь *Sval* – значение целевой функции.

Если необходимо найти максимум целевой функции, то ее коэффициенты следует взять с обратным знаком. Если в ограничениях неравенств стоят знаки ≥, то знаки левых и правых частей неравенств следует взять с обратным знаком.

Рассмотренный пример может быть решен с помощью следующего сценария:

```
function optim
% ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ
% ЦЕЛЕВАЯ ФУНКЦИЯ
%  S = (3*X(1) + 2*X(2))*1000 →max
% ОГРАНИЧЕНИЯ
%  X(1) + 2*X(2) <= 6
%  2*X(1) + X(2) <= 8
%  X(2) – X(1) <= 1
%  X(2) <= 2
%  X(1) >= 0; X(2) >= 0
%
S = [–3; –2];
A =  [1 2
      2 1
     –1 1
      0 1];
b = [6; 8; 1; 2];
lb = zeros(2,1);  % ФОРМИРОВАНИЕ ВЕКТОРА [0; 0]
[x,Sval] = linprog(S,A,b,[],[],lb)
Результат будет в виде:
Optimization terminated successfully.
x =
    3.3333
    1.3333
Sval =
  –12.6667
```


Таким образом, первый продукт должен производиться в объеме 3,3333 т, а второй – 1,3333 т. При этом прибыль составит 12,6667 тыс. ден. ед. и будет максимальной.

Решение задач нелинейного программирования

Для решения задач нелинейного программирования в MatLab используется функция:

fmincon('ц_ф',x0, a, b, ar, br,xmin, xmax, 'н_о'),

где 'ц_ф' – целевая функция или имя М-файла, реализующего эту функцию;

x_0 – массив-строка, задающий начальные значения переменных;

a – матрица коэффициентов линейных ограничений-неравенств (коэффициенты каждого ограничения указываются в отдельной строке матрицы);

b – массив-столбец правых частей линейных ограничений неравенств;

ar – матрица коэффициентов линейных ограничений-равенств (коэффициенты каждого ограничения указываются в отдельной строке матрицы);

br – массив-столбец правых частей линейных ограничений-равенств;

x_{min} – массив-столбец ограничений на минимальные значения переменных;

x_{max} – массив-столбец ограничений на максимальные значения переменных;

′н_о′ – имя М-файла (функции), реализующего нелинейные ограничения задачи.

Моделирование динамических систем в пакете Simulink

Основные сведения о системе Simulink

Система *Simulink* предназначена для моделирования динамических систем, состояние которых изменяется во времени. Система Simulink может применяться для моделирования самых разнообразных объектов и процессов: электрических схем, систем передачи и обработки сигналов, механизмов, тепловых процессов и т. д. Модель в системе Simulink строится в виде набора

стандартных блоков, описывающих моделируемый объект или явление. На основе такой модели система Simulink автоматически строит описание объекта моделирования в виде систем дифференциальных уравнений, решает эти системы и отображает характеристики объекта моделирования.

Для начала работы с системой Simulink требуется в командном окне ввести команду **simulink**. На экран выводится окно библиотек Simulink (**Simulink Library Browser**).

В состав системы Simulink входят основная библиотека блоков (собственно Simulink) и ряд специализированных библиотек. Для удобства пользования библиотеки Simulink разбиты на группы и подгруппы блоков.

Приведем примеры некоторых из них:

1. **Sources** (источники) – набор блоков, используемых для имитации источников моделируемых величин, например: **Clock** (сигнал, имитирующий независимую переменную), **Constant** (постоянный сигнал), **Pulse Generator** (импульсный сигнал), **Random Number** (случайный сигнал), **Sine Wave** (синусоидальный сигнал), **Step** (ступенчатый сигнал), **From File** (ввод величины из файла), From **Workspace** (ввод величины из рабочей области MATLAB) и т. д.;

2. **Sinks** (приемники) – набор блоков, используемых для имитации приема и отображения моделируемых величин, например: **Display** (отображение числовой величины), **Scope** (осциллограф), **To File** (вывод результатов моделирования в файл), **To Workspace** (вывод результатов моделирования в рабочую область MATLAB) и т. д.;

3. **Continuous** (непрерывные процессы): **Derivative** (производная), **Integrator** (интегрирование), **Delay** (задержка) и т. д.:

4. **Math Operations** (математические операции): **Add** (суммирование и вычитание входных величин), **Divide** (деление и перемножение входных величин), **Gain** (умножение на число или матрицу), **Sum** (то же, что **Add**), **Product** (то же, что **Divide**), **Math Function** (набор математических функций) и т. д.

Имеется также большой набор специализированных библиотек, например, **Communications Blockset** (коммуникационные системы), **Neural Network Toolbox** (нейронные сети), **SimMechanics** (механизмы), **SimPowerSystems** (электротехника и электроника) и т. д.

Для создания модели требуется в окне библиотек Simulink выбрать команду **File, New, Model**. Создается пустое окно модели.

Необходимые блоки перетаскиваются из библиотек в окно модели с помощью мыши.

Двойной щелчок мыши по любому из блоков, размещенных в окне модели, вызывает на экран окно параметров выбранного блока. Эти параметры могут представлять собой, например, амплитуду и частоту моделируемого сигнала, сопротивление резистора, формулу математического преобразования и т. д.

Результаты моделирования не только отображаются на экране с помощью соответствующих блоков, но и представляются в виде матриц, которые могут выводиться в рабочую область MATLAB. Это позволяет выполнять их дальнейшую обработку, используя все средства MATLAB. Сохранение файла модели осуществляется командой **File, Save**. Файл сохраняется под указанным именем с расширением MDL.

Контрольные вопросы

1. Каково назначение независимых окон графического интерфейса пользователя MatLab?
2. Выводится ли результат на экран, если команда завершается точкой с запятой?
3. Можно ли в качестве разделителя элементов матриц использовать запятую?
4. Как построить в одном окне графики нескольких функций?
5. Имеет ли файл-сценарий входные или выходные параметры?
6. В чем особенность поэлементного умножения векторов?
7. Для решения дифференциальных уравнений в MatLab имеется ряд специальных функций, называемых решателями. Как называется решатель **ode55**?
8. Как выполняется обращение к решателю дифференциальных уравнений в MatLab **ode55?**
9. Какая функция используется в MatLab для решения задач нелинейного программирования?
10. Для моделирования каких систем эффективно применение пакета Simulink?
11. Какой набор блоков используется для имитации приема и тображения моделируемых величин?

2.6. Система подготовки презентаций

Презентация – это упорядоченная последовательность слайдов и слайд-фильмов, раздаточные материалы, а также конспект и планы докладов, хранящиеся в одном файле. *Слайд* – отдельная страница презентации, включающая разные объекты презентаций: заголовок, текст, графику, диаграммы, таблицы, рисунки, рисованные объекты, фотографии, формулы, видеоклипы, видеофильмы. Слайды можно распечатывать на бумаге или на прозрачной пленке.

Раздаточный материал – это распечатанные в компактном виде несколько слайдов на одной странице с целью закрепления восприятия слушателями темы доклада и возможности самостоятельно вернуться к теме доклада.

Конспект доклада – текст доклада, при печати которого на каждой странице будут выведены уменьшенное изображение слайда и текст, составляющий его содержание.

Как правило, для создания презентаций используются два программных продукта: Microsoft PowerPoint или Open Office Impress. Файлы Power Point имеют расширение .pptx.

Общие принципы создания презентаций

Существуют некоторые общие принципы, которыми рекомендуется руководствоваться при создании презентаций.

На восприятие одного слайда необходимо от 1 до 5 минут.

На слайде может быть от 20 до 60 слов.

Каждый слайд должен иметь свой заголовок, т. е. один слайд – одна мысль.

Информация должна быть структурирована – списки, таблицы и рисунки работают лучше, чем текстовые блоки.

Презентация должна быть выдержана в едином строгом стиле.

Классификация презентаций

Презентации можно классифицировать по некоторым признакам, определяющим требования к их оформлению и представлению: *по способу представления, по способу управления представлением, по области применения.*

По способу представления информации презентации делятся на инейные и со сценарием. В линейных презентациях материал расположен по порядку: начало, продолжение, завершение. Такой вид

презентации используется в обучающих презентациях, рекламных роликах и др. Презентации со сценарием предполагают показ слайдов, снабженных анимированными объектами, видеоматериалом, звуковым сопровождением, а также спецэффектами.

По способу управления представлением презентации можно разделить на интерактивные и непрерывные. Интерактивные презентации выполняются под управлением пользователя. Непрерывные презентации позволяют предоставлять информацию в автоматическом режиме.

По области применения презентации можно разделить на следующие виды.

Маркетинговые – отражают направления деятельности компании, виды услуг.

Торговые – используются дилерами или торговыми агентами при заключении сделок, продаже товаров и услуг.

Обучающие презентации – используются при обучении персонала, в учебных заведениях при чтении лекций, выполнении лабораторных работ и практических заданий.

Корпоративные презентации – ориентированы на потенциальных инвесторов или освещают финансовую деятельность компании и т. д.

Работа в Microsoft Power Point

Среда разработки презентаций

Рабочее окно программы *Power Point* представляет собой стандартное окно *Windows* и включает строку заголовка, ленту, панель быстрого доступа, рабочее окно, строку состояния.

В рабочем окне размещены три объекта: шаблон слайда, окно структуры документа и заметки к слайду. *Шаблон слайда* занимает большую часть экрана, на нем, собственно, и ведется разработка слайда. *Окно структуры* документа предназначено для быстрого просмотра и перемещения по слайдам и имеет две вкладки: *Слайды* и *Структура*. В режиме *Слайды* информация выводится в виде последовательности минислайдов. В режиме *Структура* на экран выводится только текстовая информация, разделенная на кадры, рисунки не выводятся. В данном окне можно копировать, вставлять, удалять и перемещать слайды. Заметки к слайду могут содержать дополнительную текстовую информацию, поясняющую содержание текущего слайда.

Структура ленты и строки состояния рабочего окна программы *Power Point* аналогичны структуре ленты и строки состояния этих

объектов в текстовом процессоре *MS Word*. В левой части строки состояния слева указаны номер текущего слайда и общее число слайдов в презентации, тема оформления слайдов, используемый язык редактирования документа. В правой части строки состояния расположены кнопки режимов просмотра, настройки масштаба и кнопка *Вписать слайд в окно*. Кнопка Вписать слайд в окно подгоняет размеры слайда к размеру рабочего окна, сохраняя соотношение сторон слайда.

Методика создания презентаций

Можно указать несколько способов создания презентации: начать с чистого листа и все операции выполнять в *MS Power Point*; использовать для презентации заранее подготовленный текст документа; использовать шаблоны *MS Power Point*; использовать существующую презентацию.

Начать с чистого листа. Этот способ можно использовать, когда мало текстовой информации и предполагается доклад с демонстрацией кадров, давая пояснения. При больших объемах текста этот путь не эффективен.

Использование готового текстового документа. Для этого надо руководствоваться следующим алгоритмом:

1. Выполнить некоторые подготовительные операции над исходным текстом: создать копию текстового документа и использовать ее для дальнейшей работы; выделить в документе фрагменты текста приемлемого объема, которые будут помещены в один слайд, и оформить их стилем заголовка 2; расставить заголовки к каждому фрагменту текста и оформить их стилем заголовка 1; определить текст, который будет помещен в качестве заметок к слайдам; сохранить полученный документ на диске и закрыть документ в текстовом процессоре.

2. Загрузить документ в программу Power Point: ввести команду ***Файл***, ***Открыть***, установить в окне диалога ***Открытие документа*** тип файлов ***Все файлы***; загрузить подготовленный файл презентации. Программа автоматически формирует слайды, внося заголовки, оформленные стилем Заголовок 1, в заголовок слайда, а текст, оформленный стилем Заголовок 2, в слайд.

3. Открыть в текстовом процессоре *MS Word* подготовленный ранее файл презентации: вставить пустые слайды в местах размещения таблиц, формул, рисунков.

4. Добавление новых слайдов в *MS Power Point* осуществляется командой ***Создать слайд*** вкладки ***Главная*** ленты; скопировать и вставить в слайды необходимые рисунки; скопировать и вставить в область Заметки к слайду текст заметок.

Использование шаблона

MS Power Point предоставляет пользователю возможность создать презентацию на определенную тему, используя готовые шаблоны. Для этого введите команду Файл, Создать: в открывшемся окне диалога можно выбрать понравившуюся тему. Представленная здесь же презентация MS Power Point 2010 позволит ознакомиться с возможностями программы загрузив шаблон и открыв его в режиме показа. Можно найти подходящие шаблоны на сайте Office.com.

Использование существующей презентации. В этом случае важно сохранить стиль оформления, а не содержание. Сохраните слайды, которые будут вам необходимы, и замените в них текст. Остальные слайды удалите. В принципе, достаточно оставить заголовок и один из слайдов, чтобы сохранить стиль оформления документа.

Управление внешним видом рабочей среды

Лента содержит несколько постоянно открытых вкладок: Главная, Вид, Вставка, Дизайн, Переходы, Анимация, Показ, Рецензирование. Каждая из вкладок содержит набор команд, сгруппированных по функциональному назначению. Управление внешним видом рабочего окна программы осуществляется с помощью вкладки Вид и группы команд в правой части строки состояния, о которых упоминали ранее. Вкладка Вид содержит несколько групп команд, позволяющих управлять режимами просмотра, настраивать параметры слайдов, выдач, заметок, внешний вид окна редактора, управлять масштабом изображения, открытыми окнами.

Режимы просмотра презентации и образцы документов

MS Power Point предоставляет пользователю четыре режима просмотра презентации: Обычный, Сортировщик слайдов, Чтение и Показ слайдов.

Обычный режим просмотра слайдов предназначен для разработки презентации.

Сортировщик слайдов позволяет быстро просматривать презентацию и осуществлять перемещение отдельных слайдов или групп слайдов в требуемое положение. Этот режим соответствует многостраничному режиму просмотра документов в текстовом процессоре Word.

Режим чтения – это полноэкранный режим просмотра документа. В этом режиме отображаются и спецэффекты, примененные к адрам.

Показ слайдов – это представление презентации целевой аудитории. Управление показом в ручном режиме осуществляется с помощью клавиш управления курсором «вверх» и «вниз», PgUp, PgDn, а также команд контекстного меню. Команда Страница заметок из группы Режимы просмотра презентаций позволяет просмотреть страницу заметок вместе со слайдом.

Создание слайдов

MS Power Point 2010 имеет мощные средства для разработки слайдов: ввода и редактирования текста, вставки различных объектов. Все необходимые элементы для создания слайда сосредоточены на вкладке Главная. На ней размещены:

– группа Буфер обмена – содержит команды вырезания, копирования, вставки с использованием буфера обмена, форматирования по образцу – достаточно щелкнуть дважды мышкой по кнопке Форматирование по образцу, чтобы применить этот формат к другим частям документа;

– группа Слайды позволяет создать новый слайд и предоставляет пользователю шаблоны размещения информации на слайде;

– группы Шрифт, Абзац, Рисование, Редактирование аналогичны соответствующим командам MS Word 2010.

Вкладка Рецензирование позволяет проверять орфографию, добавлять примечания к тексту заметок, пользоваться справочниками, переводчиком текста с других языков, выбирать язык редактирования справок, примечаний, интерфейса.

Шаблоны слайдов содержат местозаполнители. Пустые местозаполнители служат для ввода текста, местозаполнители с картинками служат для вставки объектов: таблиц, графиков и диаграмм, рисунков SmartArt, рисунков из файлов, картинок, клипов мультимедиа и даже фильмов.

Вставка объектов

Вкладка Вставка MS Power Point 2010 мало чем отличается от одноименной вкладки ленты MS Word 2010. Здесь наше внимание может привлечь группа Мультимедиа. Она предоставляет возможность вставки видеоматериалов из файла, видеосайта, организатора клипов. Список типов видеофайлов, которые можно использовать в S Power Point, практически не ограничен. Для воспроизведения файла звука или видео требуется универсальный проигрыватель, который

воспроизводит мультимедийные файлы и управляет такими устройствами воспроизведения, как приводы компакт- и видеодисков.

Группа Ссылки позволяет вставить гиперссылки на web-страницу, рисунок, адрес электронной почты или программу.

Объект SmartArt служит для визуального представления информации в виде списков, последовательности переходов, иерархии, отражения связи между объектами. После выбора элемента SmartArt открывается вкладка ленты для работы с данным объектом, содержащая две вкладки: Конструктор и Формат, позволяющие вести настройку параметров объектов.

Оформление слайдов и представление презентации

Выполняется оформление слайдов с помощью вкладок Дизайн, Переходы и Анимация.

Вкладка Дизайн предоставляет пользователю возможность выбрать тему из предлагаемого списка. Выбранная тема может быть применена ко всем слайдам или к выделенным слайдам. Обычно выбирается одинаковый тип для всех слайдов. Исключение можно сделать для титульного листа.

Вкладка Переходы позволяет создать эффекты, происходящие при смене кадра. После выбора варианта перехода можно изменить его параметры. Здесь же можно подобрать звук, сопровождающий смену кадра и временные параметры перехода.

Вкладка Анимация – это добавление к тексту или объекту специального видео- или звукового эффекта. Можно применить анимацию к тексту, рисункам, фигурам, таблицам, графическим элементам SmartArt и другим объектам в презентации. К одному и тому же объекту анимационные эффекты можно применять последовательно один за другим.

Подготовка и представление презентации

Настройка параметров представления презентации осуществляется с помощью команд вкладки Показ слайдов. Power Point позволяет использовать несколько режимов показа слайдов: непрерывный показ с начала, показ с текущего кадра, произвольный показ только отдельных кадров, широковещательный показ удаленным зрителям, использующим средства web-браузера.

Создание страниц заметок

Страницы заметок можно создавать непосредственно во время разработки каждого слайда. По окончании создания презентации страницы

заметок можно оформить с помощью команды Вид, Образец заметок. При печати слайдов в верхней части страницы выводится миниатюра слайда.

Раздаточные материалы

Раздаточные материалы – это распечатанные слайды. Подготовка раздаточного материала к печати осуществляется командой Вид, Образец выдач. После ввода команды появляется вкладка Образец выдач. На этой вкладке можно установить параметры страницы, ориентацию выдач (книжная или альбомная), ориентацию слайда (также книжная или альбомная), число слайдов на странице, указать необходимость вывода верхнего и нижнего колонтитулов, даты и времени, изменить тему и фон.

Сохранение и использование презентаций

Сохранение презентации осуществляется командами Сохранить, Сохранить как и Сохранить и отправить меню Файл. Первые две команды сохраняют файлы на своем компьютере. Команда Сохранить и отправить позволяет выполнить при сохранении некоторые преобразования и отправить файл по назначению. В частности, при сохранении презентацию можно преобразовать в видеофайл – команда Создать видео.

При отправке презентации по электронной почте рекомендуется использовать форматы .pdf и .xps. Достоинством этих форматов является то, что документы этих форматов одинаково выглядят на большинстве компьютеров, сохраняются шрифты, исходное форматирование и изображения, содержимое нельзя легко изменить.

Контрольные вопросы

1. Что такое презентация, слайд?
2. Что такое раздаточный материал, конспект доклада?
3. Перечислите этапы создания презентации с использованием текстового документа.
4. Какие существуют режимы просмотра презентации?
5. Какие объекты могут использоваться для оформления презентации?
6. Какие эффекты можно использовать для улучшения восприятия материала?
7. Как подготовить раздаточный материал?
8. Как подготовить конспект доклада?
9. Как производится настройка параметров показа презентации?
10. Какие способы представления презентации вам известны, каковы их особенности?

3. СЕТЕВЫЕ ТЕХНОЛОГИИ И ИНТЕРНЕТ

3.1. Классификация компьютерных сетей

Как и любые объекты исследований, компьютерные сети можно классифицировать по различным признакам.

По территориальной распространенности – это глобальные и локальные компьютерные сети.

Глобальные сети (WAN, Wide Area Networks) позволяют организовать взаимодействие между компьютерами на больших расстояниях. Эти сети работают на относительно низких скоростях и могут вносить значительные задержки в передачу информации. Протяженность глобальных сетей может составлять тысячи километров и они интегрированы с сетями масштаба страны.

Локальные сети (LAN, Local Area Networks) обеспечивают наивысшую скорость обмена информацией между компьютерами и типичная локальная сеть занимает пространство в одно или несколько зданий. Протяженность локальных компьютерных сетей составляет всего лишь несколько километров.

Сравнительно недавно появились *городские сети* или *сети мегополисов* (MAN, Metropolitan Area Networks). Такие сети предназначены для обслуживания территории крупного города – мегаполиса.

В своем классическом построении глобальные и локальные компьютерные сети отличаются по следующим признакам:

1. Протяженность и качество линий связи. Локальные компьютерные сети по определению отличаются от глобальных сетей небольшими расстояниями между узлами сети. Это делает возможным использование в локальных сетях более качественных линий связи.
2. Сложность методов передачи данных. В условиях низкой надежности физических каналов и различных внутренних стандартах на передачу данных в разных странах в глобальных сетях требуются более сложные, чем в локальных сетях, методы передачи данных и соответствующее оборудование.
3. Скорость обмена данными в локальных сетях (100 Мбит/с–10 Гбит/с) существенно выше, чем в глобальных (128 Кбит/с–56 Мбит/с).

4. Разнообразие услуг. Высокие скорости обмена данными дали возможность реализовать в локальных сетях широкий набор услуг, таких как услуги файловой службы, услуги печати, услуги баз данных и др., в то время как глобальные сети в основном были предназначены для почтовых и файловых услуг с ограниченными возможностями.

5. Масштабируемость. Локальные сети обладают плохой масштабируемостью из-за жесткости базовых топологий, определяющих способ подключения станций и длину линии. При этом характеристики сети резко ухудшаются при достижении определенного предела по количеству узлов или протяженности линий связи. Глобальным сетям присуща хорошая масштабируемость, так как они изначально разрабатывались в расчете на работу с произвольными топологиями и сколь угодно большим количеством абонентов.

Ведомственная принадлежность

По принадлежности различают ведомственные и государственные сети. Ведомственные принадлежат одной организации (компании) и располагаются на ее территории.

Государственные сети – сети, используемые в государственных структурах.

Скорость передачи данных

По скорости передачи информации компьютерные сети бывают:

- низкоскоростные (до 10 Мбит/с);
- среднескоростные (от 10 до 100 Мбит/с);
- высокоскоростные (свыше 100 Мбит/с).

По типу среды передачи данных:

- проводные (коаксиальные, на витой паре, оптоволоконные);
- беспроводные (с передачей информации по радиоканалам, в инфракрасном диапазоне).

Организация взаимодействия компьютеров

С точки зрения организации взаимодействия компьютеров, сети делят на одноранговые и иерархические сети (с выделенным сервером).

3.2. Семиуровневая модель структуры протоколов связи

Существуют два основных метода *коммутации (соединения абонентов)* в сетях – *коммутация каналов и коммутация пакетов*. Сети с *коммутацией каналов* исторически появились первыми

в виде первых телеграфных и телефонных сетей. Коммутация каналов подразумевает образование составного физического канала из последовательно соединенных отдельных канальных участков для прямой передачи данных между узлами сети.

В основе принципа работы большинства компьютерных сетей (в том числе сети Интернет) положен *метод пакетной коммутации.*

Коммутация пакетов – эта схема была специально разработана для компьютерных сетей, где различные компьютеры сети могут иметь различное быстродействие. Это основной метод коммутации в современных сетях. При коммутации пакетов все передаваемые сообщения разбиваются передающим компьютером на небольшие части (от 46 до 1500 байт), называемые *пакетами*. Каждый пакет снабжается заголовком, в котором указывается адресная информация, необходимая для доставки пакета к принимающему компьютеру, а также номер пакета, используемый для «сборки» сообщения на принимающем компьютере. Пакеты транспортируются в сети как независимые информационные блоки. Специальные устройства сети *коммутаторы* принимают пакеты от передающих компьютеров и на основании адресной информации передают их друг другу до конечного принимающего компьютера. За счет буферизации (задержки) пакета во внутренней памяти коммутатора (если требуемый участок сети занят передачей другой информации) выравнивается скорость передачи данных в сети в целом и повышается пропускная способность сети.

С другой стороны, по своей сущности компьютерная сеть является совокупностью компьютеров и сетевого оборудования, соединенных каналами связи. Поскольку компьютеры и сетевое оборудование могут быть разных производителей, то возникает проблема их совместимости. Без принятия всеми производителями общепринятых правил построения оборудования создание компьютерной сети было бы невозможно. Поэтому разработка и создание компьютерных сетей может происходить только в рамках утвержденных *стандартов.*

В основу стандартизации компьютерных сетей положен принцип *декомпозиции,* т. е. разделения сложных задач на отдельные более простые подзадачи. Каждая подзадача имеет четко определенные функции и строго установленные связи между подзадачами. При более внимательном рассмотрении работы компьютера в сети можно выделить две основные подзадачи:

– взаимодействие программного обеспечения пользователя с изическим каналом связи (посредством сетевой карты) в пределах одного компьютера;

– взаимодействие компьютера через канал связи с другим компьютером.

Современное программное обеспечение компьютера имеет многоуровневую модульную структуру, т. е. программный код, написанный программистом и видимый на экране монитора (модуль верхнего уровня), проходит несколько уровней обработки, прежде чем превратится в электрический сигнал (модуль нижнего уровня), передаваемый в канал связи.

При взаимодействии компьютеров через канал связи оба компьютера должны выполнять ряд соглашений. Например, они должны согласовать величину и форму электрических сигналов, длину сообщений, методы контроля достоверности и т. д. Соглашения должны быть такими, чтобы они были поняты каждым модулем на соответствующем уровне каждого компьютера.

Суть работы многоуровневого протокола можно пояснить как «письмо в конверте». Каждый уровень протокола надписывает на «конверте» свою информацию. Сетям нужно только понимать «надпись» на «конверте», чтобы предать его в место назначения, а до содержания письма им дела нет.

На рис. 3.1 схематически показана модель взаимодействия двух компьютеров в сети. Для упрощения показаны четыре уровня модулей для каждого компьютера. Процедура взаимодействия каждого уровня этих компьютеров может быть описана в виде набора правил взаимодействия каждой пары модулей соответствующих уровней.

Формализованные правила, определяющие последовательность и ормат сообщений, которыми обмениваются модули, лежащие на одном уровне, но в различных компьютерах, называются *протоколами*.

Модули, реализующие протоколы соседнего уровня и находящиеся в одном компьютере, также взаимодействуют друг с другом в соответствии с четко определенными правилами и с помощью стандартизованных форматов сообщений. Эти правила называются *интерфейсом* и определяют набор сервисов, предоставляемых данным уровнем соседнему уровню.

Другими словами, в сетевых технологиях традиционно принято, что протоколы определяют правила взаимодействия модулей одного уровня, но в разных компьютерах, а интерфейсы – соседних уровней

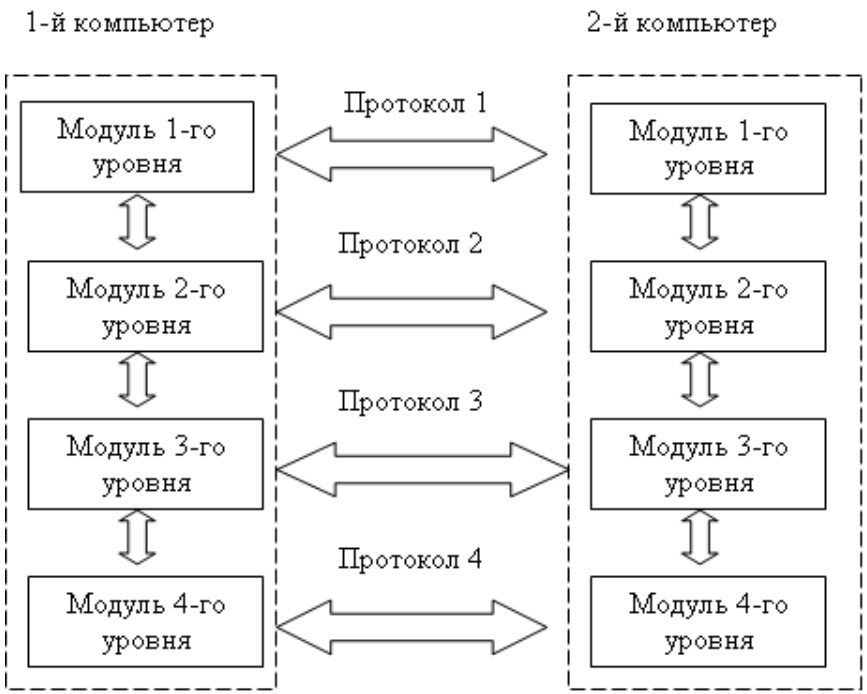


Рис. 3.1. Взаимодействие двух компьютеров в сети

в одном компьютере. Модули должны обрабатывать: во-первых, свой собственный протокол, а во-вторых, интерфейсы с соседними уровнями.

Иерархически организованный набор протоколов для взаимодействия компьютеров в сети называется *стеком коммуникационных протоколов*.

Коммуникационные протоколы могут быть реализованы как программно, так и аппаратно. Протоколы нижних уровней, как правило, реализуются комбинацией программно-аппаратных средств, а ротоколы верхних уровней – чисто программными средствами.

Отметим, что протоколы каждого уровня обладают независимостью друг от друга, т. е. протокол любого уровня может быть изменен, не оказывая при этом никакого влияния на протокол другого уровня. Главное, чтобы интерфейсы между уровнями обеспечивали необходимые связи между ними.

Принцип взаимодействия компьютеров в сети можно объяснить на примере сотрудничества двух фирм. Два генеральных менеджера каждой из фирм осуществляют сделки между собой на основании заключенных договоров и соглашений. Указанные взаимодействия являются «протоколом уровня генеральных менеджеров». На каждой из фирм у менеджеров есть секретари, причем каждый менеджер имеет свой метод и стиль работы с секретарем. Один, например, предпочитает устные указания, а второй дает только письменные распоряжения. Таким образом, каждая фирма имеет свой собственный интерфейс «главный менеджер – секретарь», что не мешает, однако, нормально работать генеральным менеджерам между собой. Секретари в свою очередь договорились обмениваться информацией с помощью факсов, реализуя протокол «секретарь – секретарь». В случае, если секретари перейдут на электронную почту, то генеральные менеджеры этого даже и не заметят – главное, чтобы секретари выполняли их распоряжения, т. е. должен безукоризненно работать интерфейс «менеджер – секретарь». С другой стороны, менеджеры могут заключить совершенно новый договор, т. е. изменить «протокол уровня генеральных менеджеров». Передача не старого, а нового договора на уровне секретарей пройдет для этих секретарей абсолютно не замеченной.

В рассмотренном примере мы определили два уровня протоколов – уровень генеральных менеджеров и уровень секретарей. Каждый из указанных уровней имеет свой собственный протокол, который может быть изменен независимо от протокола другого уровня. Такую независимость обеспечивает правильное функционирование интерфейсов «менеджер – секретарь». Независимость протоколов каждого уровня друг от друга и взаимодействие самих уровней посредством интерфейсов является важнейшей предпосылкой для создания ряда стандартных протоколов для компьютерных сетей.

В начале 80-х гг. была создана модель взаимодействия двух компьютеров в сети, названная OSI (Open system interconnection – взаимодействие открытых систем). Она была предложена Международной организацией по стандартизации (ISO – International Standards Organization). В модели OSI реализована более подробная, семиуровневая архитектура, причем каждому из уровней соответствует определенная функция (рис. 3.2).

Рассмотрим принцип взаимодействия двух компьютеров в рамках модели *OSI*.

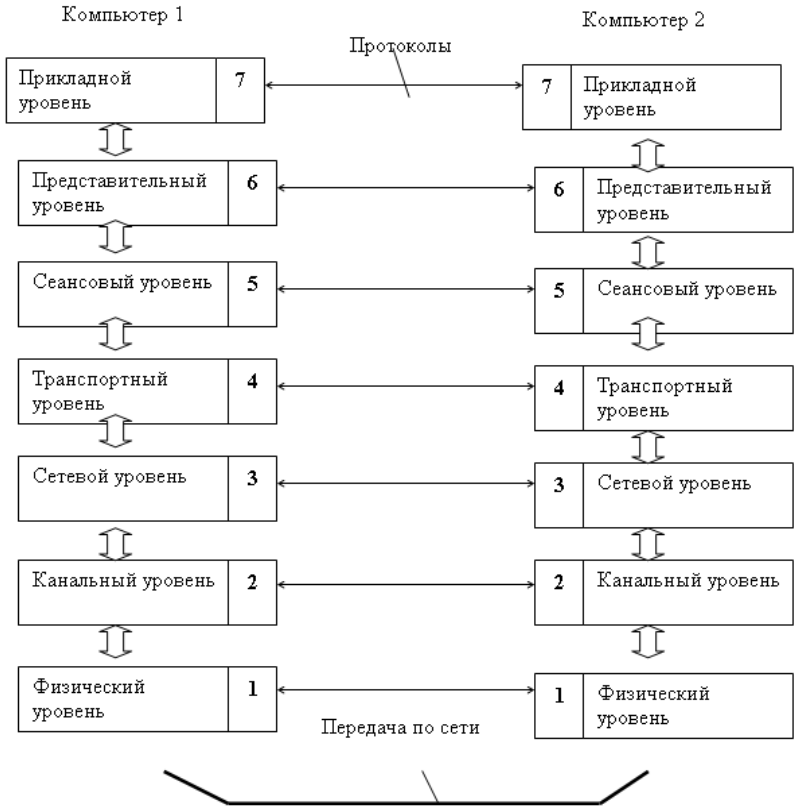


Рис. 3.2. Модель OSI

Взаимодействие компьютеров в сети начинается с того, что приложение (программа пользователя) одного компьютера обращается к прикладному уровню другого компьютера, например, к файловой системе. Приложение первого компьютера формирует с помощью операционной системы сообщение стандартного формата, состоящее из *заголовка и поля данных*.

Заголовок содержит служебную информацию, которую необходимо предать через сеть прикладному уровню другого компьютера, чтобы сообщить ему, какую работу необходимо выполнить. (Например, о размере файла и где он находится). Кроме этого в заголовке имеется информация для следующего нижнего уровня, чтобы

он «знал», что делать с этим сообщением. В поле данных находится информация, которую необходимо поместить в найденный файл.

Сформировав сообщение, прикладной уровень направляет его вниз представительному уровню. Прочитав заголовок, представительный уровень выполняет требуемые действия над сообщением и добавляет к сообщению собственную служебную информацию – заголовок представительного уровня, в котором содержаться указания для протоколов представительного уровня второго компьютера. Полученное в результате сообщение передается вниз сеансовому уровню, который в свою очередь добавляет свой заголовок и т. д. При достижении сообщением нижнего, физического уровня, у него имеется множество заголовков, добавленных на каждом предыдущем уровне (сообщение вложено внутрь, как в матрешку). В таком виде оно и передается по сети (рис. 3.3, *а*).

Второй компьютер принимает его на физическом уровне и последовательно перемещает его вверх с уровня на уровень (рис. 3.3, *б*). Каждый уровень анализирует и обрабатывает заголовок своего уровня, выполняя соответствующие этому уровню функции, а затем удаляет этот заголовок и передает сообщение дальше вышележащему уровню. Отметим, что сообщение может оканчиваться также некоторой служебной информацией – *концевиком».* (Например, дополнительными контрольными разрядами).

Как видно из рис. 3.3, информация, передающаяся в линию связи содержит большое количество служебных заголовков, которые по величине могут превосходить даже собственно данные. В результате взаимодействия протоколов всех уровней по единому стандарту на прикладном уровне второго компьютера получаются данные, переданные первым компьютером.

В стандарте *OSI* для обозначения единиц данных, с которыми имеют дело протоколы различных уровней, используются специальные названия: *кадр (frame), пакет (packet), дейтаграмма (datagram), сегмент (segment).*

Рассмотри подробнее уровни модели OSI.

Физический уровень (Physical layer) имеет дело с передачей битов информации по физическим каналам связи. Такими каналами могут быть, например, коаксиальный кабель, витая пара, оптоволоконный кабель. На этом уровне стандартизируются характеристики электрических сигналов, уровни напряжения и тока, тип кодировки

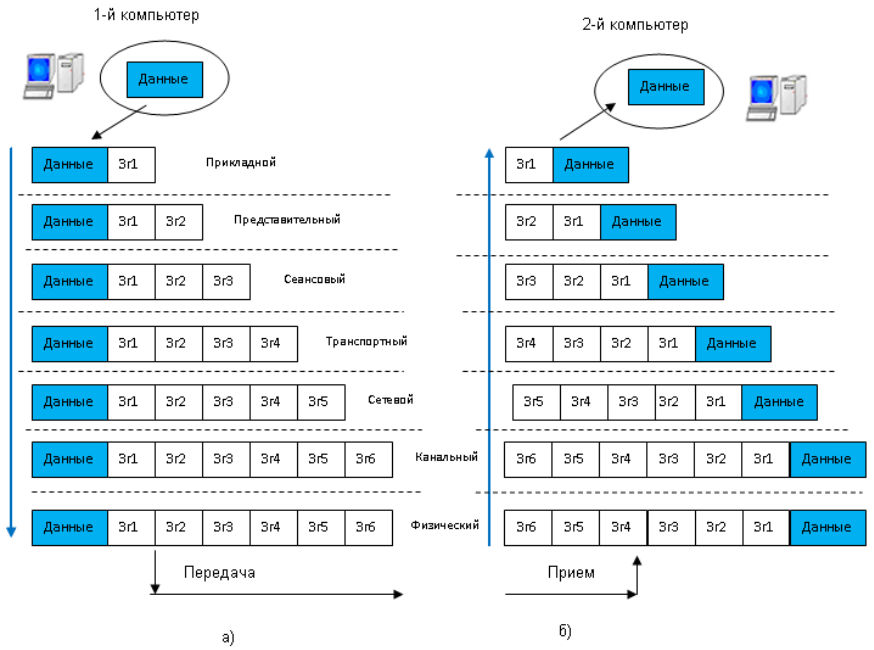


Рис 3.3. Взаимодействие компьютеров в сети

информации, скорость передачи сигналов, а также типы разъемов и назначение каждого контакта.

Канальный уровень (Data Link layer) обеспечивает надежную передачу данных через физический канал. Канальный уровень оперирует блоками данных, называемых *кадрами*. Основной задачей канального уровня является прием кадра из сети и отправка его в сеть. При выполнении этой задачи канальный уровень осуществляет физическую адресацию передаваемых сообщений, контролирует соблюдение правил использования физического канала, выявляет неисправности, управляет потоками информации. Примерами протоколов канального уровня для локальных сетей являются протоколы Ethernet, Token Ring, FDDI, для глобальных – PPP, SLIP, LAP-B, LAP- D.

Для реализации протоколов канального уровня используется специальное оборудование – *коммутаторы*. Раньше использо-

вались *концентраторы и мосты,* которые в настоящее время сняты с производства.

Сетевой уровень (Network layer) служит для образования единой системы, объединяющей несколько сетей, причем эти сети могут быть различной топологии, использовать совершенно различные принципы сообщений между конечными узлами и обладать произвольной структурой. Сети соединяются между собой специальными устройствами, называемыми маршрутизаторами. Маршрутизатор – это устройство, которое собирает данные о топологии межсетевых соединений и на ее основании пересылает пакеты информации из одной сети в другую. Последовательность маршрутизаторов, через которые проходит пакет, называется *маршрутом*, а выбор маршрута называется *маршрутизацией*. Маршрутизация является главной задачей сетевого уровня. На сетевом уровне действуют три протокола: *сетевой протокол* – для определения правил передачи пакетов от конечных узлов к маршрутизаторам и между маршрутизаторами; *протокол маршрутизации* – для сбора информации о топологии сети; *протокол разрешения адресов* – для отображения адреса узла, используемого на сетевом уровне в локальный адрес сети (ARP-адрес).

Транспортный уровень (Transport layer) предназначен для оптимизации передачи данных от отправителя к получателю с той степенью надежности, которая требуется. Основная задача транспортного уровня – это обнаружение и исправление ошибок в сообщениях, пришедших с описанных выше уровней.

Начиная с транспортного уровня, все дальнейшие протоколы реализуются программным обеспечением компьютера, включаемого обычно в состав сетевой операционной системы.

Сеансовый уровень (Session layer) управляет диалогом между двумя компьютерами. На этом уровне устанавливаются правила начала и завершения взаимодействия. На сеансовом уровне определяется, какая из сторон является активной в данный момент, а какая принимает данные. Приложение должно различать разные потоки данных в пределах одного соединения. Например, приложение может одновременно запрашивать два файла с одного сервера, при этом, благодаря сеансовому уровню, оно будет различать эти два потока.

Представительный уровень (Presentation layer) выполняет преобразование данных между устройствами с различными форматами данных, не меняя при этом содержания. Благодаря этому уровню информация, передаваемая прикладным уровнем одного компьютера всегда понятна прикладному уровню другого компьютера. На этом уровне, как правило, происходит шифрование и дешифрование данных, благодаря которому обеспечивается секретность предаваемого сообщения.

Прикладной уровень (Application layer) является пользовательским интерфейсом для работы с сетью. Этот уровень непосредственно взаимодействует с пользовательскими прикладными программами, предоставляя им доступ в сеть. С помощью протоколов этого уровня пользователи сети получают доступ к разделяемым ресурсам, таким как файлы, принтеры, гипертекстовые *web*-страницы, электронная почта и т. д.

Необходимо отметить, что три нижних уровня модели *OSI* – физический, канальный и сетевой – зависят от сети, т. е. их протоколы тесно связаны с технической реализацией сети и с используемым коммутационным оборудованием. Три верхних уровня – сеансовый, представления и прикладной – ориентированы на программное обеспечение и мало зависят от особенностей построения сети (топологии, оборудования и т. д.). Транспортный уровень является промежуточным. Он скрывает детали функционирования нижних уровней от верхних уровней. Благодаря этому уровню можно разрабатывать приложения, не зависящие от технических средств транспортировки сообщений.

Модель OSI является так называемой *открытой системой*, т. е. она имеет опубликованные, общедоступные спецификации и стандарты, принятые в результате достижения согласия многих разработчиков и пользователей после всестороннего обсуждения. Эта модель доступна всем разработчикам и для ее использования не требуется получение специальных лицензий. Если две сети построены с соблюдением правил открытости, то у них есть возможность использования аппаратных и программных средств разных производителей, придерживающихся одного и того же стандарта. Такие сети легко сопрягаются друг с другом, просты в освоении и обслуживании.

3.3. Организационная структура Интернет

Ярким примером открытой системы является глобальная компьютерная сеть Интернет. Эта сеть развивалась в полном соответствии с требованиями, предъявляемыми к открытым системам. В разработке стандартов Интернет принимали и принимают участие тысячи специалистов и пользователей этой сети из различных университетов, научных организаций и фирм-производителей вычислительной аппаратуры и программного обеспечения из разных стран.

Само название стандартов, определяющих работу Интернет – Request For Comments – переводится как «запрос на комментарии», т. е. для введения новых стандартов в этой сети проводится открытый опрос мнений пользователей и только затем вносятся изменения. В результате сеть Интернет сумела объединить в себе самое разнообразное оборудование и программное обеспечение огромного числа сетей, разбросанных по всему миру.

Стек протоколов TCP/IP *(*Transmission Control Protocol/Интернет Protocol*)* является базовым протоколом сети Интернет. Протокол TCP/IP предложил сотрудник DARPA Роберт Канн в 1972 г. для взаимодействия компьютера с открытой сетевой архитектурой.

Основными преимуществами протокола TCP/IP являются:

1. *Независимость от сетевой технологии отдельной сети.* TCP/ IP не зависит от оборудования, так как он определяет только элемент передачи, который называется *дейтаграммой,* и описывает способ ее движения по сети.

2. *Всеобщая связанность сетей.* Протокол позволяет любой паре компьютеров взаимодействовать друг с другом. Каждому компьютеру назначается логический адрес, а каждая передаваемая дейтаграмма содержит адреса отправителей и получателей. Промежуточные маршрутизаторы используют адрес получателя для принятия решения о маршрутизации.

3. *Подтверждение.* Протокол TCP/IP обеспечивает подтверждение правильности прохождения информации при обмене между отправителем и получателем.

4. *Стандартные прикладные протоколы.* Протокол TCP/IP включает в свой состав поддержку основных приложений, таких как электронная почта, передача файлов, удаленный доступ и т. д.

В стеке TCP/IP определены 4 уровня взаимодействия, каждый из которых берет на себя определенную функцию по организации надежной работы глобальной сети:

Уровень I	Прикладной уровень
Уровень II	Основной (транспортный) уровень
Уровень III	Уровень межсетевого взаимодействия
Уровень IV	Уровень сетевых интерфейсов

Соответствие стека *TCP/IP* модели *OSI* показано на рис. 3.4.

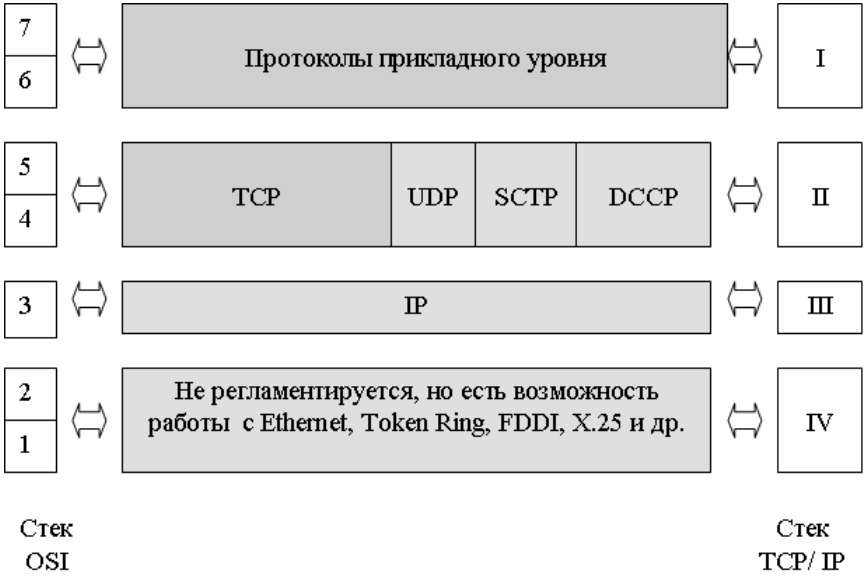


Рис. 3.4. Соответствие стека TCP/IP модели OSI

Как видно из рис. 3.4, протокол TCP занимает транспортный и сеансовый уровень, а на сетевом уровне используется протокол IP.

IP-адресация компьютеров в сети построена на концепции сети, состоящей из хостов. *Хост* представляет собой объект сети, который может передавать и принимать *IP*-пакеты, например, компьютер или маршрутизатор.

Отметим, что в модели TCP/IP программные модули, соответствующие транспортному и сеансовому уровню, устанавливаются только на конечных компьютерах.

Программный модуль протокола TCP/ IP реализуется в операционной системе компьютера в виде отдельного системного модуля (драйвера). Интерфейс между прикладным уровнем и TCP представляет собой библиотеку вызовов, такую же, как, например, библиотека системных вызовов для работы с файлами. Пользователь может самостоятельно настраивать протокол TCP/ IP для каждого конкретного случая (количество пользователей сети, пропускная способность физических линий связи и т. д.).

Уровень межсетевого взаимодействия

Уровень межсетевого взаимодействия является стержнем всей архитектуры протокола, который реализует концепцию передачи пакетов в режиме без установления соединений, то есть дейтаграммным способом. Именно этот уровень обеспечивает возможность перемещения пакетов по сети, используя тот маршрут, который в данный момент является оптимальным. Этот уровень также называют *уровнем Интернет*, подчеркивая его основную функцию – передачу данных через составную сеть. Основным протоколом уровня межсетевого взаимодействия является протокол IP (*Интернет Protocol*). IP-протокол проектировался для передачи пакетов в составных сетях, состоящих из большого количества локальных сетей, поэтому он хорошо работает в сетях со сложной топологией. Так как IP-протокол является дейтаграммным протоколом, то он не гарантирует доставку пакетов до узла назначения.

Основной (транспортный) уровень

Так как на сетевом уровне не происходит установление соединения, то нет никаких гарантий, что межсетевым уровнем пакеты будут доставлены в место назначения неповрежденными.

Обеспечения надежной связи между двумя конечными компьютерами осуществляет основной уровень стека TCP/IP, называемый также *транспортным*.

На этом уровне работают основные протоколы:

– TCP (Transmission Control Protocol) – основной протокол управления передачей;

– UDP (User Datagramm Protocol) – протокол дейтаграмм пользователя;

– SCTP (Stream Control Transmission Protocol) – протокол передачи с управлением потоком;

– DCCP (Datagram Congestion Control Protocol) – предоставляет механизмы для отслеживания перегрузок в сети, избегая необходимости создавать их на прикладном уровне.

Прикладной уровень

Прикладной уровень объединяет все службы пользователей сети. Прикладной уровень реализуется различными программными системами и постоянно расширяется. Наиболее известными прикладными службами в сети Интернет являются, например, электронная почта (E-mail), система новостей UseNet, всемирная паутина World Wide Web (WWW), передача файлов (FTP), удаленный терминал и терминальные серверы (TELNET) и др.

Уровень сетевых интерфейсов

В отличие от физического и канального уровня модели OSI в архитектуре стека TCP/IP существует несколько другая интерпретация уровня сетевых интерфейсов. Протоколы этого уровня должны обеспечить интеграцию в составную сеть локальных сетей, использующих различные технологии. Поэтому разработчики той или другой технологии должны предусмотреть возможность инкапсуляции (включения) в свои кадры IP-пакетов. Уровень сетевых интерфейсов в протоколах TCP/IP не регламентируется, но он поддерживает все популярные стандарты физического и канального уровня: Ethernet, Token Ring, FDDI, Gigabit Ethernet, Fast Ethernet и др. Разработаны спецификации для соединения с сетями X.25, frame relay, ATM.

Отметим, что в настоящее время каждый из разработчиков сетевых технологий канального и физического уровня стремиться обеспечить их совместимость с протоколом TCP/IP.

Адресация в сети Интернет

Для адресации компьютеров в сети Интернет используются *IP*-адреса. *IP*-адреса отправителя и получателя принято называть *адресом хоста.*

IP-адрес любого из хостов состоит *из адреса (номера) сети (сетевого префикса) и адреса хоста в этой сети.*

В соответствии принятым в момент разработки IP-протокола соглашением, адрес представляется четырьмя десятичными числами, разделенными точками. Каждое из этих чисел не может превышать 255 и представляет один байт 4-байтного IP-адреса. Выделение всего лишь четырех байт для адресации всей сети Интернет связано с тем, что в то время массового распространения локальных сетей пока не предвиделось. О персональных компьютерах и рабочих станциях вообще не было речи.

В результате под IP-адрес было отведено 32 бита, из которых первые 8 бит обозначали сеть, а оставшиеся 24 бита – компьютер в сети. IP-адрес назначается администратором сети во время конфигурирования компьютеров и маршрутизаторов.

Главным координатором в Интернете, занимающимся распределением и управлением IP-адресов, доменов и обратных доменов, является некоммерческая организация IANA (Интернет Assigned Numbers Authority), агентство по выделению имен и уникальных параметров протоколов Интернет). www.iana.org/, работающая под управлением организации Интернет Corporation for Assigned Names and Numbers, или ICANN— международной некоммерческой организации при участии правительства США для регулирования вопросов, связанных с доменными именами, IP-адресами и прочими аспектами функционирования Интернета.

Отметим, что маршрутизатор может входить сразу в несколько сетей, поэтому каждый порт маршрутизатора имеет свой IP-адрес. Конечный компьютер также может входить в несколько сетей, а значит иметь несколько IP- адресов. Таким образом, IP-адрес характеризует не отдельный компьютер или маршрутизатор, а одно сетевое соединение. Как уже отмечалось выше, адрес состоит из двух частей – номера сети и номера узла в сети. Для того, чтобы определить, какая часть адреса относится к номеру сети, а какая к номеру узла, в начале адреса несколько бит отводится для определения класса сети.

Основные сервисы сети Интернет

Сервисы Интернет – это сервисы, предоставляемые в сети Интернет пользователям, программам, системам, уровням, функциональным

блокам. В сети Интернет сервисы предоставляют сетевые службы. Наиболее распространенными Интернет-сервисами являются:

– электронная почта (*E-mail*), обеспечивающая возможность обмена сообщениями одного человека с одним или несколькими абонентами;

– телеконференции, или группы новостей (Usenet), обеспечивающие возможность коллективного обмена сообщениями;

– сервис *FTP* – система файловых архивов, обеспечивающая хранение и пересылку файлов различных типов;

– сервис *Telnet* предназначен для управления удаленными компьютерами в терминальном режиме;

– *World Wide Web* (*WWW*, *W3*) – гипертекстовая (гипермедиа) система, предназначенная для интеграции различных сетевых ресурсов в единое информационное пространство;

– сервис *DNS* (система доменных имен) обеспечивающий возможность использования для адресации узлов сети мнемонических имен вместо числовых адресов;

– сервис *IRC*, предназначенный для поддержки текстового общения в реальном времени (chat).

Базовые сервисы сети Интернет

IP-телефония – новая услуга, предлагаемая провайдерами, основанная на преобразовании голоса в данные, которые могут быть переданы через Интернет. Сегодня развитие этой технологии достигло такой стадии, что услуги IP-телефонии стали достаточно надежными и качественными, чтобы рассматриваться как альтернатива услугам традиционной телефонной связи. Большинство клиентов используют новую технологию для снижения стоимости услуг телефонной связи при сохранении, по возможности, качества связи без ввода дополнительных ограничений для пользователей. Экономия средств, происходит главным образом за счет компрессии голоса, интеграции сетей передачи голоса и данных и за счет более привлекательных (на сегодня) тарифов в Интернет по сравнению с услугами традиционных телефонных операторов.

Электронная почта также является одной из первых служб Интернет, появившейся в середине 70-х гг. Основная концепция, лежащая в основе электронной почты, достаточно проста: вы выходите

в компьютерную систему, набираете и адресуете текстовое сообщение пользователю другой системы. Затем сообщение курсирует по лабиринту взаимосвязанных компьютерных систем до тех пор, пока не будет доставлено адресату. Сейчас в Интернет существуют бесплатные службы, разработанные исключительно для работы электронной почты, которые вы можете использовать в тех случаях, если Интернет вас интересует только как возможность передачи электронных сообщений.

Практически вся информация в компьютерном мире хранится в виде файлов. На этапе зарождения Интернета появилось специальное средство для обмена файлами по сети – сетевой протокол FTP (что, собственно, и расшифровывается как File Transfer Protocol – протокол передачи файлов).

Телеконференции представляют собой что-то вроде клубов по интересам, объединяющих пользователей определенной программы, поклонников известных кинорежиссеров или популярных телеведущих, других людей, связанных общими интересами и взглядами. Телеконференции стали следующим этапом развития систем почтовых сообщений. Их особенностями стали, во-первых, хранение сообщений и предоставление заинтересованным лицам доступа ко всей истории обмена, а во-вторых, группировка сообщений в темы, направления обсуждений.

Гостевые книги – первая и самая простая форма организации общения в виде web-приложений. Простейшая гостевая книга представляет собой список сообщений, показанных от последних к первым. Каждый посетитель может оставить свое сообщение.

Форумы – это форма общения является практически прямой реализацией идеологии телеконференций. Сообщения пользователей в форумах группируются по темам, которые задаются, как правило, первым сообщением. Все посетители могут увидеть тему и разместить свое сообщение – в ответ на уже написанные. Исторически первые форумы появились как усовершенствование гостевых книг и организовывали сообщения в ветви – как и в телеконференциях. Самым распространенным видом форумов сейчас являются форумы табличные, в которых обсуждение темы идет линейно, – это позволяет быстрее прочесть обсуждение. Как правило, темы группируются в тематические форумы, управление системой осуществляют администраторы и модераторы.

Блоги (от англ. *web log* – web-журнал, web-протокол). В этих сервисах каждый участник ведет журнал, т. е. оставляет записи в хронологическом порядке. Темы записей могут быть различными; самый распространенный подход – это ведение блога как собственного дневника. Другие посетители могут оставлять комментарии на эти записи. Чаще всего блог ведут не на своем отдельном сайте (хотя исторически именно эта форма была первой), а в рамках крупной системы, похожей на общедоступный почтовый сервис.

В этом случае пользователь, помимо возможности вести свой журнал, получает возможность организовывать ленту просмотра – список записей из журналов друзей, регулировать доступ к записям, искать себе собеседников по интересам. На базе таких систем создаются сообщества – журналы, которые ведутся коллективно. Например: Блоги@Mail.Ru: Школьный портал.

Любые средства коммуникации служат для общения людей друг с другом. С развитием телекоммуникаций все большее количество пользователей начинают работать в Сети в режиме on-line. Для них полезно иметь возможность взаимодействовать в режиме реального времени, когда абонент получает сообщение практически мгновенно (задержка в ответе не становится принципиальным перерывом в обмене). К ним можно отнести:

IRC (Интернет Relay Chat – трансляция разговора в Интернет) – чрезвычайно популярная служба Интернет. IRC является системой, которая позволяет пользователям общаться друг с другом, подключившись к одному серверу IRC. Однако в ходе общения вы не говорите, а набираете свои реплики на клавиатуре.

ICQ – это также очень популярное средство «онлайнового» общения пользователей через Интернет. Название этой программы является омонимом и образовано из созвучия слов I seek you («Я ищу тебя») и трехбуквенного сочетания ICQ. Существуют и другие способы общения в режиме on-line в сети.

Социальная сеть – это виртуальная сеть, являющаяся средством обеспечения сервисов, связанных с установлением связей между его пользователями, а также разными пользователями и соответствующими их интересам информационными ресурсами, установленными на сайтах глобальной сети.

Социальные сервисы представляют собой онлайновые инструменты, с помощью которых пользователи могут не только

общаться между собой, но и сами создавать контент *web*-страниц. Создание единого *Web 2.0* (*Web* второго поколения) характеризует информационное пространство, состоящее из множества информационных единиц, сети документов, которые распределены по различным сайтам и сервисам. Эта сеть превращается в сеть данных, поиск которых производится пользователями с применением наиболее удобных для них инструментов, интерфейсов, технологий и сервисов, которые обеспечивают доступ к содержимому сайтов.

Web-сервис, который позволяет пользователям систематизировать ссылки, описания, снабжая их поисковыми критериями называется *социальными закладками*.

Социальные геосервисы – сервисы сети Интернет, которые позволяют находить, отмечать, комментировать, снабжать фотографиями различные объекты в любом месте на изображении Земного шара с достаточно высокой точностью, используются реальные данные, полученные с помощью околоземных спутников.

Адреса социальных геосервисов

http://maps.google.com/ – Карты Google

http://wikimapia.org/ – Карты Google + WikiWiki

http://earth.google.com/ – Объемная модель Земли Google

http://panoramio.ru – Фотосервис с возможностью привязки к цифровым картам

Сервис публикации фотографий. На этом сервисе каждый посетитель имеет возможность опубликовать свои фотографии, указав их поисковые признаки. Фотографии можно оценивать и комментировать.

YouTube. Самый быстрорастущий сайт в интернете: хранилище видеороликов обо всем на свете, которые может закачать любой желающий.

Интернет-аукцион. Интернет-аукцион – электронная торговая система, в которой продажа товаров происходит непосредственно между людьми.

Интернет-телевидение IPTV – это трансляция программ телевидения по мультисервисной сети. Обычно провайдер получает сигнал, например, со спутника и передает его по специальному протоколу (MMS, RTSPM и другие) конечным пользователям, которые могут смотреть видео как на экране компьютерного монитора, так и на телевизоре.

3.4. Инструментальные средства создания *web*-сайтов. Основы *web*-дизайна

Создание *web*-сайта состоит из нескольких этапов и взаимосвязанных процессов, таких как проектирование сайта, создание макетов его страниц, написания наполнения и помещения его на сайт, обслуживание сайта и его программной основы.

Одним из этапов разработки *web*-сайта является *web*-дизайн, который в узком смысле термина означает визуальное оформление *web*-страниц. В какой-то степени это аналогия верстки газеты или журнала или создание изображения в полиграфическом дизайне. В то же время web-дизайн может включать в себя и проектирование структуры сайта, его навигации и в некоторых случаях даже движков ресурса. Другими словами, продукт web-дизайна должен включать в себя не только визуальные аспекты сайта, но и его юзабилити – т. е. удобство при использовании.

Пять областей охватывают основные аспекты web-дизайна.

Содержимое. Сюда входят форма и организация содержимого сайта. Возможный диапазон – от того, как написан текст до того, как он организован, представлен и структурирован с помощью технологии разметки, такой как HTML.

Зрительные образы. Это относится к компоновке экранного пространства на сайте. Эта компоновка обычно создается с помощью HTML, CSS или даже Flash и может включать графические элементы, выполняющие функции украшения или навигации. Визуальная сторона сайта – это наиболее очевидный аспект *web*-дизайна, но не единственная и не самая важная сторона дисциплины.

Технология. Хотя применение разнообразных базовых Web-технологий вроде HTML или CSS попадает в эту категорию, под технологией в этом контексте чаще подразумеваются различные интерактивные элементы сайта, в особенности созданные с использованием программных методов. Это могут быть элементы в диапазоне от языков сценариев, работающих на стороне клиента, наподобие JavaScript, до серверных приложений, таких как Java-сервлеты, PHP-сценарии.

Доставка. Скорость и безотказность доставки сайта по сети Интернет или внутренней корпоративной сети связаны с применяемым аппаратным программным обеспечением и задействованной сетевой архитектурой.

Назначение. Причина, по которой сайт существует, часто связанна с экономическими вопросами. Это следует учитывать, принимая любые решения. Конечно, степень, в которой каждая сторона *web*-дизайна оказывает воздействие на сайт, может изменяться в зависимости от типа создаваемого сайта. Личная домашняя страничка обычно не связана с экономическими соображениями, характерными для Интернет-магазина. Внутренняя сеть производственной компании может не попадать под влияние соображений, связанных с визуальным представлением, важных для общедоступного *web*-сайта, рекламирующего остросюжетный фильм.

В настоящее время большинству сайтов свойственна динамичность и интерактивность. Для реализации этих параметров используются web-приложения. Это готовые программные комплексы для решения задач сайта. *Web*-приложение входит в состав сайта, но само по себе без содержимого сайтом является только технически – это оболочка или шаблон, который необходимо наполнить и активизировать. Как раз этим занимаются специалисты по продвижению и раскрутке сайтов.

Как правило, одному сайту соответствует одно доменное имя. Именно по нему любой сайт идентифицируется в глобальной сети. Впрочем, это не единственный возможный вариант. Один сайт может размещаться на нескольких доменах, а также несколько сайтов могут существовать под одним доменом.

За последние десятилетия всемирная паутина стала отличной информационно-рекламной платформой и потому компании различного масштаба (от крупных транснациональных и мировых корпораций до частных предпринимателей), в том числе не связанные непосредственно с деятельностью в сети, создают собственные сайты следующих типов:

– *Сайт-визитка.* На таких сайтах размещаются самые общие данные о владельце сайта. Предоставляется информация о виде деятельности, истории бизнеса, информация о сотрудниках, прайслист, контактные данные, реквизиты, схема проезда.

– *Представительский сайт.* От описанной выше визитки отличается расширенной функциональностью: подробное описание услуг, портфолио, отзывы, форма обратной связи и т. д.

– *Корпоративный сайт.* Содержит полную информацию о компании-владельце, услугах/продукции, событиях в жизни компании.

Отличается от предыдущих двух типов сайтов полнотой представленной информации, зачастую содержит различные функциональные инструменты для работы с контентом (поиск и фильтры, календари событий, фотогалереи, корпоративные блоги, форумы). Может содержать закрытые разделы для тех или иных групп пользователей – сотрудников, дилеров, контрагентов и пр.

Процесс разработки сайтов проходит в несколько этапов.

1. Разработка макетов шаблонов *web*-страниц. Этим занимаются *web*-дизайнеры, в задачи которых входит: определить, каким образом конечный потребитель будет получать доступ к информации и услугам сайт, т. е. разработать пользовательский интерфейс.

Готовые шаблоны предоставляются заказчику на одобрение. Чтобы макеты выглядели более наглядно, в них помещается произвольное содержимое. Если заказчик удовлетворен внешним видом шаблонов, то наступает следующая фаза разработки – верстка страниц сайта.

2. Верстальщик получает макеты шаблонов в виде простых изображений (например, в формате JPEG или PNG), либо разбитых по слоям (например, в PSD или AI). Его основная задача – получить из этих графических макетов гипертекстовые web-страницы с подготовленными для интернета изображениями.

Самое сложное на этом этапе – обеспечить совместимость со множеством браузеров, так как в некоторых из них одни и те же элементы разметки могут отображаться не так, как задумано. Когда достигнуто правильное отображение в большинстве браузеров, переходят к завершающему этапу.

3. Web-программирование. Программисту передаются готовые шаблоны страниц, а также указания дизайнеров по работе и организации элементов сайта. С нуля создается программная основа сайта. Выбор языка программирования в данном случае – вопрос непринципиальный. После того, как сайт готов к эксплуатации, остается наполнить его задуманной информацией.

Существует множество инструментов, с помощью которых web-дизайнер осуществляет верстку страниц. Такие программы называются HTML-редакторами. В web-дизайне используется два типа редакторов – визуальные и невизуальные (текстовые).

Первые работают по принципу **WYSIWYG** (от англ. – What You See Is What You Get – что видишь, то и получаешь. Другими слова-

ми, это способ подразумевает, что при редактировании материал выглядит так же, как он и будет выглядеть в конечном результате.

Текстовыми же редакторами, в основном, пользуются профессиональные *web*-дизайнеры, так как такие инструменты подразумевают написание кода самостоятельно. С помощью текстовых редакторов создается чистый программный код, который позволяет *web*-дизайнерам полностью воспроизводить задуманное, без вмешательства инструмента по умолчанию. Существуют также редакторы, которые поддерживают оба принципа работы.

Наиболее известными визуальными редакторами являются следующие программы, некоторые из которых будут рассмотрены ниже.

Визуальные редакторы

Macromedia Dreamweaver MX – профессиональный инструмент для создания web-сайтов и приложений.

Разработчики утверждают, что Macromedia Dreamweaver MX предназначена для проектирования, разработки и администрирования профессиональных web-сайтов и приожений. Кроме того, Dreamweaver легко интегрируется с другими программами от Macromedia, например такими, как Flash. Dreamweaver гораздо больше, чем просто визуальный редактор, это достаточно мощный и сложный инструмент, а всякий сложный инструмент требует, чтобы на его освоение было потрачено какое-то время, прежде, чем пользователь сможет работать в нем.

Adobe GoLive и LiveMotion – визуальный редактор для верстки *web*-страниц.

Microsoft FrontPage – простейший визуальный редактор. *FrontPage* поддерживает HTML, JavaScript (скриптовый язык), CSS (каскадные таблицы стилей), DHTML (сочетание HTML, JavaScript, CSS и DOM – объектной модели документа), совместим с технологиями ASP (создание web-приложений), XML (язык разметки, текстовый формат, применяемый для хранения структурированных данных), VBScript (скриптовый язык программирования), XSL (расширяемый язык таблиц стилей – смена CSS). FrontPage также имеет неплохие возможности по управлению изображениями, flash-роликами. Любой пользователь, имеющий опыт работы с такими приложениями как Word, Excel, Access, не будет испытывать особых сложностей в работе с редактором FrontPage – знакомые панели инструментов, наборы мастеров и редакторов, шаблонов, автоматических функций и т. д. FrontPage – прекрасное решение для начинающих пользователей и *web*-мастеров.

Microsoft Expression Web и SharePoint Designer отличается тем, что предоставляет своим пользователям больше свободы, производит чистый, совместимый со стандартами код, работает с CSS и CSS-макетами. Он представляет собой конструктор и имеет сходство с *SharePoint Designer*.

SharePoint. Среди возможностей – поддержка интерактивных страниц ASP. NET, создание вида данных из RSS, XML, Office XML, сотрудничество со встроенным Workflow Designer, использование инструментов CSS, отслеживание статистики страниц, проверка орфографии, интеграция с *SharePoint Server*.

Hotdog – программа имеет простой и понятный пользователю интерфейс. Эта программа интегрируется с Flash, SQL, PHP, ASP, может работать с GIF-изображениями (оптимизация, анимация), включает в себя HTML-компрессор, может создавать файлы-справки (CHM).

Текстовые редакторы

Homesite – самый популярный и мощный текстовый редактор. В нем достаточно легко работать с *HTML*-кодом, также есть поддержка XHTML, CSS-редактор.

HTML Pad поддерживает JavaScript, VBScript, SSI, ASP и Perl, умеет создавать макросы, включает в себя много различных справочных материалов по CSS и HTML и многое другое.

Notepad (Блокнот). В этой программе нет функций, которые облегчат создание сайта (подсветки кода, вставки готовых конструкция кода), но она имеется на компьютере у каждого пользователя в папке Стандартные.

Популярные конструкторы сайтов

Конструктор сайтов – это система из набора инструментов, которая позволяет создавать сайты онлайн и администрировать их без каких-либо специализированных знаний. С ее помощью можно выбрать тип будущего сайта (визитка, магазин и т. д.), готовый шаблон дизайна, цветовое оформление и модули, которые будут на нем отображаться.

Зачастую они предоставляют достаточно возможностей для создания сайтов, которые по качеству могут превзойти продукты небольших web-студий, выполняющих заказы для малого и среднего бизнеса.

В сети существует множество сервисов, помогающих создать web-приложение как новичку, так и человеку далекому от программирования, верстки сайтов и графических редакторов.

uCoz – это бесплатный конструктор сайтов и хостинг для сайтов, созданных с ее использованием.

Модули *uCoz* могут использоваться как в единой связке для создания полнофункционального сайта, так и по отдельности, например, в качестве блог-платформы, *web*-форума и др. На данный момент в системе создано более миллиона сайтов.

Модули – *web*-приложения, скомбинировав которые можно получить сайт любой сложности: от простой домашней странички с гостевой книгой до большого портала.

Taba.ru – онлайн-конструктор сайтов позволяет быстро создать многофункциональный сайт, предоставляя доменное имя 2-го или 3-го уровня и хостинг от 1 Гб.

Конструктор сайтов обладает удобным и понятным интерфейсом. Готовые функциональные модули добавляются, перемещаются и настраиваются одним кликом мыши. Он позволяет посмотреть данные о новых пользователях, принять или отклонить их регистрацию.

Nethouse – конструктор сайтов, который позволяет пользователю создать качественный и функциональный сайт для бизнеса за короткое время. Чтобы получить готовый сайт, достаточно зарегистрироваться и заполнить всю необходимую информацию – Контакты, Услуги и цены, О себе. Nethouse предлагает пользователю выбрать один из множества красивых шаблонов и дизайнов, а также по желанию изменить фон.

APS – стандарт пакетов автоматической установки для модели Software-as-a-Service (SaaS).

APS-формат разработан, чтобы упростить реализацию бизнес-модели SaaS для облачных сервис-провайдеров и разработчиков ПО. APS-пакет – упакованное в соответствии с APS-стандартом *web*-приложение.

На данный момент APS-стандарт поддерживается хостинг-панелями Parallels: Plesk, Small Business Panel, POA и SysCP.

Всего в APS-каталоге опубликованы более 200 наименований CMS и различных web-приложений (как коммерческих, так и open source).

Wix – имеет огромное количество качественных и разнообразных шаблонов (более 1000). Наряду с HTML5 есть возможность создания сайтов во Flash и поддерживается функции Drag&Drop. Сайты на Wix хорошо индексируются поисковыми системами.

Большинство конструкторов сайтов бесплатны (расширение функционала потребует перехода на платный тариф), имеют более-менее удобный интерфейс, неплохой набор шаблонов, акции и т. д. Конкуренция велика, поэтому каждый ресурс стремится привлечь как можно большее количество клиентов всеми доступными средствами. Лучшие конструкторы сайтов дают возможность на бесплатном пакете сделать свой сайт и продвигать его в Интернете.

3.5. Языки разметки гипертекста HTML и XML

Любой Интернет-сайт состоит из одного или нескольких связанных между собой HTML-документов и сопутствующих файлов (например, файлы картинок). HTML (Hypertext Markup Language) – язык разметки гипертекста.

XML (eXtensible Markup Language) используется для разметки стандартных документов во многом так же, как HTML. Однако XML ориентирован на работу со структурированными данными, такими как результаты запроса, метаинформация об узле *web* или элементы и типы схем документов.

Все файлы, используемые сайтом, должны иметь имена, состоящие только из латинских букв, цифр и знаков подчеркивания (_).

Первым файлом сайта, который автоматически загружается, является файл с именем **index** или **default** (лучше всего использовать **index**).

Каждый HTML-документ составляется в соответствии с определенными правилами и должен иметь расширение *.htm или *.html.

Программа, которая расшифровывает HTML-документ, называется браузером.

Любой браузер реализует следующие функции: осуществляет алгоритм обмена файлами с сайта, расшифровывает и отображает содержимое HTML-документа, обеспечивает выполнение программ на языках скриптов JavaScript или VBScript. Для реализации и исполнения всех этих функций не нужны никакие другие программные продукты.

Алгоритм работы браузера при обслуживании сайта заключается в следующем. Вы запускаете сайт по сети или головной файл сайта из локальной папки компьютера. Браузер при этом получает полный URL-адрес сайта, т. е. попросту папку, где искать файлы сайта, и отображает содержимое головного файла. Если в процессе работы с файлом требуется новый файл картинки или HTML-документа, то браузер делает запрос на сервер, откуда и получает требуемый файл. С этим файлом он делает все, что предусмотрено сценарием.

Сеть задействуется только в момент получения браузером нужного файла. Если по сценарию происходит частое переключение между HTML-файлами, то это замедляет работу сайта и задействует ресурсы сети (так называемый трафик).

Для устранения этого недостатка используется динамический HTML (DHTML), реализуемый программами на языках скриптов. Программы скриптов могут изменять сценарий, модифицировать окно браузера и все это делается в пределах того HTML-документа, с которым в данный момент работает браузер.

Единственное, чего не может браузер даже с использование скриптов –сохранять введенную в процессе сценария информацию.

Структура HTML-документа, основные элементы

HTML-документ оформляется по определенным правилам и всегда содержит в себе элементы языка HTML. Элементы могут быть парными (контейнеры) и непарными (дескрипторы).

Контейнер имеет следующую структуру:

<имя-элемента>любой текст</имя-элемента>

Как видно, контейнер открывается по <имя-элемента > и закрывается по </имя-элемента>. Действие контейнера распространяется на весь текст, который этот контейнер окаймляет (в данном примере это любой текст).

Существуют парные элементы, которые допускается не закрывать, так как они автоматически закрываются по контексту HTML-документа.

Дескриптор имеет следующую структуру:

<имя-элемента>

Как видно, дескриптор не закрывается по </имя-элемента>, а смысл его использования не связан с сопутствующим текстом документа, а заложен в названии самого элемента.

При открытии любого элемента по <имя-элемента> можно указывать допустимые атрибуты этого элемента. Атрибуты описывают свойства элемента, записываются в любой последовательности и отделяются друг от друга пробелом. Допустимые значения атрибутов указываются через знак равно (=) за именем атрибута и должны заключаться в кавычки («значение») за исключением зарезервированных слов этого атрибута (не будет ошибки, если и зарезервированное слово будет заключено в кавычки). **Значения** атрибута необходимо набирать с **соблюдением больших и малых букв**. Перечень допустимых атрибутов всегда приводится вместе с описанием конкретного элемента. Примером использования парного элемента может служить, например, задание для фрагмента текста цвета и размера символов:

<Font color=red size=4>фрагмент текста</Font>

Текст HTML-документа набирается в любом текстовом редакторе, например Блокнот (но ни в коем случае не в редакторе Word). Имена элементов, их атрибуты и значения можно набирать большими или малыми буквами. Принято элементы набирать большими буквами.

Для правильного отображения в окне браузера HTML-документ набирается по определенным правилам.

Структура HTML-документа имеет следующий вид:

<HTML>
<HEAD>
<TITLE> название документа </TITLE>
<META...>
</HEAD>
<BODY>
Тело документа
</BODY>
</HTML>

Элементы <BODY> и <HTML> можно не закрывать.

Как видно HTML-документ состоит из двух частей: заголовок (HEAD) и тело (BODY). В каждой части документа используются свои элементы.

Браузером отображается только часть документа, заключенная между <Body> и </Body>

Элемент <TITLE> предназначен для указания названия документа и предназначен для лиц, изучающих ваш HTML-документ. Отображается элемент <TITLE>, как правило, в заголовке окна браузера.

Элементы <META> используются для указания автора файла и для указания ключевых слов, на которые можно ориентироваться при поиске информации в поисковых сайтах:

<META NAME = "Author" Content = "Глеб Окунев"> – указание автора документа;

<META NAME = "Keywords" Content = "информация, технология, система"> – указание ключевых слов для поиска поисковыми сайтами.

Лучше всего с использованием элемента META указывать также вид кодировки русских букв:

<META Charset=”windows-1251”>

Таким образом, в самом простейшем виде HTML-документ будет иметь следующий вид:

HTML>
<HEAD>
<TITLE> автобиография </TITLE>
< META Name = "Author" Content = " Глеб Окунев">
< META Name = "Keywords" Content = "автобиография">
<META Charset=”windows-1251”>
</HEAD>
<BODY>
Моя автобиография

Элемент <Body> может содержать следующие атрибуты:

Text=“значение” – цвет символов всего документа;

BgColor=“значение” – цвет фона документа.

BackGround=”имя файла картинки” – фоном документа будет картинка (в качестве фона можно использовать или цвет по атрибуту **BgColor** или картинку по атрибуту **BackGround**).

Отметим, что браузер воспринимает картинки со следующими расширениями графических файлов: *.bmp, *.gif, *.jpg, *.jpeg. Файл картинки должен быть помещен в папку, где находится файл данного HTML-документа. В ”имя файла картинки” необходимо указывать полное имя файла вместе с расширением, например, в виде:

BackGround=”aaa.gif”.

Значениями атрибутов **Text** и **BgColor** могут быть зарезервированные слова цветов английского языка (в этом случае значение не надо заключать в кавычки) или палитра цветов в виде: #C1D191 (в этом случае значение надо заключать в кавычки). Палитра цветов определяется в шестнадцатеричной системе исчисления, где первые два символа определяют красный цвет (00 – нет красного, FF – наибольший процент красного), вторые две цифры определяют зеленый цвет и третьи две цифры – синий цвет. Зарезервированными значениями цветов могут быть следующими:

aqua (цвет морской волны) #00FFFF
black (черный) #000000
blue (голубой) #0000FF
fuchsia (фуксия) #FF00FF
gray (серый) #808080
green (зеленый) #008000
lime (ярко зеленый) #00FF00
maroon (темно-бордовый) #800000
navy (темно-синий) #000080
purple (фиолетовый) #800080
red (красный) #FF0000
silver (серебряный) #C0C0C0
yellow (желтый) #FFFF00
white (белый) #FFFFFF

К каждому зарезервированному слову может быть без пробела добавлена приставка dark (темно) или light (светло), например, в виде darkBlue.

Пример задания атрибутов в элементе Body приведен ниже:

<Body BgColor=yellow Text=”#C0C0C0”>

Кроме задания палитры цветов в шестнадцатеричной системе исчисления можно указывать процентное содержимое красного, зеленого и синего:

<Body BgColor=yellow Text=”70%,10%,50%”>,

где для атрибута *Text* задан цвет с 70 % красного, 10 % зеленого и 50 % синего смеси цветов.

Если процентное содержимое всех цветов одинаковое, то это будет серый цвет (от черного, если процентное содержимое всех цветов равно нулю, до белого, если процентное содержимое всех цветов равно ста).

Основные элементы тела документа Body

В теле документа записывается отображаемая часть сайта. В принципе здесь можно записывать любой полезный текст, который будет отображаться в окне браузера. В отличие от редакторов оформительская часть текста (например, курсив или жирный текст) должна быть указана определенным элементом, поэтому нет смысла набирать в редакторе текст отличный от обычного его написания.

При наборе текста переносы в окне редактора не будут соответствовать переносам в окне браузера. Переносы в окне браузера выполняются после заполнения соответствующей строки окна браузера. Это же относится и к пробелам между словами – всегда между словами будет один пробел независимо от их количества в редакторе. Принудительный перенос на новую строку в браузере осуществляется элементом
, например в виде:

**первая строка
 вторая строка и т. д.**

В окне браузера этот текст буде выглядеть следующим образом:

первая строка первая строка
вторая строка вторая строка и т. д.

Как видим, перенос на новую строку в окне браузера выполняется по
 несмотря на то что, в редакторе этот текст записан в одной строке. Ниже приведены основные оформительские элементы HTML.

Элемент базового шрифта:

<BaseFont
Color=red
Face=”Times New Roman”
Size=3>

Данный элемент задает тип, размер и цвет шрифта, которые применяются по умолчанию для всего документа. Этот элемент необходимо размещать сразу же за элементом <BODY>.

В атрибуте **Color** указывается цвет символов одним из допустимых английских слов (в этом случае слово можно не заключать в кавычки) или в палитре красный-зеленый-синий (в этом случае значение надо заключать в кавычки) точно также, как это описано в соответствующем атрибуте элемента <BODY>.

В атрибуте **Face** указывается название допустимого шрифта с соблюдением принятого написания шрифта, т. е. с учетом прописных и заглавных букв в соответствующих местах.

В атрибуте **Size** указывается число от 1 до 7, где 1 – самый малый размер шрифта, 7 – самый большой размер шрифта.

По умолчанию полагается шрифт *Times New Roman* размером 3 черного цвета. Элемент фонового звука:

<BGSound
Balance=0
Loop=100
Volume=-500
SRC=”aaa.wav”>

Атрибут **SRC** указывает вместе с расширением имя звукового файла, предназначенного для воспроизведения. Форматами файла могут быть .wav, .au, .mid. Звуковой файл должен быть помещен в туже папку, где находиться файл данного документа.

Атрибут **Loop** указывает на то, сколько раз будет воспроизводиться звуковой файл. Если вместо числа указать ключевое слово *infinite*, то файл будет воспроизводиться бесконечно.

Атрибут **Balance** может содержать число от –10 000 до 10 000 и балансирует звук между динамиками (значение 0 – одинаковое звучание в обоих динамиках).

Атрибут **Volume** может принимать значение от –10 000 до 0 и определяет громкость звука.

Элемент комментария:

<!- комментарий ->

Этот элемент служит только для комментирования текста HTML-документа и его содержимое в окне браузера не отображается.

Элемент заголовков:

<H1 Align = Right> текст </H1>

Center – по центру

Left – слева

Justify – по ширине окна

Текст, помещенный между <H1> и </H1>, будет являться заголовком.

Вместо **H1** можно указывать H2, H3, H4, H5, H6. Элемент **H1** отображает заголовок самым большим шрифтом, элемент **H6** – соответственно самым малым шрифтом. Атрибут **Align** используется для выравнивания заголовка относительно окна браузера. Значение **Right** выравнивает заголовок по правой границе окна браузера, **Center** – по центру, **Left** – по левой границе окна браузера.

Элемент центрирования текста:

<Center>текст</Center>

Элемент вставки горизонтальной линии:

<HR Align = Center Width = "50%" Size = 2 Color=Red>

Атрибут **Align** определяет выравнивание линии.

Атрибут **Width** определяет длину горизонтальной линии в пикселах или в процентах относительно ширины экрана.

Атрибут **Size** определяет толщину горизонтальной линии в пикселах.

Атрибут **Color** определяет цвет линии (см. описание значений цветов для элемента **Body**).

Шрифтовые элементы:

Действие любого из приведенных ниже элементов распространяется на текст, который этот элемент ограничивает.

<I>текст</I> – курсив;

<B>текст</B> – выделенный текст (полужирное начертание);

<Strong>текст</Strong> – сильное выделение текста (жирное начертание);

<U>текст</U> – подчеркивание;

<Strike>текст</Strike> – зачеркнутый текст;

<TT>текст</TT> – моноширинный шрифт (как на пишущей машинке);

<Big>текст</Big> – большой шрифт. Этот элемент увеличивает размер текста на определенную величину. Можно использовать вложенные друг в друга элементы <Big> для многократного увеличения размера шрифта;

<Small>текст</Small> – малый шрифт. Этот элемент уменьшает размер шрифта на определенную величину. Можно использовать вложенные друг в друга элементы <Small> для многократного уменьшения размера шрифта. В ниже приведенном примере:

<Big><Small> текст</Small></Big>

текст будет печататься стандартным размером шрифта, так как <Big> увеличивает размер шрифта, а элемент <Small> уменьшает его на эту же величину.

_{текст} – подстрочный текст (нижний индекс);

^{текст} – надстрочный текст (верхний индекс);

<Font Color=red Size=4>текст</Font> – используется для указания характеристик шрифта. Чаще всего используются два атрибута элемента <Font>: атрибут задания цвета символов **Color** (значения этого атрибута такие же, как и для элемента <Body>) и атрибут

указания размера шрифта **Size** (значением этого атрибута должно быть число от 1 для самого малого размера шрифта до 7). Вместо элемента <Font> рекомендуется использовать листы стилей.

Блочные элементы:

** текст ** – новый абзац;

<BlockQuote> текст </BlockQuote> – текст отображается с отступом от края листа, текст начинается с нового абзаца;

<PRE>текст</PRE> – определяет предварительно отформатированный текст. Переносы на новую строку и количество пробелов между словами будут такими же, как и в редакторе БЛОКНОТ;

** текст ** – блок текста. Практически всегда используется совместно с листами стилей.

текст – выделенный фрагмент текста. Очень часто используется совместно с листами стилей для придания фрагменту текста заданных свойств.

<acronym Title=”Республика Беларусь”>РБ</acronym> – аббревиатура и ее расшифровка. Расшифровка определяется атрибутом **Title** и появляется при установке курсора мыши на аббревиатуру.

<marquee behavior=”alternate” scrolldelay=100 width=500 height=30> текст </marquee> – движение текста (справа-налево до упора в левую границу и затем, наоборот) в блоке размером 500x30 со скоростью, определенной временем задержки 100 миллисекунд; если опустить атрибут **behavior**, то движение текста будет только справа-налево.

<cite Title=”Источник”>цитата</cite> – указывает, что текст, помещенный в него, является цитатой из книги или другого источника. Текст выводится при этом курсивом. С использованием листов стилей тексту можно придать нужное раскрашивание. Атрибутом **Title** можно указывать источник, откуда взята цитата; эта информация будет появляться при установке мыши на цитату.

Создание списков и вставка графики

Ненумерованный список определяется элементом <UL>, внутри которого может находиться элемент заголовка списка <LH> и обязательно один или несколько элементов <LI> непосредственно самих элементов списка. Фрагмент документа для формирования ненумерованного списка имеет следующий вид:

<UL>

<LH> заголовок списка</LH>

<LI> первый элемент списка
<LI> второй элемент списка
<LI> третий элемент списка
</UL>

В элементе <UL> можно указывать тип маркера списка следующим образом:

<UL Type = Disk> – заполненный кружок

Square – заполненный квадрат

Circle – незаполненный кружок

Если тип маркера указан в явном виде, то именно он и отображается. Если тип маркера не указан, то браузером автоматически происходит его изменение для вложенных списков.

В качестве маркера может быть использована любая картинка, подключение которой выполняется с использованием листов стилей, что будет рассмотрено в соответствующем разделе.

Пример фрагмента формирования ненумерованного списка приведен ниже:

<UL>
<LH>список клиентов:
<LI>Иванов
<LI>Петров
<LI>Сидоров
</UL>

Организация нумерованного списка

Нумерованный список формируется с использованием элемента <OL>, внутри которого может находиться элемент формирования заголовка списка <LH> и обязательно один или несколько элементов <LI> формирования непосредственно самих элементов списка. Фрагмент документа для формирования нумерованного списка имеет следующий вид:

<OL>
<LH> заголовок списка </LH>
<LI> первый элемент списка
<LI> второй элемент списка
<LI> третий элемент списка
</OL>

По умолчанию нумерация осуществляется арабскими цифрами, начиная с 1. Это умолчание можно изменить указанием атрибута **Type** в виде:

<OL Type=a>

Атрибут **Type** может принимать следующие значения:

1 – арабские цифры;

a – строчные буквы латинского алфавита;

A – прописные буквы латинского алфавита;

I – прописные римские цифры;

i – строчные римские цифры

Кроме этого нумерацию можно изменить в элементах <LI>:

<LI VALUE = 3> – дальнейшая нумерация будет с 3-го символа.

Пример формирования вложенного списка с использованием нумерованного и ненумерованного списков приведен ниже:

<OL>
<LH>список магистрантов:
<LI>БГАТУ
<UL>
<LI>агромеханический факультет
<LI>агроэнергетический факультет
<LI>факультет предпринимательства и управления
</UL>
<LI>БГЭУ
<UL>
<LI>учетно-экономический факультет
<LI>факультет международных экономических отношений
</UL>
<LI>БГУ
</OL>

Список определений

Элемент <DL> списка определений в отличие от нумерованных и ненумерованных списков состоит из двух частей: первая служит для задания термина, вторая – для вывода определения этого термина. В одном <DL> может находиться определение нескольких терминов:

<DL>
<LH> заголовок списка </LH>
<DT> термин 1
<DD> определение термина 1
<DT> термин 2
<DD> определение термина 2
</DL>

Текст после элемента <DD> будет отображаться с отступом от левой границы окна браузера.

Внутри <DD> могут находиться нумерованные и ненумерованные списки.

Пример списка определений приведен ниже:

<DL>
<LH> заголовок списка </LH>
<DT> термин 1
**<DD> определение термина 1
возможно из нескольких строк**
<DT> термин 2
<DD>
**определение термина 2
вторая строка определения
 третья строка**
</DL>

Вставка графики

Вставка графики в документ HTML производится с помощью элемента <IMG>:

<Img src="имя графического файла" Align= Height= Width= Border= Hspace= Vspace= Alt= >

src – определяет имя графического файла вместе с его расширением, сам графический файл должен находиться в той же папке, что и HTML-документ.

Align – задает расположение изображения относительно текста документа: *Left* – изображение располагается слева от текста; *Right* – изображение располагается справа от текста; *Top* – вверх изображения по строке текста; *Bottom* – (умолчание) низ изображения по строке текста;

Middle – середина изображения по строке текста.

Height – высота изображения в пикселах. Если это свойство не задано, то высота изображения определяется самим файлом;

Width – ширина изображения в пикселах. Если это свойство не задано, то ширина изображения определяется самим файлом;

Border – ширина рамки в пикселах вокруг изображения (0 – нет рамки);

Hspace – ширина незаполненного пространства (расстояние от границы изображения до текста) в пикселах слева и справа от изображения;

Vspace – ширина незаполненного пространства в пикселах снизу и сверху от изображения.

Alt – всплывающая подсказка появляющаяся при установке курсора мыши на картинку. Чаще всего используется, если картинка является гиперссылкой.

DynSrc – тоже, что и **Src**, только используется для анимационных видеофайлов *.gif или *.avi.

Loop – используется совместно с **DynSrc** и задает количество повторов видеоролика либо ключевое слово *Infinite* при бесконечном повторе.

Вставка таблицы

Таблицы являются одними из самых используемых при формировании HTML-документа элементов. Часто таблицы используются при формировании меню и еще более часто для рационального и компактного размещения информации на странице.

Пример формирования таблиц:

<HTML>
<Head>
</Head>
<Body>
<H1> Простейшая таблица</H1>
<Table Border=1> <!–это начало таблицы →
<Caption Align=Center>У таблицы может быть заголовок</Caption>
<TR> <!– это начало первой строки →
<TD><!– это начало первой ячейки (столбца) →
Первая строка, первая колонка
</TD><!– это конец первой ячейки (столбца) →
<TD><!– это начало второй ячейки (столбца) →
Первая строка, вторая колонка
</TD><!– это конец второй ячейки (столбца) →
</TR><!– это конец первой строки →
<TR> <!– это начало второй строки →
<TD><!– это начало первой ячейки (столбца) →
Вторая строка, первая колонка
</TD><!– это конец первой ячейки (столбца) →
<TD><!– это начало второй ячейки (столбца) →
Вторая строка, вторая колонка

</TD><!– это конец второй ячейки (столбца) →
</TR><!– это конец первой строки →
</Table> <!– это конец таблицы →
</Body>
</HTML>

Если по логике ячейка должна быть пустой, то необходимо использовать неразрывный пробел : **<TD> </TD>**

Элемент **<Table>** используется для указания начала и конца таблицы и для указания общих свойств для всей таблицы.

Элемент <Table> может содержать следующие атрибуты:

Width – ширина таблицы в пикселях (Width=200) или в процентах от ширины страницы (Width=50%).

Align – устанавливает расположение таблицы по отношению к левой и правой границам документа. Допустимые значения:

Left – выравнивание по левой границе;

Center – выравнивание по центру;

Right – выравнивание по правой границе.

Border – устанавливает ширину рамки вокруг таблицы в пикселях (умолчание 0).

BorderColor – цвет рамки таблицы.

Cellspacing – устанавливает расстояние между рамками ячеек таблицы в пикселях (умолчание 2).

Cellpadding – устанавливает расстояние от границы ячейки до информации в ней (умолчание 1).

Background – фоном всей таблицы будет картинка.

BgColor – цвет фона таблицы.

Frame – указывает какие внешние стороны таблицы должны отображаться с рамкой. Атрибут может принимать следующие значения:

Above – только верхний край таблицы;

Below – только нижний край таблицы;

LHS – только левый край таблицы;

RHS – только правый край таблицы;

HSides – верхний и нижний края таблицы;

VSides – левый и правый края таблицы;

Box – все рамки таблицы; присваивается по умолчанию, если значение атрибута **Border** больше нуля;

Void – отключает вывод всех внешних рамок.

Rules – управляет выводом внутренних рамок таблицы. Атрибут может принимать следующие значения:

All – выводятся все рамки ячеек;

Cols – выводятся только разделительные линии столбцов;

Groups – выводятся только разделительные линии между группами ячеек, заданных с помощью элементов <THead>, <TBody>, <TFoot> и <ColGroup>;

Rows – отображаются линии только для строк;

None – отключает вывод всех внутренних линий.

Элемент <TR> определяет начало и конец строки таблицы и общие свойства для ячеек строки таблицы.

Элемент <TR> может содержать следующие атрибуты:

Align – устанавливает выравнивание текста в ячейках по горизонтали (значения см. выше для Align таблицы).

Valign – устанавливает выравнивание текста в ячейках по вертикали. Может принимать следующие значения:

Top – выравнивание по верхнему краю;

Middle – выравнивание по центру;

Bottom – выравнивание по нижнему краю.

Bgcolor – цвет фона ячеек строки.

BorderColor – цвет рамок ячеек таблицы.

Вместо элемента <TR> можно использовать элемент <THEAD> для строки, являющейся верхним колонтитулом (шапка) таблицы, и элемент <TFOOT> для строки, являющейся нижним колонтитулом (подведение итогов) таблицы. Пример использования элемента <THEAD> приведен ниже:

```
<Table border=1 BgColor=red >
<THead Align–Center BGColor=Yellow>
<TD>Столбец1</TD><TD>Столбец2</TD>
</THead>
<TR>
<TD>Текст первого столбца</TD>
<TD>Текст второго столбца</TD>
</TR>
</Table>
```

Элемент <TD> используется для указания ячейки в строке и ее индивидуальных свойств.

Элемент <TD> может содержать следующие атрибуты:

Align – устанавливает выравнивание текста в ячейке по горизонтали. Может принимать значения *Left*, *Right*, *Center* (см. выше для **Align** таблицы). Кроме этих значений может быть значение *Justify* – выравнивание по ширине ячейки, и значение *Char* – выравнивание по указанному символу (это значение пока не поддерживается в Интернет Exproler). Если используется *Char*, то необходимо указать атрибуты **Char** и **CharSet**.

CharSet – указывает смещение при выравнивании по символу (см. зние *Char* атрибута **Align**: <TD Border=2 Align= Char Char="." CharSet=2>).

Valign – устанавливает выравнивание текста в ячейке по вертикали. Значения см. выше.

Colspan – устанавливает размах ячейки по горизонтали. Например Colspan=2 означает, что ячейка простирается на две колонки.

Rowspan – устанавливает размах ячейки по вертикали. Например Rowspan=2 означает, что ячейка простирается на две строки.

Width – ширина ячейки в пикселях или в процентах от ширины таблицы.

Height – высота ячейки в пикселях или в процентах от высоты таблицы.

Bgcolor – цвет фона ячейки.

Background – фоном ячейки будет картинка.

BorderColor – цвет рамок ячейки.

Вместо элемента <TD> можно использовать элемент <TH>. Этот элемент указывает на то, что данная ячейка является "шапкой" таблицы. Браузером элемент <TH> отображается несколько другим стилем, чем элемент <TD>. Элемент <TH> содержит тот же набор атрибутов и свойств, что и элемент <TD>.

В том случае, когда группа ячеек таблицы за исключением нижнего и верхнего колонтитулов должна иметь одинаковые атрибуты, они группируются элементом <TBODY>:

```
<Table>
<TBody BgColor=red Align=Center>
<TR>
<TH>ЭВМ</TH><TH>Быстродействие</TH>
</TR>
<TR>
<TH>Crey</TH><TH>1000000</TH>
<TH>IBM 9000</TH><TH>10000</TH>
</TR>
</TBody>
</Table>
```


3.6. Скриптовые языки программирования

Скриптовый язык (англ. scripting language, также называют язык сценариев) – язык программирования, разработанный для записи «сценариев», последовательностей операций, которые пользователь может выполнять на компьютере. Простые скриптовые языки раньше часто называли языками пакетной обработки (*batch languages* или *job control languages*). Сценарии всегда интерпретируются, а не компилируются.

Компиляция – преобразование программы, представленной на одном из языков программирования, в коды на машинно-ориентированном языке, которые принимаются и исполняются непосредственно процессором. Результатом компиляции является объектный файл с необходимыми внешними ссылками для компоновщика. Программа уже переведена в машинные инструкции, однако еще не полностью готова к выполнению. Интерпретация – процесс непосредственного покомандного выполнения программы без предварительной компиляции.

Сценарий (скрипт) – это программа, которая автоматизирует некоторую задачу, которую без сценария пользователь делал бы вручную, используя интерфейс программы.

Поскольку сценарии интерпретируются из исходного кода динамически при каждом исполнении, они выполняются обычно значительно медленнее готовых программ, оттранслированных в машинный код на этапе компиляции. Поэтому сценарные языки не применяются для написания программ, требующих оптимальности и быстроты исполнения. Но из-за простоты они часто применяются для написания небольших, одноразовых («проблемных») программ. Также, в плане быстродействия скриптовые языки можно разделить на языки динамического разбора (*sh, command.com*) и предварительно компилируемые (*Perl*).

Языки динамического разбора считывают инструкции из файла программы минимально требующимися блоками и исполняют эти блоки не читая дальнейший код.

Предкомпилируемые языки вначале считывают всю программу, компилируют ее всю либо в машинный код, либо в какой-то внутренний формат и затем исполняют получившийся код.

Для создания пользовательских расширений язык сценариев удобен в ряде случаев:

– безопасность. Скриптовый язык обеспечивает программируемость без риска дестабилизации системы. Так как скрипты не компилируются,

а интерпретируются, то неправильно написанная программа выведет диагностическое сообщение, не вызывая падение системы;

– наглядность. Язык сценариев используется, если необходим выразительный код. Концепция программирования в скриптовом языке может кардинально отличаться от основной программы;

– простота. Код имеет собственный набор программ, поэтому одна строка может выполнять те же операции, что и десятки строк на обычном языке. Поэтому для написания кодов не требуется программист высокой квалификации;

– кроссбраузерность. Скриптовые языки ориентированы на кроссбраузерность. Например, JavaScript может исполняться браузерами практически под всеми современными операционными системами.

Выделяют следующие типы скриптовых языков:

– ***универсальные:*** Forth, AngelScript, Perl, PHP, Python, Tcl (Tool command language), Squirrel, REBOL, Ruby, AutoIt, Lua;

– ***встроенные в прикладные программы:*** VBA, UnrealScript, AutoLISP, Emacs Lisp, Game Maker Language, MQL4 script, ERM;

– ***командные оболочки:*** sh, AppleScript, bash, csh, ksh, JCL, cmd.exe, command.com, REXX, Visual Basic Script;

– ***встраиваемые:*** Guile, Script.NET, ActionScript, Lingo (используется в редакторе Director), Sleep, браузерные Jscript и JavaScript.

Некоторые приложения имеют встроенную возможность расширения сценариями, написанными на любом универсальном скриптовом языке, например, автоматический планировщик задач или библиотека SWIG.

К скриптам также относят многие консольные утилиты, которые поддерживают выполнение записанной в файл последовательности команд.

Контрольные вопросы

1. Как классифицируются компьютерные сети?
2. Семиуровневая модель структуры протоколов связи это?
3. Какие существуют основные протоколы сети Интернет?
4. Как происходит адресация в сети Интернет?
5. Какие существуют распространенные сервисы сети Интернет?
6. Какие области охватывает разработчик Web-дизайна сайта?
7. Какие инструменты существуют для разработки Web-сайта?
8. Для чего предназначены языки HTML и XML?
9. Что содержит структура HTML-документа
10. Для чего предназначены скриптовые языки программирования?

4. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ (СУБД)

4.1. Классификация систем управления базами данных

Система управления базами данных (СУБД) – это *программно-языковый комплекс*, предназначенный для создания в ЭВМ общей для многих пользователей БД и управления хранящимися в ней данными. Под управлением БД понимается поддержание ее в актуальном состоянии, что достигается путем своевременного изменения и восстановления хранящихся в БД, данных при нестандартных ситуациях, защиты данных от несанкционированного вмешательства и выполнения многих других функций, обеспечивающих адекватное отображение БД некоторой предметной области. СУБД обеспечивают эффективный доступ пользователей к содержащимся в них данным в рамках полномочий, предоставленных пользователям.

По степени универсальности различают два класса СУБД:

1. Системы общего назначения, которые не ориентированы на конкретную предметную область или на информационные потребности определенной группы пользователей.

2. Специализированные системы, функционирующие на некоторой модели ЭВМ в определенной операционной системе, имеющие средства настройки на работу с БД в конкретной предметной области.

По выполняемым функциям СУБД делят на информационные и операционные. Информационные позволяют организовать хранение информации и доступ к ней для ознакомления и выдачи простых справок. Операционные выполняют сложную обработку данных.

По языкам общения СУБД могут быть открытые, замкнутые и смешанные. В открытых системах для обращения к БД используются универсальные языки программирования. Замкнутые системы имеют собственные языки общения с пользователями.

По мощности выделяют настольные и корпоративные СУБД. Настольные системы (Access, FoxPro, Paradox) ориентированы на конечных пользователей (специалистов в конкретной предметной области). Они не предъявляют высоких требований к техническим средствам, отличаются низкой стоимостью. Корпоративные СУБД (Oracle, SyBase, DB2) обеспечивают работу в распределенной среде, имеют высокую производительность,

развитые средства администрирования и более широкие возможности поддержания целостности. СУБД MS SQL Server, Interbase имеют возможности и настольных и корпоративных систем.

По реализуемой модели данных СУБД получили названия в соответствии со схемой данных, которую они поддерживают: иерархические, сетевые, реляционные, объектно-ориентированные.

4.2. Модели данных

Модель данных представляет собой образ БД, отражающий множество структур данных и связей между ними.

Известны следующие модели данных: *иерархические, сетевые, реляционные, постреляционные, объектно-ориентированные, объектно-реляционные, многомерные.*

Иерархическая модель данных

Иерархическая модель данных представляет собой многоуровневую древовидную структуру БД. Пример схемы представлен на рис. 4.1.

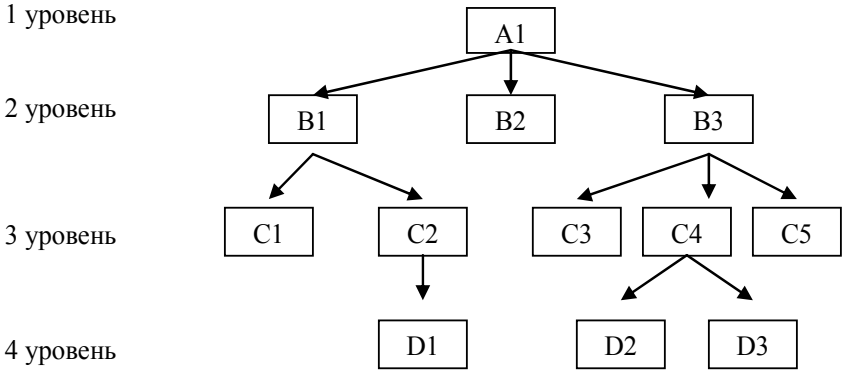


Рис. 4.1. Древовидная структура данных

Основной компонентой в древовидной структуре данных является узел. Любой узел может отображать массивы, логические записи, сегменты и поля в зависимости от уровня древовидной структуры. На самом верхнем уровне иерархии расположен только один

узел, называемый *корнем* дерева. Корень дерева имеет связи только с узлами нижестоящего уровня. Любой другой узел дерева связан только с одним узлом более высокого уровня и с одним или несколькими узлами более низкого уровня иерархии.

Узлы, связанные с узлами более высокого уровня называются *порожденными*. К каждому узлу дерева существует только один путь от корневого узла. Например, для узла D2 таким путем будет A1, B3, C4.

К достоинствам иерархической модели относятся: логичность построения, большая информационная емкость, легкость формализованного описания.

Основным недостатком модели является слабая гибкость структуры из-за установленного порядка следования узлов. Кроме того, модель плохо приспособлена к информационному поиску данных по любому произвольному сочетанию признаков их группировки.

Сетевая модель данных

Сетевая модель данных допускает не иерархические отношения. В сетевой структуре порожденный узел может быть связан более, чем с одним исходным узлом.

В сетевой структуре данных между порожденными и исходными узлами могут существовать простые и сложные связи.

Примером простой сетевой структуры является взаимосвязь между сегментами «Профессия», «Цех», «Личность» (рис. 4.2.). Слева на рисунке изображены уровни структуры. Справа – экземпляр сетевой структуры.

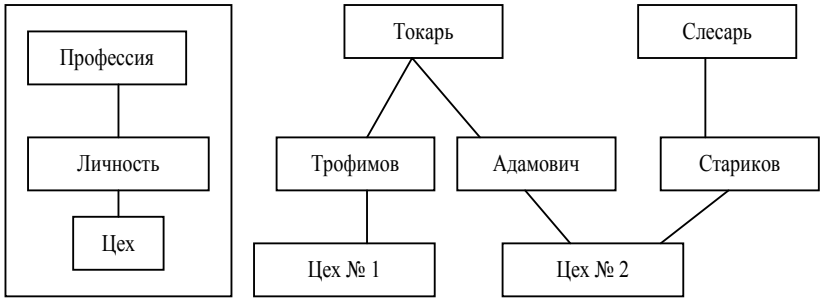


Рис. 4.2. Простая сетевая структура данных

Из рис. 4.2. видно, что порожденный узел «Цех № 2» связан с двумя исходными узлами. Пример описывает случай, когда каждая личность работает только в одном цехе и имеет только одну профессию, в тоже время одна и та же профессия может соответствовать многим работающим одного и того же цеха.

Примером сложной сетевой структуры может служить модель, отражающая отношения между предприятиями, заказчиками, изделиями и материалами. Предположим, что несколько предприятий поставляют заказчикам некоторые изделия, изготовленные из разного вида материалов. При условии, что любое предприятие может поставлять разные изделия любому заказчику и каждое изделие может быть изготовлено из разного материала, приходим к трехуровневой сложной сетевой структуре, изображенной на рис. 4.3.

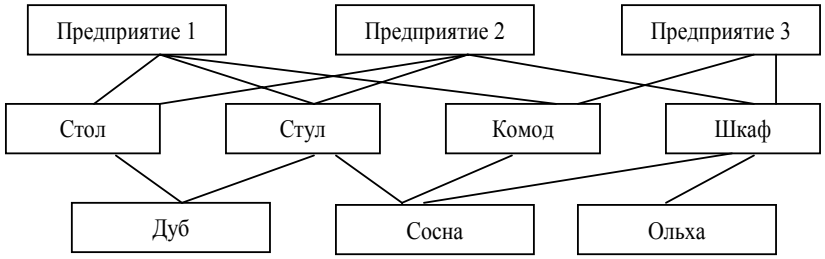


Рис. 4.3. Сложная сетевая структура данных

Реляционная модель данных

Реляционные модели данных получили свое название от английского слова relation, которое переводится как отношение (соотношение, связь). Отношением в реляционных БД называют таблицы с данными. Табличное представление данных является принципиальной особенностью реляционных моделей.

Реляционная таблица отражает некоторый объект, процесс или явление, то есть некоторую сущность. Она состоит из строк и столбцов и имеет имя, уникальное внутри БД, например, СОТРУДНИКИ. Столбец в таблице представляет собой поле, а строка – это логическая запись. Записи с конкретными значениями полей являются экземплярами сущности. Реляционная таблица обладает следующими свойствами:

– каждый элемент таблицы – один элемент данных;

– все столбцы в таблице однородные, т. е. все элементы в столбце имеют одинаковый тип (числовой, текстовый и т. д.) и длину;

– каждый столбец имеет уникальное имя – это имя поля;

– одинаковые строки в таблице отсутствуют;

– порядок следования строк и столбцов может быть произвольным.

Табл. 4.1 иллюстрирует пример реляционной таблицы.

Таблица 4.1

Реляционная таблица СОТРУДНИКИ

Табельный номер	Фамилия	Имя	Отчество	Дата рождения	Код специальности	Код цеха
001	Авдеев	Сергей	Егорович	12.03.1965	11	1
002	Бородин	Олег	Иванович	17.09.1980	05	2
003	Король	Андрей	Матвеевич	30.11.1975	01	2
004	Медведев	Яков	Петрович	06.08.1961	05	1
005	Семашко	Алла	Федоровна	04.07.1970	02	1

Структурные компоненты таблицы СОТРУДНИКИ:

– Названия столбцов – Табельный номер, Фамилия, Имя, Отчество, Дата рождения, Код специальности, Код цеха – являются **именами полей**.

– Каждый столбец – например, Табельный номер: 001, 002, 003, 004, 005– представляет отдельное **поле**.

– Множество допустимых значений конкретного поля называется **доменом**.

– Любая строка – это **логическая запись** (в дальнейшем – запись).

– Конкретная строка – 001, Ермольчик, Сергей, Викторович, 12.03.1965, 11, 1 – это **экземпляр записи**.

Реляционные таблицы в БД связаны между собой с помощью ключевых полей. ***Ключевое поле*** – это столбец, который однозначно идентифицирует каждую строку таблицы, то есть значения элементов данных в этом столбце не повторяются, они уникальны. Ключевое поле называют также ***первичным ключом***, если речь идет о главной таблице из двух связанных между собой таблиц. Ключ (ключевое поле) является ***простым***, если записи однозначно определяются значениями одного столбца. Если ключ содержит несколько столбцов (полей), то он называется ***составным ключом***.

В приведенной выше таблице 4.1 простым первичным ключом может быть поле «Табельный номер», составным ключом – совокупность полей «Фамилия», «Имя», «Отчество».

Связи поддерживаются внешними ключами. ***Внешний ключ*** – это ключ второй таблицы. Он однозначно характеризует значения первичного ключа первой таблицы. Чтобы связать две реляционные таблицы, необходимо ввести в структуру первой таблицы внешний ключ. Для понимания внешнего ключа возьмем вторую табл. 4.2.

Таблица 4.2

Реляционная таблица ЦЕХ

Код цеха	Наименование цеха	Вид	Начальник цеха
1	Литейный	Основной	Кривцов А. А.
2	Механический	Основной	Заворотнюк Е. С.
3	Ремонтный	Вспомогательный	Воложин Ф. И.

В таблице ЦЕХ в качестве первичного ключа выбрано поле «Код цеха». Это поле при связи таблиц СОТРУДНИКИ и ЦЕХ выступает в роли внешнего ключа таблицы СОТРУДНИКИ.

Инструментом для удобной работы с таблицами служит реляционная алгебра. Для использования математического аппарата реляционной алгебры все таблицы должны быть преобразованы в отношения. Таблица является отношением, если она удовлетворяет требованию уникальности первичного ключа.

В соответствии с терминологией реляционной алгебры отношение рассматривается как *множество*, строки таблицы называются *кортежами*, столбцы – *атрибутами*.

Отношения, являющиеся исходными для формирования нового отношения, называются *отношениями-операндами*.

Над отношениями выполняются традиционные операции теории множеств: *объединение, пересечение, выборка, проекция, разность, соединение, произведение, деление.*

Объединение отношений – это создание нового отношения, содержащего все кортежи отношений-операндов. При этом операнды должны иметь одинаковые атрибуты.

Пересечение отношений – создание нового отношения, содержащего кортежи, общие для сравниваемых отношений-операндов. При этом операнды должны иметь одинаковые атрибуты.

Ограничение отношения (выборка) – создание нового отношения отбором в него кортежей отношения-операнда, которые удовлетворяют условию ограничения.

Проекция отношения – создание нового отношения отбором в него определенных атрибутов отношения-операнда.

Разность отношений – создание нового отношения, содержащего те кортежи первого отношения-операнда, которые отсутствуют во втором отношении-операнде.

Соединение отношений – создание нового отношения, кортеж которого является результатом сцепления кортежей отношений-операндов. Условием для соединения отношений является наличие в них общего атрибута.

Произведение отношений – создание нового отношения, в котором имеются все атрибуты отношений-операндов, а кортежи нового отношения получены путем попарного сцепления кортежей отношений операндов.

Деление отношений – создание нового отношения, содержащего атрибуты первого отношения-операнда, отсутствующие во втором отношении-операнде, и кортежи первого отношения-операнда, которые совпали с кортежами второго отношения-операнда.

Объектно-ориентированная модель

Объектно-ориентированные модели позволяют организовать хранение данных, не раскладывая их по таблицам.

Основными понятиями объектно-ориентированных моделей являются *класс, объект, поле, свойство и метод.*

Классом (объектным типом) называется особый тип записи, который может иметь в своем составе поля, методы и свойства. *Пример класса – предприятия города.*

Объект (экземпляр класса) – это переменная объектного типа.

Поля объекта аналогичны полям записи (в реляционной таблице). Это данные, уникальные для каждого объекта. *Примеры полей: адрес предприятия, расчетный счет в банке.*

Методы – это процедуры и функции, описанные внутри класса и предназначенные для операций над полями. В отличие от полей, методы у двух объектов одного класса общие.

Свойство можно определить как специфическое поле, имеющее тип данных – класс (для сравнения – в реляционной таблице типы данных: текстовый, числовой и др.). Это значит, что такого рода поле само является объектом и принадлежит к определенному классу. *Пример свойства – продукция*.

В состав класса входит указатель на специальную таблицу, где содержится вся информация, нужная для вызова методов. От обычных процедур и функций методы отличаются тем, что им при вызове передается (неявно) указатель на тот объект, который их вызвал.

Постреляционная модель

Постреляционные модели обеспечивают хранение данных в виде таблиц, как и реляционные модели. Однако каждый элемент таблицы не обязательно представляет собой один элемент данных и имеет одинаковую длину. Данные могут иметь сложную структуру: содержать вложенные структуры, динамически изменяемые размеры, произвольные структуры, задаваемые пользователем (например, мультимедиа).

Пространственная модель

Пространственная модель (*dimensional model*) – это модель, в которой данные организованы не по третьей нормальной форме, а в виде тематических таблиц, каждая из которых содержит характеристику отдельных категорий информации (dimensions). Основная цель пространственной модели – минимизировать время выполнения запроса, поэтому допускается денормализация данных.

Объектно-реляционная модель

Объектно-реляционные модели представляют данные как объекты. Все данные по-прежнему хранятся в виде таблиц. Этот подход позволяет плавно перейти от технологии хранилища таблиц к технологии хранилища объектов. Основным недостатком объектно-реляционных моделей является необходимость при работе с базой данных разбирать объекты для размещения их в таблицах и собирать их для передачи пользователю из таблиц.

Многомерная модель

Многомерные модели предполагают хранение информации в форме логически упорядоченных многомерных массивов – *гиперкубов*. Основные понятия многомерной модели – измерение и значение (ячейка). *Измерение* – это множество, образующее одну из трех граней гиперкуба (аналог домена в реляционной модели). Измерения играют роль индексов, используемых для идентификации конкретных значений в ячейках гиперкуба. *Значения* – это подвергаемые анализу количественные или качественные данные, которые находятся в ячейках гиперкуба.

В многомерных системах используется обычно два варианта организации данных: *гиперкубическая и поликубическая*.

В ***гиперкубической*** организации данных все показатели определяются одним и тем же набором измерений. А при наличии нескольких гиперкубов все они имеют одинаковую размерность и совпадающие измерения. При ***поликубической*** организации данных может быть несколько гиперкубов с различной размерностью и с различными измерениями в качестве граней.

В многомерной модели вводятся следующие основные операции манипулирования измерениями: 1) сечение; 2) вращение; 3) детализация; 4) свертка.

При выполнении *операции сечения* формируется подмножество гиперкуба, в котором значение одного или более измерений фиксировано. Например, если зафиксировать значение измерения «Время» равным «IV квартал года», то мы получим двумерную таблицу с информацией о значениях всех параметров для всех субъектов хозяйствования в IV квартале года.

Операция вращения изменяет порядок представления измерений. Она обычно применяется к двумерным таблицам, обеспечивая представление их в более удобной для восприятия форме. Если в исходной таблице по горизонтали были расположены хозяйственные субъекты, а по вертикали параметры социально-экономической сферы, то после операции вращения параметры будут размещены по горизонтали, а названия субъектов – по вертикали.

Для выполнения *операций свертки и детализации* должна существовать иерархия значений измерения, то есть некоторая подчиненность одних значений другим. Например, 12 месяцев образуют год. При выполнении операции свертки одно из значений измерения

заменяется значением более высокого уровня иерархии. Операция детализация – это операция, обратная свертке. Она обеспечивает переход от обобщенных данных к детализированным данным.

У многомерных моделей есть недостатки, сдерживающие их применение:

– по сравнению с реляционными моделями они неэффективно используют память;

– в многомерных моделях заранее резервируется место для всех значений, даже если часть из них заведомо будет отсутствовать;

– выбор высокого уровня детализации при реализации гиперкуба может очень сильно увеличить размер многомерной БД.

4.3. Моделирование баз данных

Моделирование БД выполняется в соответствии с трехуровневой архитектурой СУБД, принятой в 1978 г. комитетом по стандартизации ANSI/SPARC (ANSI – Национальный институт стандартизации США; SPARC – Комитет планирования стандартов и меры). Трехуровневая архитектура СУБД включает в себя концептуальный, внутренний и внешний уровни.

Концептуальный уровень дает представление о логической схеме БД.

Внутренний уровень определяет физический вид БД. Он связан со способами хранения информации на физических устройствах. На этом уровне выбирают физические устройства для хранения данных, методы доступа к БД для извлечения и обновления информации, политику поддержания или повышения быстродействия системы.

Внешний уровень предназначен для отражения частной логической структуры данных для отдельного приложения (задачи) или пользователя. На этом уровне поддерживается санкционированный доступ к БД. На состав и структуру данных, доступных в приложении, наложены ограничения, а также заданы допустимые процедуры обработки этих данных.

Начальным этапом моделирования является разработка инфологической (информационно-логической) модели предметной области, основной особенностью которой является ее полная независимость от СУБД и физической среды хранения данных.

Инфологическая модель представляет собой обобщенное неформальное описание будущей БД. Это описание выполняется с использованием естественного языка, математических формул, таблиц, графиков и других, понятных проектировщикам, средств. Инфологическая модель является человеко-ориентированной.

Далее разрабатываются последовательно даталогическая (концептуальная), физическая и внешняя модели БД, которые ориентированы на конкретную СУБД и их трехуровневую архитектуру.

Даталогическая (концептуальная) **модель** соответствует концептуальному уровню архитектуры СУБД и представляет собой *интегрированные* концептуальные требования всех пользователей к БД в данной предметной области. Она состоит из множества экземпляров различных типов данных, структурированных в соответствии с требованиями конкретной СУБД к логической структуре БД. Даталогическая модель отражает логические связи между атрибутами объектов вне зависимости от их содержания и среды хранения и *может быть реляционной, иерархической, сетевой, объектно-ориентированной и т. д.* Даталогическую (концептуальную) модель еще называют ***схемой данных***. Она отображается в физическую память, образуя физическую модель БД.

Физическая модель соответствует внутреннему уровню архитектуры СУБД и характеризует размещение БД на запоминающих устройствах, методы доступа к ним, технику формирования указателей, индексирования и другие средства поддержки связей между данными.

Внешняя модель соответствует внешнему уровню архитектуры СУБД и является подмножеством концептуальной модели (подсхемой БД). Она отображает ту часть БД, которая необходима конкретному пользователю. Возможно пересечение внешних моделей по данным. На рис. 4.4 дан пример соотношения между концептуальной моделью и внешними моделями.

Инфологическая модель «сущность – связь»

Основными конструктивными элементами инфологических моделей являются сущности, атрибуты (их свойства), ключи и связи между сущностями.

Сущность – это любой объект, информацию о котором необходимо хранить в БД. Сущностями в экономических системах могут

быть готовые изделия, сотрудники, поставщики, заказы, города, денежные средства и т. д.

Рис. 4.4. Соотношение концептуальной и внешней моделей

Сущность имеет *тип и экземпляры*. Типом является название сущности, отличающее одну сущность от другой, например, ГОРОДА. Экземпляр представляет собой конкретное значение этой сущности, например, г. Полоцк Витебской области.

Атрибут – поименованная характеристика сущности. В реляционной таблице атрибуту соответствует поле. Имя атрибута для конкретной сущности должно быть уникальным, но может быть одинаковым для различного типа сущностей. Примерами атрибутов для сущности ГОРОДА могут быть: название города, код автоматической междугородной или международной телефонной связи (АМТС), административное значение (областной или районный центр), расстояние до ближайшей железнодорожной станции.

Атрибут сам может выступать в качестве сущности. Примером может служить сущность КОДЫ АМТС с атрибутами: код АМТС, название города, страна.

Ключ – набор атрибутов сущности, по значениям которых можно найти требуемый экземпляр сущности.

Связь – ассоциирование двух или более сущностей.

Типы связей

Различают следующие типы связей между сущностями:

- один к одному (1:1);
- один ко многим (1:M, в ЭВМ представлена как 1:∞);
- многие ко многим (M:M).

Связь *один к одному* обозначает, что в каждый момент времени одному экземпляру сущности *A* соответствует один и только один экземпляр сущности *B* и наоборот (Рис. 4.5).

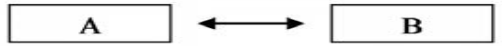


Рис. 4.5 Связь один к одному

Связь *один ко многим* имеет место в том случае, когда в каждый момент времени одному экземпляру сущности *A* соответствует ноль, один или более экземпляров сущности *B*, но каждый экземпляр сущности *B* связан только с одним экземпляром сущности *A* (Рис. 4.6)

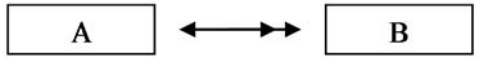


Рис. 4.6 Связь один ко многим

Связь *многие ко многим*, показывает, что в каждый момент времени одному экземпляру сущности *A* соответствует ноль, один или более экземпляров сущности *B*, и наоборот – каждому экземпляру сущности *B* соответствует ноль, один или более экземпляров сущности *A* (Рис. 4.7).

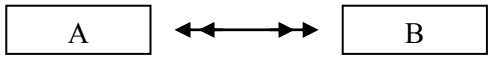


Рис. 4.7. Связь многие ко многим

ER- диаграммы

Инструментом графического представления инфологических моделей является язык ER-диаграмм (*Entity-Relationship* – сущность-связь).

В ER-диаграммах сущности изображаются *прямоугольниками*, ассоциации *шестиугольниками*, атрибуты – *овалами*, связи – *линиями* с обозначением *типа связи*.

Рассмотрим пример построения ER-диаграммы инфологической модели. Пусть модель описывает предметную область, отражающую командировки сотрудников фирмы. Все необходимые для включения в БД сведения, содержатся в сущностях, состав и атрибуты которых приведены в табл. 4.3. В столбце «Атрибуты» табл. 4.3 курсивом показаны ключи.

Таблица 4.3

Сущности и атрибуты инфологической модели

Сущность	Атрибут
СОТРУДНИКИ	*Фамилия (фам)*, номер личного дела (НЛД), код ВУЗа (ВУЗ)
ЛИЧНОЕ ДЕЛО	*Номер личного дела*, фамилия, характеристика 1 (Хар1) … характеристика N (ХарN)
КОМАНДИРОВКИ	Фамилия, город, расходы
ВУЗЫ	*Код вуза*, наименование Вуза (НВ), код специальности (КС)
СПЕЦИАЛЬНОСТИ	*Код специальности*, наименование специальности (НС)

На рис. 4.8 представлена инфологическая модель на языке ER-диаграмм.

ER-диаграмма, изображенная на рис. 4.8, удовлетворяет следующим условиям: сотрудники отдела могут ездить в разные города; каждый сотрудник может несколько раз быть в одном и том же городе; в один и тот же город могут ездить разные сотрудники; любая специальность может быть в любом вузе.

Преобразование ER- модели в реляционную модель данных

Процесс получения реляционной схемы базы данных из ER-диаграммы осуществляется по следующим правилам:

1. Каждая простая сущность превращается в отношение (таблицу). Простая сущность – это сущность, не являющаяся подтипом и не имеющая подтипов. Имя сущности становится именем отношения.

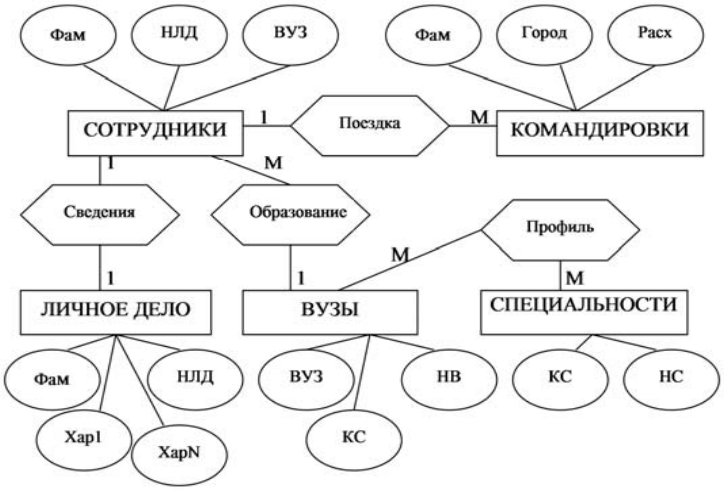


Рис. 4.8. ER-диаграмма инфологической модели БД

2. Каждый атрибут становится возможным столбцом с тем же именем. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения.

3. Компоненты уникального идентификатора сущности превращаются в первичный ключ отношения. Если имеется несколько возможных уникальных идентификаторов, выбирается наиболее используемый идентификатор.

4. Атрибуты, имеющие типы связи M:1 (и 1:1) становятся внешними ключами.

5. В таблицах, построенных на основе ассоциаций, внешние ключи используются для идентификации участников ассоциации, а в таблицах, построенных на основе характеристик и обозначений, внешние ключи используются для идентификации сущностей, описываемых этими характеристиками и обозначениями.

6. Если в концептуальной схеме присутствовали подтипы, то возможны два способа:

– все подтипы размещаются в одной таблице;

– для каждого подтипа строится отдельная таблица.

Полученные таблицы должны удовлетворять требованиям:

– каждая таблица состоит из однотипных строк и имеет уникальное имя;

– строки таблицы обязательно отличаются друг от друга хотя бы единственным значением поля, что обеспечивает однозначную идентификацию любой строки;

– таблицы имеют фиксированное число столбцов и их значений;

– в каждой таблице на пересечении строки и столбца должно находиться только одно значение или ничего;

– в столбцах таблицы размещаются однородные значения данных.

Важным этапом проектирования реляционной БД является обеспечение реляционной целостности данных.

Реляционная целостность данных – это система правил, используемых для поддержания связей между записями (строками) в связанных таблицах, а также для обеспечения защиты от случайного удаления или изменения связанных данных.

Существуют ограничения по условию целостности данных:

– *ограничение по сущностям* – каждая строка должна отличаться от остальных ее строк значением хотя бы одного столбца;

– *ограничение по ссылкам* – внешний ключ не может быть указателем на несуществующую строку той таблицы, на которую он ссылается.

Чтобы обеспечить целостность, работа с данными должна производиться с учетом перечисленных далее правил.

Невозможно ввести в связанное поле подчиненной таблицы значение, отсутствующее в связанном поле главной таблицы. Однако можно ввести пустое значение, показывающее, что для данной записи связь отсутствует.

Не допускается удаление записи из подчиненной таблицы, если существуют связанные с ней записи в главной таблице.

Невозможно изменить значение поля в подчиненной таблице, если оно является ключевым в главной таблице.

Нормализация таблиц

На завершающем этапе моделирования БД проводится процедура нормализации. Цель нормализации состоит в том, чтобы набор таблиц и состав их полей удовлетворяли условию минимальности по следующим параметрам:

– избыточность полей в таблицах;

– состав первичных ключей;

– нежелательные функциональные зависимости между полями;

– дублирование данных;

– трудоемкость выполнения операций включения, удаления и модификации данных;

– время выполнения запросов на выборку данных.

Нормализация таблиц – это пошаговый процесс композиции или декомпозиции исходных таблиц в таблицы, обладающие лучшими свойствами при включении, изменении и удалении данных.

Теория нормализации оперирует с пятью нормальными формами таблиц и основывается на наличии функциональных зависимостей между полями таблиц.

Функциональная зависимость. Поле *B* таблицы функционально зависит от поля *A* той же таблицы в том и только том случае, когда в любой заданный момент времени для каждого из различных значений поля *A* обязательно существует только одно из различных значений поля *B*. Поля *A* и *B* могут быть составными.

Полная функциональная зависимость. Атрибут *B* находится в полной функциональной зависимости от составного атрибута *A*, если оно функционально зависит от *A* и не зависит функционально от любого подмножества поля *A*.

Многозначная зависимость. Поле *A* многозначно определяет поле *B* той же таблицы, если для каждого значения поля *A* существует хорошо определенное множество соответствующих значений *B*.

При нормализации каждая последующая нормальная форма должна удовлетворять требованиям предыдущей формы и некоторым дополнительным условиям.

Таблица находится в ***первой нормальной форме*** *(1НФ)* тогда и только тогда, когда ни одна из ее строк не содержит в любом своем поле более одного значения и ни одно из ее ключевых полей не пусто.

Таблица находится во ***второй нормальной форме*** *(2НФ)*, если она удовлетворяет определению *1НФ* и все ее поля, не входящие в первичный ключ, связаны полной функциональной зависимостью с первичным ключом.

Таблица находится в ***третьей нормальной форме*** *(3НФ)*, если она удовлетворяет определению *2НФ* и ни одно из ее неключевых полей не зависит функционально от любого другого неключевого поля.

Во многих случаях приведение таблиц к *3НФ* позволяет эффективно работать с базой данных. При необходимости проводится дальнейшая нормализация.

Таблица находится в *нормальной форме Бойса-Кодда (НФБК),* если, и только если, любая функциональная зависимость между ее полями сводится к полной функциональной зависимости от возможного ключа.

Таблица находится в *пятой нормальной форме (5НФ)* тогда и только тогда, когда в каждой ее полной декомпозиции все проекции содержат возможной ключ. *Полной декомпозицией* таблицы называют такую совокупность произвольного числа ее проекций, соединение которых полностью совпадает с содержимым таблицы.

Четвертая нормальная форма *(4НФ)* является частным случаем *5НФ*, когда полная декомпозиция должна быть соединением ровно двух проекций.

4.4. Архитектура и функциональные возможности СУБД. Языковые и программные средства СУБД

Языковые средства СУБД необходимы для выполнения следующих функций:

- описания представления БД;
- выполнения операций манипулирования данными;
- управления БД.

Первая из этих функций обеспечивается *языком описания (определения) данных* (ЯОД) – *Shema Definition Language*. Описание БД средствами ЯОД является *схемой базы данных.* ***Схема БД*** описывает структуру базы данных и налагаемые на нее ограничения целостности в соответствии с теми правилами, которые регламентированы моделью данных используемой СУБД. В некоторых СУБД язык описания данных обеспечивает также возможность задания ограничений доступа к данным или полномочий пользователей.

Язык манипулирования данными (ЯМД) – *Shema Manipulation Language* содержит набор операторов манипулирования данными, позволяющих заносить данные в БД, удалять, модифицировать их и выбирать запрашиваемую информацию из БД.

В настоящее время имеются многочисленные примеры языков СУБД, объединяющих возможности описания данных и манипулирования данными в единых синтаксических рамках. Единый интегрированный язык современных СУБД содержит все необходимые средства для работы с базой данных начиная от ее создания и обеспечивает пользовательский интерфейс с БД. Наиболее популярным и стандартным для реляционных СУБД является язык SQL *(Structured Query Language* язык структурируемых запросов*),* разработанный фирмой IBM. Для поддержки объектных моделей предназначен язык OQL (Object Query Language), в основу которого положен SQL.

Примерами других языков этого класса могут служить: Quel системы Ingres, созданный Калифорнийским университетом; dBase семейства СУБД фирмы Asthon – Tate; R:Base фирмы Microrim.

Процедурным языком, при помощи которого осуществляется управление базой данных, является язык QBE *(Query-By-Example)*. Этот язык предоставляет пользователю удобный и унифицированный интерфейс для осуществления операций по ведению БД.

К программным средствам СУБД относятся языки программирования, позволяющие создавать сложные системы обработки данных, ориентированные на конкретные задачи и конкретного пользователя.

В СУБД MS Access программирование осуществляется с помощью макросов и модулей. ***Макросы*** – это небольшие программы на языке макрокоманд системы Access. Они представляют собой структуру, состоящую из одной или нескольких макрокоманд, которые выполняются либо последовательно, либо в порядке, заданном определенными условиями. Макросы позволяют запрограммировать практически все процедуры, которые составляют функциональные возможности СУБД и выполняются пользователями БД, не владеющих языками программирования.

Модули представляют собой процедуры на языке Visual Basic for Application (VBA).

VBA является общим языком для всех приложений Microsoft Office и позволяет решать любые задачи программирования, начиная от автоматизации действий конкретного пользователя и заканчивая разработкой сложных приложений, использующих Microsoft Office в качестве среды разработки. Язык VBA является объектно-ориентированным языком программирования и вычисления. Основой программ на VBA являются процедуры, состоящие из инструкций, выполняющих необходимые операции. Процедуры хранятся в модулях, из которых они запрашиваются на выполнение. Модуль служит для объединения процедур по функциональному назначению или привязки к форме или отчету.

В *web*-программировании активно используется *СУБД MySQL*. Для работы с БД этой системы применяют язык программирования PHP. Это С-подобный язык, предназначенный для быстрого создания программ на Web-сервере.

На языке PHP разрабатываются скрипты (scripting language) – кодовые (программные) инструкции по выполнению определенных действий над данными, выбранными из БД. Скрипты вставляются в HTML-документы, преобразуя их из статических в активные. Web-сервер просматривает документ, выполняет найденные в нем PHP-инструкции и результат выполнения инструкций возвращает пользователю.

С помощью PHP можно обрабатывать данные из форм, генерировать динамические страницы, создавать счетчики, гостевые книги и т. д. В PHP включена поддержка многих баз данных: FilePro, Informix, MySQL, Oracle, Sybase и др.

Структура СУБД

В структуре типичной СУБД выделяются ядро СУБД, компилятор языка базы данных (обычно SQL), подсистема поддержки времени выполнения, набор утилит.

Ядро СУБД является основной резидентной частью СУБД и предназначено для управления данными во внешней памяти, управления буферами оперативной памяти, управления транзакциями и журнализацией. Выполнение названных функций обеспечивается входящими в состав ядра СУБД компонентами: менеджером данных, менеджером буферов, менеджером транзакций и менеджером журнала.

Менеджер данных осуществляет управление данными во внешней памяти. Эта функция поддерживает необходимые структуры внешней памяти для хранения данных, непосредственно входящих в базу данных, и для служебных целей, например, для ускорения доступа к данным в некоторых случаях (обычно для этого используются индексы).

Менеджер буферов управляет буферами оперативной памяти. СУБД обычно работают с базами данных, размеры которых существенно больше доступного объема оперативной памяти. Если при обращении к любому элементу данных будет производиться обмен с внешней памятью, то вся система будет работать со скоростью устройства внешней памяти.

Буферизация данных в оперативной памяти позволяет временно содержать данные в процессе их получения, передачи, чтения или

записи в специальной области памяти. Эта процедура сглаживает скоростные и временные характеристики устройств.

Менеджер транзакций управляет объединением элементарных операций в транзакции для обеспечения целостности базы данных, управляет параллельно выполняющимися транзакциями и т. д. ***Транзакция*** – это последовательность операций над БД, рассматриваемых СУБД как единое целое. Если транзакция успешно выполняется, СУБД фиксирует изменения БД, произведенные этой транзакцией во внешней памяти. В противном случае ни одно из этих изменений никак не отражается на состоянии БД.

Менеджер журнала управляет журнализацией. Журнал – это особая часть БД, недоступная пользователям СУБД, в которую поступают записи обо всех изменениях основной части БД.

При журнализации используется стратегия «упреждающей» записи в журнал (протокол Write Ahead Log – WAL). Стратегия заключается в том, что запись об изменении любого объекта БД должна попасть во внешнюю память журнала раньше, чем измененный объект попадет во внешнюю память основной части БД. При соблюдении протокола WAL с помощью журнала можно решить все проблемы восстановления БД после любого сбоя.

Ядро СУБД обладает собственным интерфейсом, не доступным пользователям напрямую. Интерфейс используется в программах, формируемых компилятором SQL (или в подсистеме поддержки выполнения таких программ) и утилитах.

Компилятор языка БД преобразует операторы языка БД в выполняемую программу. Результат компиляции – выполняемая программа, представляется в машинных кодах или в выполняемом внутреннем машинно-независимом коде.

Подсистема поддержки времени выполнения используется для интерпретации внутреннего машинно-независимого кода при выполнении операторов программы.

Утилиты предназначены для таких процедур, которые неэффективно выполнять с использованием языка БД. К таким операциям относятся: загрузка и выгрузка БД, сбор статистики, глобальная проверка целостности БД и другие.

Функциональные возможности

Основные функциональные возможности СУБД – это, в первую очередь, создание базы данных и ее актуализация, ввод и обработка

данных, отбор данных с помощью запросов и предоставление информации пользователям БД в виде таблиц, графиков и отчетов.

Обеспечение целостности данных *на уровне БД* предполагает наличие средств, позволяющих удостовериться, что информация в БД всегда остается корректной и полной. Целостность данных должна обеспечиваться независимо от способа занесения данных в память (в интерактивном режиме, посредством импорта или с помощью специальной программы).

К средствам обеспечения целостности данных на уровне СУБД относятся:

– встроенные средства для назначения первичного ключа, в том числе средства для работы с типом полей с автоматическим приращением, когда СУБД самостоятельно присваивает новое уникальное значение;

– средства поддержания ссылочной целостности, которые обеспечивают запись информации о связях таблиц и автоматически пресекают любую операцию, приводящую к нарушению ссылочной целостности.

Большую пользу для обеспечения условий целостности данных, приносят триггеры, которые сохраняют связи между таблицами при добавлении, обновлении или удалении строк в таблицах. ***Триггером*** называют сохраненную процедуру специального типа, которая вступает в действие, когда пользователь изменяет данные в указанной таблице с помощью одной или нескольких из следующих операций: UPDATE, INSERT или DELETE. Триггеры позволяют выполнять запросы к другим таблицам и могут содержать сложные инструкции SQL

Импорт/экспорт данных. Функция *импорта* позволяет средствам СУБД обрабатывать информацию из внешних источников двумя способами:

– данные из других приложений (например, электронных таблиц) преобразуются из другого формата (например, формата электронной таблицы) и копируются в новую таблицу СУБД;

– объекты импортируются из одной БД в другую БД в рамках одной СУБД.

Экспорт представляет собой способ вывода данных и объектов БД в другую БД, электронную таблицу или формат файла, позволяющий другой БД, приложению или программе использовать эти

данные или объекты БД. Экспорт по своей сути напоминает копирование и вставку через буфер обмена.

Связывание таблиц – это установление связи с данными из другого приложения, что позволяет просматривать и изменять данные в исходном приложении и в СУБД.

Разработка и сопровождение приложений. СУБД обладают развитыми средствами для создания приложений, Этими средствами являются мощные языки программирования; средства реализации меню, экранных форм ввода-вывода данных и генерации отчетов; средства генерации прикладных программ (приложений), генерации исполняемых файлов.

Многопользовательские функции. Практически все СУБД предназначены для работы в многопользовательских средах, но обладают для этого различными возможностями. Наиболее общими функциями являются следующие:

– блокировка БД, файла, записи, поля;
– идентификация рабочей станции, установившей блокировку;
– обновление информации после модификации;
– контроль за временем обращения и повторения обращения;
– обработка транзакций;
– работа с сетевыми операционными системами.

Репликация баз данных представляет собой создание специальных копий – реплик общей БД, с которыми пользователи могут одновременно работать на разных компьютерах. Отличие реплики от обычной копии файлов БД заключается в том, что для реплики БД возможна синхронизация изменений. При проведении сеанса синхронизации все изменения, сделанные одним пользователем, могут автоматически вноситься в общую реплику, реплики других пользователей и наоборот.

Интеграция с Интернет отражает новейшие направления развития функциональных возможностей СУБД. Одно из этих направлений – публикация данных в Интернете и в корпоративной сети. СУБД позволяют публиковать объекты БД в виде статических и динамических Web-страниц. Во многих объектах БД возможно использование гиперссылок для перехода к другим документам. Существуют средства создания интерактивных Web-страниц для просмотра, ввода и анализа данных.

Основными WWW-технологиями доступа к БД являются следующие:

1. Однократное или периодическое преобразование содержимого баз данных в статические документы. Содержимое БД просматривает специальная программа-преобразователь, создающая множество файлов в виде связных HTML-документов. Полученные файлы копируются на WWW-сервер. Доступ к ним осуществляется как к статическим гипертекстовым документам сервера.

Такая технология эффективна при небольших массивах данных простой структуры с редким обновлением, а также при пониженных требованиях к актуальности данных, предоставляемых через ресурс WWW. Создание статических Web-страниц не требует использования механизма поиска и индексирования данных.

2. Динамическое создание гипертекстовых документов на основе содержимого БД.

Доступ к БД осуществляется с помощью специальной программы, запускаемой WWW-сервером в ответ на запрос WWW-клиента. Программа, обрабатывая запрос, просматривает содержимое БД, создает динамический выходной HTML-документ, возвращаемый клиенту.

Данная технология предпочтительна для больших БД со сложной структурой и при необходимости поддержки операций поиска, а также при частом обновлении и невозможности синхронизации преобразования БД в статические документы с обновлением содержимого.

Обеспечение безопасности данных – одна из важных функций современных СУБД. Средства безопасности обеспечивают выполнение таких операций, как шифрование прикладных программ, шифрование данных, защита паролем, ограничение доступа к БД или отдельным ее объектам.

4.5. Общая характеристика СУБД MS Access

СУБД *MS Access (Microsoft Access)* – одна из самых популярных систем для *IBM PC* и совместимых с ними компьютеров. *MS Access* входит в комплект *Microsoft Office* для профессиональной работы и эффективно используется совместно с другими его приложениями для ведения бизнеса на современном уровне. Новые версии *Access* предъявляют повышенные требования к компьютерной системе, в частности, к быстродействию микропроцессора, емкости запоминающих систем, операционной системе, сетевому программному обеспечению.

MS Access относится к СУБД ***реляционного*** типа. Это значит, что в БД информация организована в виде взаимосвязанных таблиц. Как и любая другая СУБД, *MS Access* предназначена для создания БД и управления ими. Основными функциями ее являются: добавление в БД записей, удаление из БД записей, обновление значений полей в записях, поиск в БД записей, удовлетворяющих заданным условиям. Для выполнения этих операций используется механизм запросов. Результатом выполнения запросов является либо отобранное по определенным критериям множество записей из таблицы, либо изменения в таблицах. Запросы к базе формируются на языке SQL.

В *MS Access* реализованы функции управления данными: защита данных от несанкционированного доступа, поддержка многопользовательского режима работы с данными, обеспечение целостности данных.

В отличие от других настольных СУБД *Access* ***хранит все данные в одном файле***, при этом распределяет данные по разным таблицам.

Таблица – это основной объект БД, хранилище информации. В базе может быть несколько таблиц, отражающих сведения из разных источников. Для совместной работы с данными из нескольких таблиц устанавливаются связи между таблицами.

К другим объектам, которые могут быть созданы в БД, относятся запросы, формы, отчеты, макросы, модули.

Запрос позволяет проанализировать данные с помощью групповых операций, вычислений и отбора данных, удовлетворяющих некоторым условиям.

Форма обеспечивает представление данных на экране в удобном для пользователя виде. Формы позволяют вводить, просматривать, изменять данные, являются средством поиска данных и отличаются многообразием.

Отчет служит для отображения данных при выводе на печать.

Макросом называют набор из одной или более макрокоманд, выполняющих определенные операции, такие как открытие форм или печать отчетов. Макросы могут быть полезны для автоматизации часто выполняемых задач. Например, при нажатии пользователем кнопки можно запустить макрос, который распечатает отчет.

Модуль – это набор описаний и процедур на языке Visual Basic для приложений, собранных в одну программную единицу.

СУБД *MS Access* полностью совместима с такими компонентами пакета Microsoft Office, как MS Excel, MS Word, MS Outlook. Взаимодействие MS Accessc и MS Outlook позволяет создавать и отправлять сообщения электронной почты с формами для ввода данных.

MS Access может работать с разными форматами файлов других систем, поддерживающих стандарт открытого доступа к данным ODBC (Open Database Connectivity) – Oracle, Microsoft SQL Server, Sybase SQL Server. Стандарт ODBC определяет язык и набор протоколов для обмена между пользовательским приложением и самими данными, хранящимися на сервере, и используется как средство коммуникации между настольным персональным компьютером (клиентом) и сервером.

С помощью *Office Access* можно сохранить отчет в формате PDF (Portable Document Format) или XPS (формат XML Paper Specification). Это позволяет произвести распечатку или публикацию файла, а также переслать его по электронной почте. Можно опубликовать свои файлы в библиотеках или перемещать приложение в Windows SharePoint Services, что дает возможность участникам группы легко взаимодействовать друг с другом посредством обозревателя.

В MS Access существуют функции и технологии, увеличивающие производительность системы: технология Rushmore; быстрая сортировка (QuickSort); средство наиболее часто выполняемых запросов (Top Value queries*)*.

MS Access представляет мощный инструментарий для разработчика. Универсальная среда разработчика со встроенным отладчиком обеспечивает возможности программирования на уровне Microsoft Visual Basic.

MS Access имеет большое количество инструментальных средств для эффективного выполнения практически любых работ с БД. К ним относятся Мастера и Построители.

Назначение Мастеров – помочь даже мало подготовленному пользователю создать свою БД, обрабатывать данные с помощью форм, запросов и отчетов, проводить анализ таблиц БД и т. д. Мастера предоставляют пользователю интерактивный пошаговый интерфейс для выполнения определенной работы.

Многие Мастера можно объединить в группы:

Создание новой БД и новых объектов БД – таблиц, форм, запросов, подчиненных отчетов и форм, списков и полей со списком в формах, страниц доступа к данным, новой БД MS SQL Server, с которой связывается новый проект MS Access.

Связывание объектов внутри БД и с документами других приложений, например, полей подчиненных форм и отчетов, таблицы или списка HTML из Интернета или интрасети с таблицей MS Access, папок Exchange или Outlook с таблицей БД MS Access, данных электронных таблиц с таблицей Microsoft Access, помещение сводной таблицы MS Excel в форму MS Access и др. Связывание обеспечивает реализацию функций Импорта и Экспорта объектов БД.

Создание элементов управления – кнопок и групп переключателей в форме, кнопок на странице доступа к данным.

Поддержка многопользовательского режима – разделение БД на данные и интерфейс с тем, чтобы несколько пользователей имели на своих компьютерах копии интерфейса, связанного с данными на сервере; создание или изменение частичной реплики реплицированной БД, разрешение конфликтов между реплицированными БД во время синхронизации.

Анализ, сервис и преобразование – анализ эффективности БД и выдача списка рекомендаций по ее совершенствованию; генерация отчета MS Access, отображающего характеристики структуры объектов БД; выдача сведений о модеме после нажатия кнопки «Автонабор» в режиме формы; преобразование БД MS Access в БД MS SQL Server; преобразование макросов в программы Visual Basic.

Построители служат вспомогательным средством, облегчающим работу пользователей. К ним относятся:

– Построитель запросов – создает правильный синтаксис для запроса;

– Построитель полей – создает поля в таблице;

– Построитель строк подключения ODBC – создает правильный синтаксис для связи с объектами ODBC;

– Построитель смарт-тегов – отображает список доступных смарт-тегов и их действий. С помощью смарт-тегов можно сэкономить время при выполнении тех действий в приложении Ms Access, для которых обычно предназначены другие программы. Например,

с помощью смарт-тега имени пользователя происходит добавление имени в папку MS Outlook «Контакты».

Построитель цветов – предлагает палитру для создания настраиваемых цветов.

Построитель выражений – помогает пользователю составить в интерактивном режиме необходимую формулу.

Выражение представляет собой однозначно заданную пользователем последовательность элементов, состоящую из идентификаторов, операторов (+, –, *, ^ и др.), функций и констант. Выражения могут быть использованы в различных местах базы данных: в таблицах, запросах, формах, отчетах и макросах. В MS Access выражения используются, когда необходимо выполнить следующие действия:

1. Вычислить значения, не содержащиеся в данных в явном виде. Можно вычислить значения для полей таблицы, запросов и элементов управления в форме или отчете.
2. Задать значение по умолчанию для поля таблицы или элемента управления в форме или отчете. Заданные значения отображаются при открытии таблицы, формы или отчета.
3. Задать условие на значение. Условия на значение контролируют значения, которые могут быть введены в поле или элемент управления.
4. Задать условие отбора в запросе.

В MS Access имеются службы Графического конструктора связей (Graphical System Relationships Builder – графический построитель схемы данных) и Графического запроса (Graphical query). Эти средства позволяют не только создать БД, но и наглядно сконструировать ее.

4.6. Основные объекты MS Access

Таблицы

Таблицы в MS Access создают с помощью шаблонов таблиц и полей, в режиме таблицы путем ввода данных и в режиме конструктора.

Шаблоны таблиц и полей используются для быстрого создания таблицы. *Шаблон таблицы* – это пустая заготовка таблицы с заданной структурой, то есть составом и характеристикой полей. После выбора шаблона пользователь при необходимости может изменить

структуру в соответствии со своими требованиями. Можно добавить поля из области задач *Шаблоны полей*. *Шаблон поля* – это предопределенное поле, которое включает имя поля, тип данных, значение свойства поля *Формат* и другие свойства поля. Выбранные поля перетаскиваются на таблицу методом «drag and drop» (или двойным щелчком мыши). Далее пользователь заполняет таблицу конкретными данными с помощью клавиатуры.

В режиме конструктора пользователь по своему проекту задает структуру таблиц, указывая состав и перечень полей, их характеристики и свойства. Затем переходит в режим таблицы для ввода значений полей.

Режим таблицы дает возможность проектировать и заполнять таблицу данными при ее наглядном отображении на экране. Этот режим активизируется по умолчанию при открытии новой БД или устанавливается при использовании элемента *Таблица* во вкладке *Создание* интерфейса *Лента*. MS Access автоматически создает первое поле *Код* с типом *Счетчик*. В режиме таблицы доступны многие возможности режима конструктора, например, добавление и удаление полей, установка типов полей.

Схема данных

Создать связь между таблицами можно двумя способами:

- с помощью окна *Связи;*
- с помощью перетаскивания поля из области *Список полей* в таблицу.

Связываемые таблицы должны иметь **общее поле**, которое в первой таблице является **первичным ключом**, а во второй таблице – **внешним ключом**.

Общие поля могут иметь различные имена, но они должны иметь одинаковый тип данных. Однако, когда поле первичного ключа имеет тип *Счетчик*, тогда поле внешнего ключа может также быть числовым полем, если свойство *Размер поля* обоих полей совпадает. Если оба общих поля являются числовыми, у них должно совпадать значение свойства *Размер поля*.

Первый способ связывания таблиц выполняется в окне *Связи*. Для этого на вкладке *Работа с базами данных* в группе *Отображение* выбирается пункт *Схема данных*.

Если ни одной связи еще не определено, автоматически откроется диалоговое окно *Добавить таблицу*, в котором выбираются необходимые для связи таблицы и запросы. Если окно не открылось, на вкладке *Структура* в группе *Связи* его надо открыть кнопкой

Добавить таблицу и после добавления таблиц и запросов на вкладку *Схема данных* окно *Добавить таблицу* следует закрыть.

Связь между таблицами создается перетаскиванием, как правило, поля первичного ключа из одной таблицы на общее поле (поле внешнего ключа) в другой таблице. В появившемся диалоговом окне *Изменение связей* выполняются действия по обеспечению целостности данных, включая настройку каскадных параметров. *Каскадное обновление и удаление связанных записей* обеспечивают автоматическое изменение данных в связанной таблице при внесении изменений в первую таблицу.

Чтобы создать отношение «один-к-одному», оба общих поля должны иметь уникальный индекс. Это означает, что свойства *Индексированное* этих полей должны иметь значения Да (Совпадения не допускаются).

Чтобы создать отношение «один-ко-многим», поле на одной стороне отношения (как правило, поле первичного ключа) должно иметь уникальный индекс. Поле на стороне «многие» не должно иметь уникального индекса. Это означает, что свойство *Индексировано* этого поля должно иметь значение *Нет,* либо *Да* (Допускаются совпадения).

При втором способе используется область *Список полей,* которая отображает поля, доступные в связанных таблицах, а также поля, доступные в других таблицах БД. При перетаскивании поля из «другой» (несвязанной) таблицы и выполнении инструкций мастера подстановок автоматически создается новое отношение «один-ко-многим» между таблицей в области *Список полей* и текущей таблицей.

Запросы

Запрос представляет собой обращение к данным для получения информации и выполнения действий с данными.

Основным средством создания запросов к БД является конструктор запросов. Окно конструктора имеет две области. В верхнюю область окна заносятся таблицы или запросы, являющиеся источниками полей для создаваемого запроса. В нижней части окна располагается **бланк запроса**, каждая строка которого имеет свое назначение:

Поле. В этой строке помещаются имена полей из источников данных для создания нового запроса. Каждое поле помещается в тдельный столбец бланка запроса в этой строке.

Имя таблицы. В каждом столбце в этой строке отражается имя источника данных для находящегося в нем поля.

Сортировка. Для поля, значения которого необходимо упорядочить, указывается тип сортировки.

Вывод на экран. Строка предназначена для установки флажков в ех полях, которые должны отображаться в новом запросе на экране.

Условие отбора. В данную строку (и в строку, расположенную ниже ее) вводятся критерии отбора записей, ограничивающие поиск записей в источниках данных. Критерий вводится в ячейку на пересечении строки *Условие отбора* и того поля, по которому будет выполняться отбор записей.

При проектировании некоторых типов запросов в бланке запросов появляются новые стоки, например, *удаление, обновление, групповая операция*.

Технология создания запросов включает следующие основные действия: определение источников данных (таблиц, запросов), отбор полей из источников данных для нового запроса, запись условий отбора, формирование вычисляемых полей и выражений.

Различают два типа запросов к БД: запросы на изменение и запросы на выборку.

Запросы на изменения включают четыре типа: *запрос на добавление, запрос на удаление, запрос на обновление и запрос на создание таблицы*

Запросы на выборку предназначены для извлечения данных из таблиц для просмотра или выполнения расчетов: *простой запрос, запрос с вычисляемым полем, параметрический запрос, итоговый запрос, перекрестный запрос*, *запрос с повторяющимися записями*.

Кроме этого, существует три основных типа **запросов SQL**: *запрос на объединение, запрос к серверу и управляющий запрос.*

Рассмотрим некоторые типы запросов.

Запрос на удаление предназначен для удаления записей из одной таблицы или нескольких таблиц, связанных отношениями «один-к-одному» и «один-ко-многим». При проектировании запроса в бланк запроса помещаются только те поля, по которым будут указаны условия отбора на удаление.

Запрос на обновление используют в том случае, если необходимо обновить значения полей на новые. Например, для всех работников увеличивается премия на 10%. При проектировании запроса в бланке запроса в строке *Обновление* в поле Премия, значения

которого требуется изменить, вводится выражение, обеспечивающее замену прежних значений на новые:

[Премия]*1,1

Если премия изменяется только для сотрудников со стажем больше 15 лет, то дополнительно в строку *Условие отбора* в поле Стаж вводится критерий для отбора нужных записей: > 15.

Запрос с вычисляемым полем. Содержит, кроме полей, выбранных из источников данных, дополнительные поля, значения которых будут являться результатом вычислений.

Итоговый запрос. Создается с целью выполнения расчетов в тобранных группах записей. К итоговым операциям относятся:

Sum – суммирование отобранных значений поля;

Avg – вычисление арифметического среднего отобранных значений поля;

Min – нахождение минимального значения среди отобранных значений поля;

Max – нахождение максимального значения среди отобранных значений поля;

Count – вычисление количества отобранных значений в поле;

StDev – расчет стандартного отклонения для отобранных значений поля;

Var – расчет дисперсии для отобранных значений поля;

First –отображение значения поля в первой отобранной записи;

Last – отображение значения поля в последней отобранной записи.

Параметрический запрос. Позволяет задавать разные условия отбора записей непосредственно при запуске запроса. При проектировании запроса в строку *Условие отбора* для заданного поля вводится приглашением на ввод параметра. Текст приглашения заключается в вадратные скобки. Например, [Введите фамилию сотрудника]. При появлении на экране этого приглашения пользователь должен набрать на клавиатуре и ввести конкретную фамилию. В результате запроса на экран выводятся сведения по указанному сотруднику.

Перекрестный запрос отображает выбранные из источников данных записи в формате электронной таблицы, то есть часть полей выводится на экран по строкам, а часть – по столбцам.

Условия отбора

При создании запросов важно правильно сформулировать условия отбора записей из БД. В MS Access доступны следующие возможности:

- простой критерий выборки;
- точное несовпадение значений одного поля;
- неточное совпадение значений поля;
- выбор по диапазону значений;
- объединение критериев нескольких полей;
- условие отбора для результатов итоговых вычислений.

Простой критерий выборки. Записи выбираются по совпадающим значениям поля. Например, из поля **Город** необходимо выбрать значения Минск. Для этого в бланке запроса в строке **Условие отбора** в графе **Город** вводится с клавиатуры значение **«Минск»**.

Точное несовпадение значений одного поля. Из базы выбираются все записи, кроме тех, для которых задано условие. Например, необходимо выбрать все записи с полем **Город**, кроме тех, которые в том поле имеют значение **Минск**. Для этого в строке **Условия отбора** в графе **Город** вводится выражение **Not «Минск»** или **<> «Минск»**. Логический оператор **Not** исключает записи со значением **Минск**, оператор сравнения <> означает «не равно».

Неточное совпадение значений поля. Такое условие можно задавать, если не известны значения полей. Для выборки используется оператор сравнения **Like** (подобный). Рядом с оператором записывается образец, содержащий или точное значение, например, **Like «Петров»,** или включающий символы шаблонов, например, **Like «Пет*»**.

Access допускает следующие символы шаблонов:

? – любой один знак;

* – ноль или более знаков;

– любая одна цифра;

[список знаков] – любой один знак в списке знаков;

[!список знаков] – любой один знак, не входящий в список.

Кроме списка знаков в квадратные скобки может заключаться диапазон символов, например, [Б-Р]. Условие [б-рБ-Р] позволяет выбрать как заглавные, так и прописные буквы.

При условии **Like «[БР]*»** выбираются все фамилии, которые начинаются на Б или Р.

Выбор по диапазону значений. Для задания диапазона значений используются операторы:

> (больше),

>= (не менее, больше или равно),

< (меньше**)**,

<= (не более, меньше или равно) (например, >= 10).

Between ... and ... (служит для проверки принадлежности диапазону, верхняя и нижняя граница которого соединена логическим оператором AND (например, between 1990 and 1995).

Операторы можно употреблять с текстовыми и цифровыми полями, а также с полями дат.

Объединение критериев одного поля. Если на одно поле налагается более одного условия, то условные выражения могут быть соединены с помощью операторов **Or** (**ИЛИ**) и **And** (**И**).

Объединение критериев нескольких полей. В запросе может быть несколько условий отбора. В этом случае имеют место два варианта выборки записей:

– запись выбирается только при выполнении всех условий, что соответствует логической операции **И**. Запрос называется **И-запросом**;

– запись выбирается при выполнении хотя бы одного условия, что соответствует логической операции **ИЛИ**. Запрос называется **ИЛИ-запросом**.

При построении **ИЛИ-запроса** каждое условие, входящее в критерий, должно располагаться на отдельной строке. При построении **И-запроса** каждое условие, входящее в критерий, должно располагаться в одной строке.

В *итоговых запросах* существуют два типа критериев отбора записей.

Первый тип исключает записи, не удовлетворяющие критериям, перед выполнением итоговых вычислений. Второй тип критериев применяется к результату итоговых вычислений.

Формы

Приложение *MS Access* создает формы с помощью нескольких средств: форма, разделенная форма, несколько элементов, мастер форм, пустая форма, конструктор форм.

Форма. При использовании этого средства все поля базового источника данных размещаются в форме. В форме отображается только одна запись и есть возможность просмотра других записей.

Если MS Access обнаруживает одну таблицу, связанную отношением «один-ко-многим» с таблицей или запросом, который использовался для создания формы, MS Access добавляет таблицу данных в форму, основанную на связанной таблице или запросе. Например, если создается простая форма, основанная на таблице «Сотрудники», и между таблицами «Сотрудники» и «Заработная плата» определено отношение «один-ко-многим», то в таблице данных будут отображаться все записи таблицы «Заработная плата», относящиеся к текущей записи сотрудника. Если таблица данных в форме не нужна, ее можно удалить. Если существует несколько таблиц, связанных отношением «один-ко-многим» с таблицей, которая использовалась для создания формы, то данные таблицы в форму не добавляются.

Разделенная форма – позволяет одновременно отображать данные в двух представлениях – в режиме формы и в режиме таблицы.

Эти два представления связаны с одним и тем же источником данных и всегда синхронизированы друг с другом. При выделении поля в одной части формы выделяется то же поле в другой части. Данные можно добавлять, изменять или удалять в каждой части формы (при условии, что источник записей допускает обновление, а параметры формы не запрещают такие действия).

Работа с разделенной формой дает преимущества обоих типов формы в одной форме. Например, можно воспользоваться табличной частью формы, чтобы быстро найти запись, а затем просмотреть или изменить запись в другой части формы.

Несколько элементов. Создаваемая форма внешне напоминает таблицу. Данные расположены в строках и столбцах, и одновременно отображается несколько записей. К такой форме можно добавлять графические элементы, кнопки и другие элементы управления.

Мастер форм. Дает больше свободы для выбора полей, отображаемых в форме. Мастер позволяет указать способ группировки и ортировки данных, а также включить в форму поля из нескольких таблиц или запросов, при условии, что заранее заданы отношения между этими таблицами и запросами.

Пустая форма. Используется для быстрого построения формы с ебольшим количеством полей. MS Access открывает пустую форму и одновременно отображает область **Список полей,** из которой выбираются двойным щелчком мыши или перетаскиванием необходимые поля.

Конструктор форм. Источником данных для формы может быть только одна таблица или запрос. Основной структурной единицей формы, в которой пользователь размещает поля данных, является *Область данных*, видимая на экране по умолчанию. К другим структурным частям формы относятся *заголовок формы, верхний и нижний колонтитулы, примечание формы*, которые вызываются на экран пользователем.

В области данных размещают поля данных из источника данных посредством окна *Список полей*, а также вычисляемые поля, отсутствующие в источнике данных (создаются только в форме ленточного вида). Выражения для вычисляемых полей записываются с помощью *Построителя выражений*.

Инструментом конструирования формы являются **элементы управления**. Наиболее часто используемый элемент управления – поле. К ругим элементам управления относятся: надписи, флажки, элементы управления подчиненных форм и отчетов и др. Элемент управления «поле» может быть присоединенным, свободным и вычисляемым.

Присоединенный элемент управления – элемент управления, источником данных которого служит поле таблицы или запроса. Присоединенный элемент управления формируется посредством окна *Список полей* и служит для отображения значений полей источника данных. Это наилучший способ создания присоединенного элемента управления по двум причинам:

– присоединенный элемент управления имеет связанную с ним подпись, которой по умолчанию становится имя поля (или подпись, определенная как свойство для этого поля в источнике данных), следовательно, вводить текст подписи не требуется.

– присоединенный элемент управления наследует значения свойств полей источника данных, например, *Формат, Число десятичных, Маска ввода*.

Свободный элемент управления – элемент управления, не имеющий источника данных. Свободные элементы управления

служат для вывода на экран текста, линий, прямоугольников и исунков. Примером свободного элемента является *Надпись*.

Вычисляемые элементы управления – элемент управления, источником данных которого является выражение, а не поле.

Конструктор *MS Access* позволяет создавать **формы с одчиненной формой**. Подчиненная форма – это такая форма, которую внедряют в другую форму, называемую основной, с целью получения дополнительной информации из другой таблицы. Сначала создают подчиненную форму, затем основную форму и осле этого помещают подчиненную форму в основную. Допускается несколько уровней подчиненности форм.

Для изменения форм используются режимы макета и конструктора.

Режим макета. Режим макета представляет собой наиболее наглядный режим для изменения форм. Его можно использовать для внесения практически любых изменений в форму: корректировать данные, задавать размеры элементов управления, оформлять внешний вид формы. В этом режиме можно изменять также структуру формы, например, настроить размеры полей в соответствии с данными, которые отображены на экране.

Режим конструктора. Режим конструктора позволяет более подробно просмотреть структуру формы. Можно просматривать разделы колонтитулов и данных формы. В этом режиме форма не ыполняется, поэтому при внесении изменений невозможно просматривать базовые данные. Однако в режиме конструктора удобнее выполнять другие работы:

– добавлять в форму различные элементы управления, такие как надписи, рисунки, линии и прямоугольники.

– изменять источник элемента управления «Поле» непосредственно в поле без использования окна свойств.

– изменять размеры разделов формы, таких как «Заголовок формы» или «Область данных».

– изменять свойства формы, которые недоступны для изменения в режиме макета (например, *Представление по умолчанию* или *Режим формы*).

Отчеты

Отчет является основным объектом MS Access, предназначенным для вывода на печать данных из таблиц и запросов. В отчетах,

как правило, MS Access систематизирует данные по группам и подсчитывает итоги как общие, так и промежуточные. Кроме данных, в тчете содержится информация о макете отчета: подписях, заголовках, рисунках и другие сведения.

Приложение MS Access создает отчеты следующими средствами: отчет, мастер отчетов, пустой отчет, конструктор отчетов.

Самый быстрый способ создания отчета, так как отчет формируется без запроса дополнительной информации. В отчет включаются все записи источника данных – таблицы или запроса. Отчет при необходимости можно изменить в режиме макета или конструктора. При каждом открытии отчета в нем отображаются фактические на нный момент записи из источника данных.

Мастер отчетов. Формирует отчет в интерактивном режиме, предоставляя пользователю возможность добавлять в отчет поля из ескольких таблиц или запросов, если связи между этими таблицами и запросами заданы заранее. При этом можно указать способ группировки и сортировки данных.

Предварительный просмотр отчета в разных масштабах позволяет увидеть, как будет выглядеть отчет при печати.

Приложение MS Access имеет средство **Мастер наклеек,** которое помогает создавать наклейки большинства стандартных размеров. Источником записей для наклеек служит таблица или запрос.

Пустой отчет. Используется для быстрого создания отчета с ебольшим количеством полей. MS Access открывает пустой отчет. Одновременно в правой части окна отображается область **Список полей,** из которой необходимо выбрать двойным щелчком мыши или перетаскиванием необходимые поля.

С помощью инструментов, представленных в группе **Элементы управления** на вкладке **Форматирование**, можно добавить в отчет эмблему компании, заголовок, номера страниц, дату и время.

Конструктор отчетов. Структура отчета, как и структура формы, имеет несколько разделов, которые можно перечислить следующим образом: Заголовок отчета, Верхний колонтитул, Заголовок группы, Область данных, Примечание группы, Нижний колонтитул, Примечание отчета.

Заголовок отчета. Служит для размещения заголовка отчета. В аголовок включается эмблема компании, название отчета или дата. Если в заголовке отчета помещен вычисляемый элемент управления, использующий статистическую функцию **Sum**, сумма рассчитывается для всего отчета. Заголовок отчета печатается перед верхним колонтитулом только один раз в начале отчета.

Верхний колонтитул. Используется для размещения названий столбцов в отчетах табличной формы. Печатается вверху каждой страницы.

Заголовок группы. Содержит название группы и печатается перед каждой новой группой записей. Если поместить в заголовок группы вычисляемый элемент управления, использующий статистическую функцию **Sum**, сумма будет рассчитываться для текущей группы.

Область данных. Предназначена для размещения полей данных из источника данных посредством окна *Список полей*. В разделе создаются также вычисляемые поля, отсутствующие в источнике данных. Технологии включения полей в область данных отчета и ормы аналогичны.

Примечание группы. Размещается в конце каждой группы записей. Примечание группы можно использовать для печати сводной информации по группе.

Нижний колонтитул. Располагается внизу каждой страницы. Используется для нумерации страниц и для печати постраничной информации.

Примечание отчета. Примечание отчета можно использовать для печати итогов и другой сводной информации по всему отчету. Печатается один раз в конце отчета.

В проект отчета можно вносить изменения в режимах макета и онструктора.

Просматривать отчет можно различными способами:

– в режиме отчета, если необходимо временно изменить *состав данных* в отчете перед его печатью или скопировать данные отчета в буфер обмена. Непосредственно в режиме отчета можно применять ***фильтры*** к данным отчета;

– в режиме макета, если необходимо изменить *макет* отчета, имея перед собой его данные;

– в режиме предварительного просмотра, если требуется лишь просмотреть отчет перед печатью. Только в этом режиме будут видны несколько столбцов отчета. В предыдущих режимах в отчете отображается один столбец.

Отчет можно не выводить на печать, а отправить его получателю в виде сообщения электронной почты.

4.7. Основы языка SQL

Общие сведения

Появление и развитие языка *SQL* связано с созданием теории реляционных БД. Математической основой языка SQL является реляционная алгебра и реляционное исчисление.

Прообраз возник в 1970 г. в лаборатории Санта-Тереза фирмы IBM. В настоящее время популярность SQL настолько велика, что разработчики нереляционных СУБД снабжают свои системы SQL-интерфейсом.

Язык SQL имеет официальный стандарт – ANSI/ISO. Стандарт языка SQL регламентируется Американским институтом стандартов (American National Standard's Institute – ANSI) и Международной организацией стандартизации (International Organization for Standardization – ISO). Большинство разработчиков придерживаются этого стандарта, однако часто расширяют его для реализации новых возможностей обработки данных. Стандартизация возможностей SQL продолжается. Линия развития стандартов представлена стандартами: ISO-ANSI SQL, SQL/92, SQL2, SQL3.

SQL сочетает в себе возможности языка определения данных, языка манипулирования данными и языка запросов. При этом он еализует и основные функции реляционных СУБД.

SQL не является языком программирования в традиционном представлении. На нем пишутся не программы, а запросы к БД, поэтому этот язык называют ***языком запросов,*** языком *декларативным,* а не процедурным. Это означает, что с его помощью можно сформулировать, что необходимо получить, однако нельзя указать, как это следует сделать. В отличие от процедурных языков программирования (С, Паскаль), в языке SQL отсутствуют алгоритмические конструкции, операторы цикла, условные переходы и т. д.

Запрос в языке SQL состоит из одного или нескольких операторов, следующих один за другим и разделенных точкой с запятой. Каждая последовательность операторов языка *SQL* реализует определенное действие над БД. Оно осуществляется за несколько шагов, на каждом из которых над таблицами выполняются определенные действия.

Структура оператора *SQL*

Каждый оператор *SQL* начинается с ключевого слова, которое определяет, что делает этот оператор (SELECT, INSERT, DELETE).

В операторе содержатся предложения, содержащие сведения о том, над какими данными производятся операции. Каждое предложение начинается с ключевого слова, такого как FROM, WHERE и др.

Структура предложения зависит от его типа: ряд предложений содержит имена полей или таблиц, некоторые могут включать дополнительные ключевые слова, константы или выражения.

На рис. 4.9 приведен пример простой структуры оператора SELECT. Синтаксис этого оператора выглядит следующим образом:

```
SELECT столбцы (или *)
FROM таблица (ы)
[WHERE ограничение(я)]
[ORDER BY столбец];
```

Рис. 4.9. Структура оператора SELECT в языке *SQL*

За ключевым словом SELECT следуют сведения о том, какие именно поля необходимо включить в результирующий набор данных. Звездочка (*) означает, что в набор данных попадают все поля таблицы.

Для указания имен таблиц, из которых выбираются записи, применяется ключевое слово FROM.

Для фильтрации результатов, возвращаемых оператором SELECT, используется предложение WHERE.

Выражение IS NOT NULL означает, что соответствующий столбец результирующего набора данных не должен иметь пустых значений.

Предложение ORDER BY является необязательным и применяется для сортировки результирующего набора данных по одному или нескольким столбцам.

Основные операторы языка SQL

Наиболее важные операторы выделены в стандарте ANSI/ISO SQL.

Data Definition Language (DDL) – язык описания данных.

Эта составляющая языка содержит операторы, позволяющие создавать, модифицировать и уничтожать базы данных и объекты внутри них (таблицы, представления) (табл. 4.4).

Таблица 4.4

Операторы *DDL*

Оператор	Описание
CREATE TABLE	Добавление новой таблицы к базе данных
DROP TABLE	Удаление таблицы из базы данных
ALTER TABLE	Изменение структуры имеющейся таблицы
CREATE VIEW	Добавление нового представления к базе данных
DROP VIEW	Удаление представления
CREATE INDEX	Создание нового индекса
DROP INDEX	Удаление существующего индекса

Data Manipulation Language (DML) – язык манипулирования данными. Эта составляющая языка содержит операторы, позволяющие добавлять, выбирать, удалять и модифицировать данные. Эти операторы не обязательно должны завершать транзакцию, внутри которой они вызваны. Описание операторов представлено в табл. 4.5.

Таблица 4.5

Операторы *DML*

Оператор	Описание
SELECT	Выбор данных
INSERT	Вставка данных
DELETE	Удаление данных
UPDATE	Обновление данных

Иногда оператор SELECT относят к отдельной категории Data Query Language (DQL) – язык запрашиваемых данных.

Transaction Control Language (TCL) – язык управления транзакциями.

Транзакция – это группа операций модификации данных, имеющих логически законченный смысл, после выполнения которых база данных останется корректной. Операторы данного класса (табл. 4.6) применяются для управления изменениями, выполняемыми группой операторов DML.

Таблица 4.6

Операторы *TCL*

Оператор	Описание
COMMIT	Завершение транзакции и сохранение изменений в базе данных
ROLLBACK	Откат транзакции и отмена изменений в базе данных
SET TRANSACTION	Установка параметров доступа к данным в текущей транзакции

Data Control Language (DCL) – язык управления данными. Операторы этой группы (табл. 4.7), иногда называемые операторами Access Control Language (язык управления базой), применяются для осуществления административных функций, присваивающих или отменяющих право (привилегию) использовать базу данных, таблицу базы данных, а также выполнять те или иные операторы SQL.

Таблица 4.7

Операторы GCL

Оператор	Описание
GRANT	Присвоение привилегии
REVOKE	Отмена привилегии

Функциональные возможности языка SQL

Основные функциональные возможности языка SQL приведены ниже.

Определение данных. Эта функция SQL представляет собой описание структуры поддерживаемых данных и организацию реляционных отношений (таблиц). Для ее реализации предназначены операторы создания базы данных, создания таблиц и доступа к данным.

Создание базы данных. Для создания новой базы данных используется оператор CREATE DATABASE. В структуре оператора указывается имя создаваемой базы данных.

Создание таблиц. Базовая таблица создается с помощью оператора CREATE TABLE. В этом операторе указываются имена полей, типы данных для них, длина (для некоторых типов данных). В SQL используются следующие типы данных:

INTEGER – целое число;

CHAR – символьное значение;

VARCHAR – символьное значение, сохраняются только непустые символы;

DECIMAL – десятичное число;

FLOAT – число с плавающей запятой;

DOUBLE PRECISION – удвоенная точность с плавающей точкой;

DATETIME – дата и время;

BOOL – булевое значение.

В операторе создания таблицы указываются ограничения на значения столбцов и на таблицу. Возможные ограничения показаны в табл. 4.8.

Для реляционной модели данных существенным является указание внешнего ключа (FOREIGNKEY). При объявлении внешних

ключей необходимо наложить соответствующие ограничения на столбец, например, NOT NULL.

В SQL-предложении CHECK обозначает семантические ограничения, обеспечивающие целостность данных, чтобы, например, ограничить множество допустимых значения определенного столбца.

Нельзя использовать оператор создания таблицы несколько раз для одной и той же таблицы. Если после ее создания обнаружились неточности в ее определении, то внести изменения можно с помощью

Таблица 4.8

Ограничения на определяемые данные

Оператор	Пояснение	На значения столбцов	На таблицу
NOT NULL	Не нулевой	+	
UNIQUE	Уникальный	+	+
PRIMARYKEY	Первичный ключ	+	+
CHECK	Проверка предиката	+	+
DEFAULT	Значение по умолчанию	+	
REFERENCES	Ссылка на имя таблицы, имя столбца	+	+
FOREIGNKEY	Внешний ключ		+

оператора ALTER TABLE. Этот оператор предназначен для одной и той же таблицы. Если после ее создания обнаружились неточности в ее определении, то внести изменения можно с помощью оператора ALTER TABLE. Этот оператор предназначен для изменения структуры существующей таблицы: можно удалить или добавить поле к существующей таблице.

Манипулирование данными. SQL позволяет пользователю или прикладной программе изменять содержимое базы данных путем вставки новых данных, удаления или модификации существующих данных.

Вставка новых данных является процедурой добавления строк в базу данных и выполняется с помощью оператора INSERT.

Модификация данных предполагает изменения значений в одном или нескольких столбцах таблицы и выполняется с помощью оператора UPDATE. Пример:

```
UPDATE СЧЕТ
SET сумма=сумма+1000.00
WHERE сумма>0
```

Удаление строк из таблицы осуществляется с помощью оператора DELETE. Синтаксис оператора имеет вид:

```
DELETE
FROM таблица
[WHERE условие]
```

Предложение WHERE не является обязательным, однако, если его не включить, то будут удалены все записи таблицы. Полезно использовать оператор SELECT с тем же синтаксисом, что и оператор DELETE, чтобы предварительно проверить, какие записи будут удалены.

Обеспечение целостности данных. Язык SQL позволяет определить достаточно сложные ограничения целостности, удовлетворение которым будет проверяться при всех модификациях базы данных. Контроль за результатами транзакций, обработка возникающих ошибок и координирование параллельной работы с базой данных нескольких приложений или пользователей обеспечивается операторами COMMIT (фиксирует удачное окончание текущей транзакции и начало новой) и ROLLBACK (необходимость отката – автоматического восстановления состояния базы данных на начало транзакции).

Выборка данных – одна из важнейших функций базы данных, которой соответствует оператор SELECT. Пример использования оператора был рассмотрен в предыдущем разделе.

В SQL можно создавать вложенные последовательности запросов (подзапросы). Существуют определенные типы запросов, которые лучше реализовывать с помощью подзапросов. К таким запросам относятся так называемые проверки существования. Предположим, что требуется получить данные о студентах, которые не имеют оценку «семь баллов». Если будет возвращено пустое множество, то это означает лишь одно – у каждого студента есть, по крайней мере, одна такая оценка.

Связывание таблиц. Операторы языка SQL позволяют извлекать данные более чем из одной таблицы. Одна из возможностей сделать это заключается в связывании таблиц по одному общему полю.

В операторе SELECT должно присутствовать ограничение на совпадение значений определенного столбца (поля). Тогда из связанных таблиц будут извлекаться только те строки, в которых значения заданного столбца совпадают. Название столбца указывается только вместе с названием таблицы; в противном случае оператор будет неоднозначным.

Можно использовать другие типы связывания таблиц: оператор INTER JOIN (внутреннее соединение) обеспечивает присутствие в результирующем наборе записей, совпадающие значения в связанных полях. Внешние соединения (OUTER JOIN) позволяют включить в результат запроса все строки из одной таблицы и соответствующие им строки из другой

Управление доступом. SQL обеспечивает синхронизацию обработки базы данных различными прикладными программами, защиту данных от несанкционированного доступа.

Доступ к данным в многопользовательской среде регулируется с помощью операторов GRANT и REVOKE. В каждом операторе необходимо указать пользователя, объект (таблицу, представление), по отношению к которому задаются полномочия, и сами полномочия. Например, оператор GRANT задает пользователю X возможность производить выборку данных из таблицы TOBAP:

```
GRANT SELECT ON TOBAP TO X
```

Оператор REVOKE аннулирует все предоставленные ранее полномочия.

Встраивание SQL в прикладные программы. Реальные приложения обычно написаны на других языках, генерирующих код на языке SQL и передающих их в СУБД в виде текста в формате ASCII.

Стандартом фирмы IBM для SQL-продуктов регламентировано использование встроенного языка SQL. При написании прикладной программы ее текст представляет собой смесь команд основного языка программирования (например, C, Pascal, Cobol, Fortran, Assembler) и команд SQL со специальным префиксом, например. ExecSQL. Структура SQL-предложений расширена для размещения переменных основного языка в SQL-конструкции.

SQL-процессор видоизменяет вид программы в соответствии с требованиями компилятора основного языка программирования.

Функция компилятора состоит в трансляции (перевод) программы с исходного языка программирования на язык, близкий к машинному. После компиляции прикладная программа (приложение) представляет собой самостоятельный модуль.

Диалекты языка SQL

В современных реляционных СУБД для описания и манипулирования данными используются диалекты языка SQL. Подмножество языка SQL, позволяющее создавать и описывать БД, называется DDL (Data Definition Language).

Первоначально язык SQL назывался SEQUEL(Structured English Query Language), потом SEQUEL/2, а затем просто – SQL. Сегодня язык SQL –фактический стандарт для реляционных СУБД.

Первый стандарт языка появился в 1989 г. – SQL-89 и поддерживался практически всеми коммерческими реляционными СУБД. Он имел общий характер и допускал широкое толкование. Достоинствами SQL-89 можно считать стандартизацию синтаксиса и семантики операторов выборки и манипулирования данными, а также фиксацию средств ограничения целостности базы данных. Однако в нем отсутствовал такой важный раздел как манипулирование схемой базы данных. Неполнота стандарта SQL-89 привела к появлению в 1992 г. следующей версии языка SQL.

SQL2 (или SQL-92) охватывает практически все необходимые проблемы: манипулирование схемой базы данных, управление транзакциями и сессиями, поддерживает архитектуры клиент-сервер или средства разработки приложений.

Дальнейшим шагом развития языка является вариант SQL 3. Эта версия языка дополняется механизмом триггеров, определением произвольного типа данных, объектным расширением.

В настоящее время существует три уровня языка: начальный, промежуточный и полный. Многие производители своих СУБД применяют собственные реализации SQL, основанные как минимум на начальном уровне соответствующего стандарта ANSI и содержащие некоторые расширения, специфические для той или иной СУБД. В табл. 4.9 приведены примеры диалектов SQL.

В объектно-ориентированных БД используется язык объектных запросов OQL (Object Query Language). За основу языка OQL была взята команда SELECT языка SQL2 и добавлена возможность

направлять запрос к объекту или коллекции объектов, а также возможность вызывать методы в рамках одного запроса.

Таблица 4.9

Диалекты языка SQL

СУБД	Язык запросов
СУБД System R	SQL
DB2	SQL
Access	SQL
SYBASE SQL Anywhere	Watcom-SQL
SYBASE SQL Server	Transact_SQL
My SQL	SQL
Oracle	PL/SQL

Совместимость многих используемых диалектов SQL обусловливает совместимость СУБД. Так, СУБД SYBASE SQL Anywhere максимально, насколько это возможно для СУБД такого класса, совмес тима с СУБД SYBASE SQL Server. Одной из сторон такой совместимости является поддержка в SYBASE SQL Anywhere такого диалекта языка SQL как Transact-SQL. Этот диалект используется в SYBASE SQL Server и может применяться в SYBASE SQL Anywhere наряду с собственным диалектом языка SQL – *Watcom-SQL*.

Контрольные вопросы

1. Как можно классифицировать СУБД?
2. Какие модели баз данных существуют?
3. Что является основными элементами инфологических моделей?
4. Какие типы связей между сущностями существуют?
5. Что такое ER-диаграммы и для чего они используются?
6. Что позволяет делать процедура нормализации таблиц?
7. Назовите языковые и программные средства СУБД?
8. К какому типу относится СУБД MS Access?
9. Назовите основные объекты СУБД MS Access?
10. Для чего используются основные операторы языка SQL?

5. ЗАЩИТА ИНФОРМАЦИИ ПРИ ИСПОЛЬЗОВАНИИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

5.1. Основы информационной безопасности

Интенсивное развитие компьютерных средств и информационных технологий повышают требования к обеспечению информационной безопасности.

Информационная безопасность определяется как защищенность информации и поддерживающей инфраструктуры от случайных или преднамеренных воздействий естественного или искусственного характера, которые могут нанести ущерб субъектам информационных отношений, в том числе владельцам и пользователям информации и поддерживающей инфраструктуре.

Средства и методы поддержки информационной безопасности должны обеспечивать:

– доступность, т. е. информация, ресурсы, сервисы, средства взаимодействия и связи должны быть доступны и готовы к работе всегда, когда возникает необходимость;

– целостность, т. е. сохранение структуры информации и/или ее содержания в процессе передачи и хранения. Целостность можно подразделить на статическую – неизменность информационных объектов, и динамическую – корректное выполнение транзакций. Средства контроля динамической целостности применяются в частности при анализе потока финансовых сообщений с целью выявления кражи, переупорядочения или дублирования отдельных сообщений;

– конфиденциальность, т. е. обеспечение доступа к информации только ограниченному кругу субъектов информацией системы (пользователям, процессам, программам).

Доступность информации (ресурсов ИС) предполагает что субъекты, имеющие права доступа, могут беспрепятственно их реализовывать.

Под *доступом к информации* понимается возможность, получения информации и ее использование (ознакомление, обработка, копирование, модификация или уничтожение). Различают санкционированный и несанкционированный доступ к информации.

Санкционированный доступ к информации – это доступ, не нарушающий установленные правила разграничения доступа.

Несанкционированный доступ характеризуется нарушением установленных правил разграничения доступа и является наиболее распространенным видом компьютерных нарушений.

Права доступа – совокупность правил, регламентирующих порядок и условия доступа субъекта к информации, ее носителям и другим ресурсам ИС, установленных правовыми документами или собственником, владельцем информации.

Разграничение доступа – с одной стороны, правила, ограничивающие действия субъектов ИС над ее ресурсами, с другой – деятельность по реализации этих правил.

Атака на информационную систему – это действие, предпринимаемое злоумышленником с целью поиска и использования той или иной уязвимости системы. Таким образом, атака – это реализация угрозы безопасности. Под *угрозой информационной безопасности* понимаются события или действия, которые могут привести к искажению, несанкционированному использованию или даже к разрушению информационных ресурсов управляемой системы, а также программных и аппаратных средств. Комплекс мер, направленных на обеспечение информационной безопасности, должен гарантировать защиту информации и минимизировать риски ее искажения.

Важнейшей составляющей процесса обеспечения информационной безопасности является проведение квалифицированного *аудита безопасности ИС*, что позволяет своевременно выявить существующие недостатки и объективно оценить соответствие обеспечения информационной безопасности требуемому уровню решаемых задач организации. Оценка качества безопасности ИС выполняется специализированными аудиторскими организациями.

Защита информации – деятельность, направленная на сохранение государственной, служебной, коммерческой или личной тайны, на сохранение носителей информации любого содержания.

Политика безопасности – это совокупность норм и правил, определяющих принятые в организации меры по обеспечению безопасности информации, связанной с деятельностью организации. Цель ее формулирования для ИС – изложение взглядов руководства организации на сущность угроз информационной безопасности. Политика безопасности должна быть оформлена документально на нескольких уровнях управления: высшее руководство подготав-

ливает и утверждает документ, в котором определены цели политики безопасности, структура и перечень решаемых задач и ответственные за реализацию политики; администраторами безопасности ИС детализируется документ с учетом принципов деятельности организации, важности целей и наличия ресурсов.

Политика безопасности обычно состоит из двух частей: общих принципов и конкретных правил работы с ИС для различных категорий пользователей.

В руководство по компьютерной безопасности, разработанное Национальным институтом стандартов и технологий США (National Institute of Standards and Technology – NIST), рекомендовано включать в описание политики безопасности следующие разделы:

1. Предмет политики, где определяются цели и указываются причины разработки политики, область ее применения, задачи, термины и определения.

2. Описание позиции организации, где описываются ресурсы ИС, перечень допущенных к ресурсам лиц и процессов, порядок получения доступа к ресурсам.

3. Применимость, где описывается порядок доступа к данным ИС, ограничения или технологические цепочки, применяемые при реализации политики безопасности.

4. Роли и обязанности, где указываются ответственные должностные лица и их обязанности в отношении разработки и внедрения элементов политики.

5. Соблюдение политики, где описываются права и обязанности пользователей ИС, недопустимые действия при осуществлении доступа к информационным ресурсам и меры наказания за нарушения режимных требований, технология фиксации фактов нарушения политики безопасности и применения административных мер воздействия к нарушителям.

5.2. Критерии оценки информационной безопасности

Первые исследования в области обеспечения безопасности данных в ИС были вызваны потребностями военной сферы, где проблема безопасности стоит особенно остро. Начало было положено исследованиями вопросов защиты компьютерной информации,

проведенными в конце 70-х начале 80-х гг. XX в. Национальным центром компьютерной безопасности Министерства обороны США. Результатом этих исследований явилась публикация в 1983 г. документа под названием «Критерии оценки надежных компьютерных систем», по цвету обложки получившего название *«Оранжевая книга»*. Этот документ стал первым стандартом в области создания защищенных компьютерных систем и впоследствии основой организации системы их сертификации по критериям защиты информации.

В 1999 г. Международная организация по стандартизации (ИСО) приняла стандарт под названием «Общие критерии оценки безопасности информационных технологий» (сокращенно – Common Criteria), который способствовал унификации национальных стандартов в области оценки безопасности информационных технологий на основе взаимного признания сертификатов. Этот документ содержит обобщенное и формализованное представление знаний и опыта, накопленного в области обеспечения информационной безопасности.

Стандарт определяет инструменты оценки безопасности ИТ и порядок их использования, ряд ключевых понятий, лежащих в основе концепции оценки защищенности продуктов ИТ: профиля защиты, задания по безопасности и объекта оценки.

Профиль защиты – документ, содержащий обобщенный стандартный набор функциональных требований и требований доверия для определенного класса продуктов или систем (например, профиль защиты может быть разработан на межсетевой экран корпоративного уровня, систему электронных платежей), описания угроз безопасности и задач защиты, обоснования соответствия между угрозами безопасности, задачами защиты и требованиями безопасности.

Задание по безопасности – документ, содержащий требования безопасности для конкретного объекта оценки и специфицирующий функции безопасности и меры доверия.

Под *объектом оценки* понимается произвольный продукт информационных технологий или вся ИС в целом (корпоративная информационная система) предприятия, процессы обработки данных, подготовки решений и выработки управляющих воздействий; программные коды, исполняемые вычислительными средствами в процессе функционирования корпоративной информационной системы; данные в базах данных; информация, выдаваемая потребителям и на исполнительные механизмы; коммуникационная

аппаратура и каналы связи; периферийные устройства коллективного пользования; помещения и др.).

В данном стандарте представлены две категории требований безопасности:

– функциональные, которые определяют совокупность функций объекта оценки, обеспечивающих его безопасность;

– требования адекватности, т. е. свойство объекта оценки, дающее определенную степень уверенности в том, что механизмы его безопасности достаточно эффективны и правильно реализовано.

Безопасность в данном стандарте рассматривается не статично, а в привязке к жизненному циклу объекта. Использование стандарта позволяет:

– сравнивать между собой результаты различных сертификационных испытаний ИС и контролировать качество оценки безопасности;

– единообразно использовать имеющиеся результаты и методики оценок различных стран;

– определять общий набор понятий, структур данных и язык для формулирования вопросов и утверждений относительно информационной безопасности;

– потенциальным пользователям ИС, опираясь на результаты сертификации, определять, удовлетворяет ли данный программный продукт или система их требованиям безопасности;

– постоянно улучшать существующие критерии, вводя новые концепции и уточняя содержание имеющихся критериев.

В разных странах дополнительно разработаны отраслевые стандарты, нормативные документы и спецификации по обеспечению информационной безопасности, которые применяются национальными организациями при разработке программных средств, ИС и обеспечения качества и безопасности их функционирования.

5.3. Классы безопасности информационных систем

В соответствии с «Оранжевой книгой» политика безопасности должна включать в себя следующие элементы:

– произвольное управление доступом – метод разграничения доступа к объектам, основанный на учете личности субъекта (группы, в которую он входит). Некоторое лицо (владелец объекта) может по своему усмотрению предоставлять другим субъектам или отбирать у них права доступа к объекту;

– безопасность повторного использования объектов – дополнительные средства, предохраняющие от случайного или преднамеренного извлечения конфиденциальной информации из оперативной памяти, дисковых блоков и магнитных носителей в целом;

– метки безопасности, состоящие из уровня секретности и списка категорий;

– принудительное управление доступом основано на сопоставлении меток безопасности субъекта и объекта: метка субъекта описывает его благонадежность, метка объекта – степень конфиденциальности содержащейся в нем информации. После фиксации меток безопасности субъектов и объектов оказываются зафиксированными и права доступа.

В «Оранжевой книге» дано определение *безопасной системы* – это система, которая посредством специальных механизмов защиты контролирует доступ к информации таким образом, что только имеющие соответствующие полномочия лица или процессы, выполняющиеся от их имени, могут получить доступ на чтение, запись, создание или удаление информации. В ней выделены основные классы защищенности – *D*, *C*, *B*, *A*.

В *класс D* попадают системы, оценка которых выявила их несоответствие требованиям всех других классов.

Класс C1: ИС должна управлять доступом именованных пользователей к именованным объектам; пользователи должны идентифицировать себя до выполнения каких-либо контролируемых ИС действий; ИС должна быть защищена от внешних воздействий и от попыток слежения за ходом работы; должна обеспечиваться корректность функционирования аппаратных и программных средств путем периодической проверки; должен быть описан подход к безопасности, используемый разработчиком, и применение его при реализации ИС.

Класс C2 (в дополнение к требованиям класса C1): все объекты должны подвергаться контролю доступа; каждый пользователь системы должен уникальным образом идентифицироваться; каждое регистрируемое действие должно ассоциироваться с конкретным пользователем; предусмотрена ликвидация всех следов внутреннего использования объектов ИС; ИС должна создавать, поддерживать и защищать журнал регистрационной информации, относящейся к доступу к объектам, контролируемым ИС; тестирование

должно подтвердить отсутствие очевидных недостатков в механизмах изоляции ресурсов и защиты регистрационной информации.

Класс B1 (в дополнение к требованиям класса C2): каждый хранимый объект ИС должен иметь отдельную идентификационную метку; ИС должна обеспечить реализацию принудительного управления доступом к хранимым объектам, взаимную изоляцию процессов путем разделения их адресных пространств; должна существовать неформальная или формальная модель политики безопасности.

Класс B2 (в дополнение к требованиям класса B1): должна быть предусмотрена возможность регистрации событий, связанных с организацией тайных каналов обмена информацией; ИС должна быть внутренне структурирована и демонстрировать устойчивость к попыткам проникновения; тесты должны подтверждать действенность мер по уменьшению пропускной способности тайных каналов передачи информации.

Класс B3 (в дополнение к требованиям класса B2): для управления доступом должны использоваться списки управления доступом с указанием разрешенных режимов; должна быть предусмотрена возможность регистрации появления или накопления событий, несущих угрозу политике информационной безопасности; администратор безопасности должен извещаться о попытках нарушения политики безопасности, а система в случае продолжения попыток должна пресекать их наименее болезненным способом; должны существовать процедуры и/или механизмы, позволяющие произвести восстановление после сбоя или иного нарушения работы без ослабления защиты; должна быть продемонстрирована устойчивость ИС к попыткам проникновения.

Класс A1 (в дополнение к требованиям класса B3): тестирование должно продемонстрировать, что реализация ИС соответствует формальным спецификациям; механизм управления информационной безопасностью должен распространяться на весь жизненный цикл и все компоненты системы, имеющие отношение к обеспечению безопасности.

5.4. Угрозы информационной безопасности

Угрозы информационной безопасности делятся на два типа – естественные и искусственные. *Естественные угрозы* обусловливаются природными факторами (наводнения, землетрясения и другие

стихийные бедствия), последствиями техногенных катастроф (пожары, взрывы и др.). Чаще всего ИС страдают от *искусственных угроз* (преднамеренных). Знание возможных угроз и уязвимых мест информационной системы необходимо для того, чтобы выбрать наиболее эффективные средства обеспечения безопасности.

Угроза характеризуется следующими параметрами: источник угрозы, метод воздействия, уязвимые места, которые могут быть использованы, ресурсы (активы), которые могут пострадать.

Источники угроз безопасности могут находиться как внутри информационной системы (внутренние), так и вне ее (внешние). Для одной и той же угрозы (например, кражи) методы противодействия для внешних и внутренних источников будут разными.

Самыми частыми и опасными (с точки зрения размера ущерба) являются непреднамеренные ошибки пользователей, операторов, системных администраторов и других лиц, обслуживающих ИС. Иногда такие ошибки приводят к прямому ущербу (неправильно введенные данные, ошибка в программе, вызвавшая остановку или разрушение системы) и/или созданию «слабых» мест, которыми могут воспользоваться злоумышленники. Согласно данным Национального института стандартов и технологий США 55 % случаев нарушения безопасности информационной системы являются следствием непреднамеренных ошибок (рис. 5.1). Работа в сети Интернет делает этот фактор достаточно актуальным, причем источником ущерба могут быть действия как отдельных пользователей и организаций, так и технологии сети Интернет, что особенно опасно.

Рис. 5.1. Источники угроз информационной безопасности

Большой ущерб наносят кражи и подлоги, где в большинстве расследованных случаев виновниками оказывались штатные сотрудники организаций, отлично знакомые с режимом работы и защитными мерами.

Преднамеренные попытки получения несанкционированного доступа через внешние коммуникации занимают в настоящее время около 10 % всех возможных нарушений. В сети Интернет почти каждый интернет-сервер по нескольку раз в день подвергается попыткам проникновения.

Угрозы информационной безопасности можно разделить:

– на конструктивные (основной целью несанкционированного доступа является получение копии конфиденциальной информации);

– деструктивные (несанкционированный доступ приводит к потере или изменению данных или прекращению сервиса).

В общем случае источники угроз определить нелегко. Они могут варьироваться от неавторизованных вторжений злоумышленников до компьютерных вирусов.

Классификация угроз информационной безопасности приведена в табл. 5.1.

Утечка конфиденциальной информации – это бесконтрольный выход конфиденциальной информации за пределы ИС или круга лиц, которым она была доверена по службе. Пути утечки конфиденциальной информации: разглашение, уход информации по различным техническим каналам, несанкционированный доступ.

Появляется множество вредоносных программ, что не позволяет разработать постоянные и надежные средства защиты от них. Разновидности вредоносных программ (компьютерных вирусов):

– по виду среды обитания: файловые вирусы, загрузочные вирусы, файлово-загрузочные вирусы;

– по способу запуска на выполнении: нерезидентные вирусы, резидентные вирусы;

– по способу маскировки: немаскирующиеся вирусы, маскирующиеся вирусы (самошифрующиеся, невидимые и мутирующие);

– по способу распространения: троянские программы, программы-репликаторы («черви») и захватчик паролей.

Таблица 5.1

Классификация угроз информационной безопасности

Признак классификации угроз	Угроза
Доступность	С использованием доступа, скрытых каналов
Способ воздействия	Непосредственное воздействие на объект атаки; воздействие на систему разрешений; опосредованное воздействие
Использование средств атаки	Использование штатного программного обеспечения (ПО), разработанного ПО
Территориальный	Глобальные, региональные, локальные
Объект, на который нацелена угроза	Данные, программы, аппаратура, поддерживающая инфраструктура
Способ осуществления	Случайные либо преднамеренные действия природного или техногенного характера
Расположение источника угроз	Внутри или вне ИС

Источниками угроз могут выступать:

– сама ИС (нарушение информационного обслуживания, т. е. задержка предоставления информационных ресурсов абоненту, вызывающая нерациональные действия пользователя);

– пользователи при незаконном захвате привилегий.

На угрозы информационной безобасности влияют различные *факторы*:

– политические: изменение геополитической обстановки, информационная экспансия, изменение политической системы, системы управления, нарушение информационных связей в результате образования новых государств, стремление стран к более тесному сотрудничеству, низкая общая правовая и информационная культура в обществе;

– экономические: переход к рыночной экономике, критическое состояние отраслей промышленности, расширяющаяся кооперация с зарубежными странами;

– организационно-технические: недостаточная нормативно-правовая база в сфере информационных отношений, рост объемов информации, передаваемой по каналам связи, обострение криминогенной обстановки и др.

В последнее время широкое распространение получила *компьютерная преступность* – любые незаконные, неправомерные, неэтичные действия, связанные с автоматической обработкой и передачей данных. Существенными причинами активизации компьютерных преступлений являются:

– переход от традиционной «бумажной» технологии хранения и передачи сведений на электронную при недостаточном развитии технологий защиты информации;

– объединение вычислительных систем, создание глобальных сетей и расширение доступа к информационным ресурсам;

– усложнение программных средств и связанное с этим уменьшение их надежности и увеличение числа уязвимых мест.

Превращение компьютерной преступности в мировое явление потребовало международного сотрудничества и совместного противодействия компьютерным преступникам. В этих целях совершенствуется правовая база, в частности, вслед за европейскими странами в рамках СНГ заключаются межгосударственные договоры и соглашения, направленные на борьбу с компьютерной преступностью. Страны СНГ обязуются сотрудничать в целях обеспечения эффективного предупреждения, выявления, пресечения расследования и раскрытия преступлений в сфере компьютерной информации, обеспечивать гармонизацию национального законо-

дательства в области борьбы с преступлениями в сфере компьютерной информации. Преступление в сфере компьютерной информации – уголовно наказуемое деяние, предметом посягательства которого является компьютерная информация.

По соглашению стран СНГ в качестве уголовно наказуемых признаются совершаемые умышленно действия:

– неправомерный доступ к охраняемой законом компьютерной информации, если это повлекло уничтожение, блокирование, модификацию, копирование информации, нарушение работы компьютера или сети;

– создание, использование или распространение вредоносных программ;

– нарушение правил эксплуатации компьютера или сети имеющим доступ лицом, повлекшее уничтожение, блокирование или модификацию охраняемой законом информации и причинение существенного вреда или тяжкие последствия;

– незаконное использование компьютерных программ и баз данных, являющихся объектами авторского права, присвоение авторства, если это причинило существенный ущерб.

Сотрудничество осуществляется в формах обмена информацией; запросов о проведении оперативно-розыскных мероприятий; планирования и проведения скоординированных мероприятий и операций по предупреждению, выявлению, пресечению, раскрытия и расследования преступлений в сфере компьютерной информации; создания информационных систем, обеспечивающих выполнение задач по предупреждению, выявлению, пресечению, раскрытию и расследованию преступлений в сфере компьютерной информации; проведения совместных научных исследований по представляющим взаимный интерес проблемам борьбы с преступлениями в сфере компьютерной информации; обмена нормативными правовыми актами, научно-технической литературой по борьбе с преступлениями в сфере компьютерной информации и др.

5.5. Методы и средства защиты информации

Выделяют два подхода к обеспечению информационной безопасности:

– *фрагментарный*, который направлен на противодействие четко определенным угрозам в заданных условиях (например,

средства управления доступом, автономные средства шифрования, специализированные антивирусные программы и т. п.). Его достоинством является высокая избирательность к конкретной угрозе. Существенным недостатком является отсутствие единой защищенной среды обработки информации, небольшое видоизменение угрозы ведет к потере эффективности защиты;

– *комплексный*, который ориентирован на создание защищенной среды обработки информации, объединяющей в единый комплекс разнородные меры противодействия угрозам, что позволяет гарантировать определенный уровень безопасности и является несомненным достоинством комплексного подхода. К недостаткам этого подхода относят: ограничения на свободу действий пользователей, чувствительность к ошибкам установки и настройки средств защиты, сложность управления. Данный подход использует большинство государственных и крупных коммерческих предприятий и учреждений.

Защиту информации следует рассматривать как регулярный процесс, осуществляемый путем комплексного использования технических, программных средств и организационных мероприятий на всех этапах разработки, испытаний и эксплуатации ИС. Требования по защите, предъявляемые к ИС, должны рассматриваться как часть общих функциональных требований к ней. В мировой практике используется понятие *комплексная система защиты* – совокупность законодательных, организационных и технических мер, направленных на выявление, отражение и ликвидацию различных видов угроз безопасности.

Комплексная информационная безопасность – такое состояние условий функционирования человека, объектов, технических средств и систем, при котором они надежно защищены от всех возможных видов угроз в ходе непрерывного процесса подготовки, хранения, передачи и обработки информации.

Корпоративные проекты информационной безопасности разрабатываются при объединении различных ИС и их компонент, подсистем связи, подсистем обеспечения безопасности в единую информационную систему с общими техническими средствами, каналами связи, ПО и базами данных, что предполагает обязательную непрерывность процесса обеспечения безопасности как во времени (в течение всей жизни ИС), так и в пространстве

(по всему технологическому циклу деятельности) с обязательным учетом всех возможных видов угроз.

По какой бы технологии не строилась комплексная система информационной безопасности, требуется решение ряда сложных разноплановых частных задач в их тесной взаимосвязи (принцип системности и комплексности). Наиболее очевидными из них являются задачи разграничения доступа к информации, ее технического и криптографического «закрытия», устранение «паразитных» излучений технических средств, технической и физической укрепленности объектов, охраны и оснащения их тревожной сигнализацией. Стандартный набор средств защиты информации в составе современной ИС обычно содержит средства, реализующие методы программно-технической защиты информации.

Современные комплексные системы защиты осуществляют полный спектр управления всеми процессами, происходящими в ИС. Они позволяют:

– собирать информацию со всех устройств идентификации и контроля, обрабатывать ее и управлять исполнительными устройствами;

– собирать и обрабатывать информацию с оборудования охранных систем сигнализации, систем видеонаблюдения, пожаротушения, вентиляции, энергосбережения и др.;

– создавать журналы учета состояния этих систем и происхождения изменений, демонстрировать оператору состояние систем и аварийные ситуации;

– контролировать состояние всей структуры в режиме реального времени при подключении информационных каналов, связывающих главный объект с филиалами или другими объектами.

Для обеспечения информационной безопасности используются следующие *методы:* законодательные (законы, нормативные акты, стандарты и т. п.); административно-организационные (действия общего характера, предпринимаемые руководством организации, и конкретные меры безопасности, направленные на работу с людьми); программно-технические.

К *законодательным методам* относят комплекс мер, направленных на создание и поддержание в обществе негативного (в том числе карательного) отношения к нарушениям и нарушителям информационной безопасности.

Административно-организационные методы. Администрация организации должна осознавать необходимость поддержания режима безопасности и выделять на эти цели соответствующие ресурсы; основой защиты является политика безопасности и комплекс организационных мер (управление персоналом, физическая защита, поддержание работоспособности, реагирование на нарушения режима безопасности, планирование восстановительных работ). В любой организации должен существовать набор регламентов, определяющих действия персонала в соответствующих ситуациях.

Программно-технические методы и средства:

– защищенные виртуальные частные сети для защиты информации, передаваемой по открытым каналам связи;

– межсетевые экраны для защиты корпоративной сети от внешних угроз при подключении к общедоступным сетям связи;

– управление доступом на уровне пользователей и защита от несанкционированного доступа к информации;

– гарантированная идентификация пользователей путем применения токенов (смарт-карты, touch-memory, ключи для USB-портов и т. п.) и других средств аутентификации;

– защита информации на файловом уровне (шифрование файлов и каталогов) для обеспечения ее надежного хранения;

– защита от вирусов с использованием специализированных комплексов антивирусной профилактики и защиты;

– обнаружение вторжений и активного исследования защищенности информационных ресурсов;

– криптографическое преобразование данных для обеспечения целостности, подлинности и конфиденциальности информации.

В настоящее время для организации защищенных VPN-каналов широко используется комплекс стандартов сети Интернет – IPSec (IP Security), поддержка которого является обязательным условием для перспективных VPN-продуктов. Средства VPN предприятия могут эффективно поддерживать защищенные каналы трех типов: с удаленными и мобильными сотрудниками (защищенный удаленный доступ), сетями филиалов предприятий (защита интранет), сетями предприятий-партнеров (защита экстранет).

Для защиты VPN применяются межсетевые экраны, которые реализуют следующую схему доступа:

– доступ контролируется в одной точке, располагающейся на пути соединения внутренней сети с сетью Интернет или другой публичной сетью, являющейся источником потенциальных угроз;

– все субъекты доступа делятся на группы по IP-адресам (внутренние и внешние пользователи); внешним пользователям разрешается для доступа к внутренним ресурсам сети использовать один-два сервиса, например электронную почту, а трафик остальных сервисов отсекается.

Применение нескольких межсетевых экранов в пределах одной внутренней сети требует организации их скоординированной работы на основе единой политики доступа, что позволяет корректно обрабатывать пакеты пользователей независимо от того, через какую точку доступа проходит их маршрут.

При предоставлении информации в сети для гарантированной идентификации пользователей необходим специальный механизм, состоящий из следующих *процедур:*

– идентификация – распознавание пользователя по его идентификатору (имени), который пользователь сообщает сети по запросу, сеть проверяет его наличие в своей базе данных;

– аутентификация – проверка подлинности заявленного пользователя, которая позволяет достоверно убедиться, что пользователь именно тот, кем себя объявляет (пароль);

– авторизация – процедура предоставления пользователю определенных полномочий и ресурсов сети.

Эффективным средством повышения надежности защиты данных на основе гарантированной идентификации пользователя являются *электронные токены,* которые хранят персональные данные пользователя системы.

Антивирусная защита должна устанавливаться в узлах, на которых информация хранится, обрабатывается и передается в открытом виде.

Постоянные изменения ИС (реконфигурация программных средств, подключение новых рабочих станций и т. п.) могут привести к появлению новых угроз и уязвимых мест в системе защиты. В связи с этим особенно важно своевременное их выявление и внесение изменений в соответствующие настройки системы информационной безопасности, для чего используются *средства обнаружения вторжений*, которые дополняют защитные функции межсетевых экранов. Межсетевые экраны пытаются отсечь потенциально опасный трафик и не пропустить его в защищаемые сегменты, в то время как средства обнаружения вторжений анализируют результирующий трафик в защищаемых сегментах и выявляют атаки

на ресурсы сети или потенциально опасные действия и могут использоваться в незащищенных сегментах, например перед межсетевым экраном, для получения общей картины об атаках, которым подвергается сеть извне.

Особую роль в программно-технических методах защиты информации играют криптографические преобразования данных и электронная цифровая подпись.

Криптографический алгоритм или шифр – это математическая формула, описывающая процессы зашифрования и расшифрования. Для того чтобы зашифровать открытый текст, криптоалгоритм работает в сочетании с ключом – словом, числом или фразой. Одно и то же сообщение одним алгоритмом, но с разными ключами будет преобразовываться в разный *шифротекст.* Защищенность шифротекста целиком зависит от стойкости криптоалгоритма и секретности ключа.

В традиционной криптографии один и тот же ключ используется как для зашифрования, так и для расшифрования данных (рис. 5.2). Такой ключ называется *симметричным* ключом (закрытым). Data Encryption Standart (DES) – пример симметричного алгоритма, широко применявшегося на Западе с 70-х гг. XX в. в банковской и коммерческой сферах. Алгоритм шифрования был реализован в виде интегральной схемы с длиной ключа в 64 бита. В настоящее время стандарт DES сменяется стандартом Advanced Encryption Standard (AES), где длина ключа составляет до 256 бит.

Рис. 5.2. Этапы шифрования с симметричным ключом

Симметричное шифрование обеспечивает скорость выполнения криптографических операций, но имеет два существенных недостатка, во-первых, большое количество необходимых ключей (каждому пользователю отдельный ключ); во-вторых, сложности передачи закрытого ключа.

Для установления шифрованной связи с помощью симметричного алгоритма отправителю и получателю нужно предварительно согласовать ключ и держать его в тайне. Если они находятся в географически удаленных местах, то должны прибегнуть к помощи доверенного посредника,

чтобы избежать компрометации ключа в период транспортировки. Злоумышленник, перехвативший ключ, сможет читать, изменять и подделывать любую информацию, зашифрованную или заверенную этим ключом.

Проблема управления ключами была решена криптографией с открытым ключом, или асимметричным, концепция которой была предложена в 1975 г. В этой схеме применяется пара ключей; открытый, который зашифровывает данные, и соответствующий ему закрытый – их расшифровывает. Тот, кто зашифровывает данные, распространяет свой открытый ключ по всему свету, в то время как закрытый держит в тайне. Любой человек с копией открытого ключа может зашифровать данные, но прочитать данные сможет только тот, у кого есть закрытый ключ (рис. 5.3).

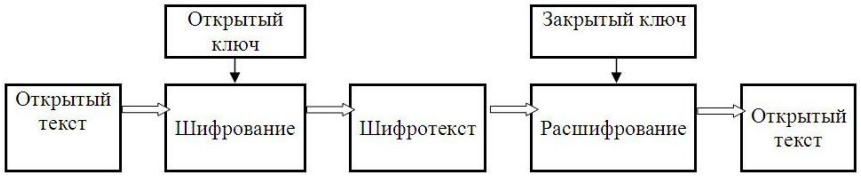


Рис. 5.3. Этапы шифрования с асимметричным ключом

Хотя открытый и закрытый ключ математически связаны, однако вычисление закрытого ключа из открытого практически невыполнимо.

Асимметричное шифрование позволяет людям, не имеющим договоренности о безопасности, обмениваться секретными сообщениями. Необходимость отправителю и получателю согласовывать тайный ключ по специальному защищенному каналу полностью отпала. Все коммуникации затрагивают только открытые ключи, тогда как закрытые хранятся в безопасности. Примерами криптосистем с открытым ключом являются *Elgamal*, *RSA*, *Diffie-Hellman*, *DSA* и др.

Использование криптосистем с открытым ключом предоставляет возможность создания *электронной цифровой подписи* (ЭЦП). ЭЦП – это реквизит электронного документа, предназначенный для удостоверения источника данных и защиты электронного документа от подделки. Цифровая подпись позволяет получателю сообщения убедиться в аутентичности источника информации (в том, кто является автором информации), проверить, была ли информация изменена (искажена), пока находилась в пути. Таким образом, цифровая подпись является средством аутентификации и контроля целостности данных и служит той же цели, что печать или

собственноручный автограф на бумажном листе. Сравнительные характеристики цифровой и обычной подписей приведены в табл. 5.3.

Таблица 5.3

Сравнительные характеристики цифровой и обычной подписей

Обычная подпись	Цифровая подпись
Каждая личность использует индивидуальные характеристики – почерк, давление на ручку и т. д.	Каждая личность использует для подписи свой уникальный секретный ключ
Попытка подделки подписи обнаруживается с помощью графологической экспертизы	Любая попытка подписать документ без знания соответствующего ключа не имеет успеха
Подпись и подписываемый документ передаются только вместе на одном листе бумаги	Цифровая подпись документа есть функция от содержания этого документа и секретного ключа
Переделать подпись отдельно от документа нельзя	Цифровая подпись может передаваться отдельно от документа
Подпись не зависит от содержания документа, на котором оно поставлена	Копия цифрового подписанного документа не отличается от его оригинала
Копии подписанного документа недействительны, если каждая из этих копий не имеет своей оригинальной, а не скопированной подписи	

Простой способ генерации цифровых подписей показан на рис. 5.4.

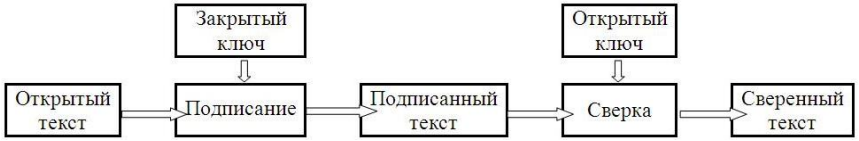


Рис. 5.4. Этапы применения цифровой подписи

Вместо шифрования информации открытым ключом информация шифруется собственным закрытым с одновременной генерацией открытого ключа. Если информация может быть расшифрована открытым ключом автора документа, то этим подтверждается авторство. В противном случае подпись считается поддельной

Для того чтобы не зашифровывать весь текст и затем пересылать его в зашифрованном виде, при формировании ЭЦП используется новый компонент – *односторонняя хэш-функция*, которая выбирает фрагмент произвольной длины, называемый прообразом (сообщение любого размера) и генерирует строго зависящий от прообраза код фиксированной длины. Хэш-функция гарантирует, что если информация будет каким-либо образом изменена, то в результате получится совершенно иное хэш-значение (дайджест сообщения). Полученный дайджест зашифровывается закрытым ключом отправителя и представляет собой электронную подпись, которая может прикрепляться к документу и передаваться вместе с исходным сообщением или же передаваться отдельно от него. При получении сообщения заново вычисляется дайджест подписанных данных, расшифровывается ЭЦП открытым ключом отправителя, тем самым сверяется целостность данных и их источник. Если вычисленный и полученный с сообщением дайджесты совпадают, то информация после подписания не была изменена.

Если в процессе формирования ЭЦП применяется стойкая односторонняя хэш-функция, то нет никакого способа взять чью-либо подпись с одного документа и прикрепить ее к другому или же любым образом изменить подписанное сообщение. Малейшее изменение в подписанном документе будет обнаружено в процессе сверки ЭЦП (рис. 5.5).

Цифровой сертификат ключа – это информация, прикрепленная к открытому ключу пользователя и дающая возможность другим установить, является ли ключ подлинным и верным. Цифровые

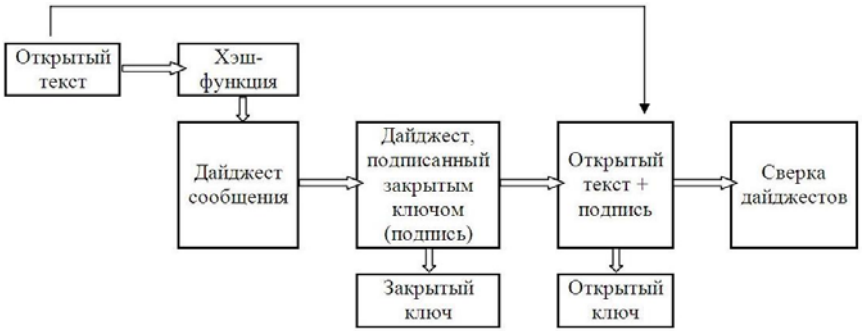


Рис. 5.5. Детализация процесса применения цифровой подписи

сертификаты ключей упрощают задачу определения принадлежности открытых ключей предполагаемым владельцам и аналогичны физическому сертификату (паспорту, водительскому удостоверению и пр.). Они нужны для того, чтобы сделать невозможной попытку выдать ключ одного человека за ключ другого. ЭЦП на сертификате указывает, что сведения сертификата были заверены доверенным третьим лицом или организацией.

Система сертификации может реализовываться в виде хранилища-депозитария, называемого сервером сертификатов (сервером-депозитарием открытых ключей), или инфраструктурой открытых ключей, предполагающего дополнительные возможности администрирования ключей.

Сервер-депозитарий – это сетевая база данных, санкционирующая пользователей на включение и извлечение из нее цифровых сертификатов. Он может выполнять некоторые функции администрирования, необходимые организации для поддержания политики безопасности, например, хранить только ключи, удовлетворяющие определенным критериям.

В настоящее время создаются центры сертификации, которые издают цифровые сертификаты и подписывают их своим закрытым ключом. Используя открытый ключ центра сертификации, любой пользователь, желающий проверить подлинность конкретного сертификата, сверяет подпись центра сертификации и удостоверяется в целостности содержащейся в сертификате информации и, что более важно, во взаимосвязанности сведений сертификата и открытого ключа.

По мнению многих специалистов, будущее системы защиты – это централизованное управление и единственные «точки входа» для пользователей. В таких централизованных системах администратор может управлять доступом и проверкой полномочий из одного пункта, а сервер санкционирования или единый сервер паролей должен содержать не только БД паролей, но и правила ограничения прав доступа.

5.6. Правовые аспекты информационной безопасности

Правовое обеспечение безопасности ИС – совокупность законодательных и морально-этических средств, регламентирующих правила и нормы поведения при обработке и использовании информации, информационных ресурсов и ИС.

Законодательные средства – законодательные акты страны, которыми регламентируются правила использования и обработки информации ограниченного доступа и устанавливаются меры ответственности за нарушение этих правил (гражданское, административное, уголовное право).

Морально-этические средства – всевозможные нормы, которые сложились традиционно или складываются по мере распространения вычислительных средств в данной стране или обществе. Они большей частью не являются обязательными, однако несоблюдение их обычно ведет к потере авторитета, престижа человека или группы лиц.

В соответствии с нормами законодательства Республики Беларусь физические и юридические лица имеют равные права на разработку и производство ИС, технологий и средств их обеспечения. Право каждого гражданина на свободный поиск, получение, передачу, производство и распространение информации – одно из основных прав гражданина, закрепленных Конституцией Республики Беларусь. Существует информация, набор сведений, распространение которых может наносить ущерб предприятиям, учреждениям, организациям или государству в целом. Один из правовых способов взаимного согласования и увязки этих основополагающих прав – установление различного правового режима для различных видов информации, информационных ресурсов и ИС.

В соответствии с нормами белорусского законодательства документированная информация может быть двух видов: открытая (общедоступная) и информация ограниченного доступа.

Документированная информация с ограниченным доступом делится на две категории: *государственные секреты* (защищаемые государством сведения в соответствии с законом «О защите государственных секретов») и *конфиденциальная информация,* представляющая собой документированную информацию, доступ к которой ограничен в соответствии с законодательством Республики Беларусь. Чаще других при разработке ИС появляется необходимость обработки двух категорий конфиденциальной информации, а именно персональных данных и коммерческой тайны.

Персональные данные – сведения о фактах, событиях и обстоятельствах жизни гражданина, позволяющие идентифицировать его личность. В соответствии с Законом «Об информации, информатизации и защите информации» персональные данные отнесены к категории конфиденциальной информации. В Республике Беларусь, как и в других странах мира, осуществляется правовое регулирование сбора, обработки, хранения, распространения и использования персональных данных.

Информация составляет служебную или коммерческую тайну, если она имеет действительную или потенциальную коммерческую ценность в силу неизвестности ее третьим лицам, к ней нет свободного доступа на законных основаниях собственник или обладатель информации принимает меры к охране ее конфиденциальности.

ИС могут находиться в собственности физических и юридических лиц, государства. Собственник ИС может использовать ее самостоятельно или передать право владения и пользования другим лицам. Он устанавливает порядок предоставления информации, условия доступа пользователей к ней, условия распространения документов, обеспечивает защиту информации, осуществляет необходимое лицензирование и сертификацию ИС в соответствии с нормами законодательства Республики Беларусь.

Различные компоненты (чаще всего программные продукты и базы данных) и в целом ИС могут являться объектами авторского права. Это происходит в тех случаях, когда ИС являются результатом творческого труда и представлены в тех формах, которые признаются в стране в качестве объектов авторского права.

Во многих странах несанкционированное копирование программ в целях продажи или бесплатного распространения рассматривается как государственное преступление, карается штрафом или тюремным заключением. Правовые методы защиты программных

продуктов также включают: патентную защиту, присвоение статуса производственных секретов, лицензионные соглашения и контракты.

Лицензионные соглашения распространяются на все аспекты правовой охраны программных продуктов, включая авторское право, патентную защиту, производственные секреты. Наиболее часто используются лицензионные соглашения на передачу авторских прав. *Лицензия* – договор на передачу одним лицом (лицензиаром) другому лицу (лицензиату) права на использование имени, продукции, технологии или услуги. *Лицензиар* увеличивает свои доходы сбором лицензионных платежей, расширяет область распространения программы; *лицензиат* извлекает доходы за счет их применения. В лицензионном соглашении оговариваются все условия эксплуатации программных продуктов, в том числе создание копий. *Автором* программных продуктов признается физическое лицо, в результате творческой деятельности которого они созданы. Автору, независимо от его имущественных прав, принадлежат личные авторские права: авторство, имя, неприкосновенность (целостность) программ. Программные продукты могут использоваться третьими лицами – *пользователями* на основании договора с правообладателем.

Первый кодекс компьютерной этики IEEE Code of Ethics был разработан в Институте инженеров электроники и электротехники в 1979 г. Затем этические кодексы были приняты Ассоциацией разработчиков компьютерных технологий (ACM Code of ethics and professional conduct), Ассоциацией профессионалов информационных технологий (Code of ethics of the Association of Information Technology Professionals), международной гильдией программистов (IPG Code of Ethics). В настоящее время их насчитывается уже не один десяток, некоторые содержат различные «заповеди» для работающих в сети, другие формулируют только основные принципы самодисциплины. Компьютерная этика устанавливает единые, основанные на этических представлениях принципы деятельности в сети Интернет, распространив их и на компьютерных профессионалов, и на пользователей сетей. Международная федерация по информационным технологиям (IFIP) рекомендовала принять кодексы компьютерной этики национальным организациям с учетом местных культурных и этических традиций.

Во всех кодексах содержатся нормы, основанные на соблюдении четырех главных моральных принципов: privacy (тайна частной

жизни), accuracy (точность), property (частная собственность) и accessibility (доступность). Модель компьютерной этики, основанная на этих принципах, получила название PAPA по первым буквам слов, составляющих сущность модели.

Принцип *privacy* выражает право человека на автономию и свободу в частной жизни, право на защиту от вторжения в нее органов власти и других людей. Соблюдение его особенно важно в связи с созданием автоматизированных банков Данных, содержащих различные сведения о личности.

Точное соблюдение инструкций по эксплуатации систем и обработке информации, честное и социально-ответственное отношение к своим обязанностям предполагают нормы, основанные на принципе *accuracy.*

Принцип *property* означает неприкосновенность частной собственности и является основой имущественного порядка в экономике. Следование этому принципу означает соблюдение права собственности на информацию и норм авторского права.

Принцип *accessibility* – один из главных принципов информационного общества, определяет право граждан на информацию и предполагает доступ каждого субъекта общества к информационным технологиям и к любой, необходимой для него информации, разрешенной для доступа.

Нормативно-правовая база, регламентирующая права, обязанности и ответственность субъектов, действующих в информационной сфере в Республике Беларусь, развивается по следующим направлениям:

– разработка концепции информационной безопасности;

– разработка нормативно-правовых, организационно-методических документов, регламентирующих деятельность органов государственной власти в области обеспечения информационной безопасности;

– разработка правовых и организационных мероприятий, обеспечивающих сохранение и развитие информационных ресурсов;

– формирование правового статуса субъектов системы информационной безопасности, определение их ответственности за обеспечение информационной безопасности.

Правовое обеспечение информационной безопасности в Республике Беларусь включает в себя:

1. Нормотворческую деятельность по созданию законодательства, регулирующего общественные отношения в области информационной безопасности.

2. Исполнительную и правоприменительную деятельность по исполнению законодательства в области информации, информатизации, защиты информации органами государственной власти и управления, организациями (юридическими лицами), гражданами.

Нормотворческая деятельность:

1. Оценка состояния действующего законодательства и разработка программы его совершенствования.

2. Создание организационно правовых механизмов обеспечения информационной безопасности.

3. Формирование правового статуса всех субъектов в системе информационной безопасности и определение их ответственности за обеспечение информационной безопасности.

4. Разработка организационно правового механизма сбора и анализа статистических данных о воздействии угроз информационной безопасности и их последствиях с учетом всех категорий информации.

5. Разработка законодательных и других нормативных актов, регулирующих порядок ликвидации последствий воздействий угроз; восстановление права и ресурсов; реализация компенсационных мер.

Исполнительная и правоприменительная деятельность:

1. Разработка процедур применения законодательства и нормативных актов к субъектам, совершившим преступления и проступки при работе с закрытой информацией.

2. Разработка составов правонарушений с учетом специфики уголовной, гражданской, административной и дисциплинарной ответственности.

Деятельность по правовому обеспечению информационной безопасности строится на трех фундаментальных положениях:

1. Соблюдение законности (предполагает наличие законов и иных нормативных документов, их применение и исполнение субъектами права в области информационной безопасности).

2. Обеспечение баланса интересов отдельных субъектов и государства (предусматривает приоритет государственных интересов как общих интересов всех субъектов. Ориентация на свободы, права и интересы граждан не принижает роль государства в обеспечении национальной безопасности в целом и в области информационной безопасности в частности).

3. Неотвратимость наказания (выполняет роль важнейшего профилактического инструмента в решении вопросов правового обеспечения).

Основные правовые акты, регламентирующие защиту информации в Республике Беларусь:

Постановление Совета Министров Республики Беларусь от 26 мая 2009 г. № 675 «О некоторых вопросах защиты информации» утверждает: Положение о порядке защиты информации в государственных информационных системах, а также информационных системах, содержащих информацию, распространение и (или) предоставление которой ограничено; Положение о порядке аттестации систем защиты информации; Положение о порядке проведения государственной экспертизы средств защиты информации.

Закон «Об информации, информатизации и защите информации» 10.11.2008 г. № 455-З регулирует общественные отношения, возникающие при поиске, получении, передаче, сборе, обработке, накоплении, хранении, распространении и (или) предоставлении информации, а также пользовании информацией; создании и использовании информационных технологий, информационных систем и информационных сетей, формировании информационных ресурсов; организации и обеспечении защиты информации.

Технический регламент Республики Беларусь «Информационные технологии. Средства защиты информации. Информационная безопасность» (ТР 2013/027/BY) принят постановлением Совета Министров Республики Беларусь 15.05.2013 г. № 375, «распространяется на выпускаемые в обращение средства защиты информации независимо от страны происхождения, за исключением средств шифрованной, других видов специальной связи и криптографических средств защиты государственных секретов»; устанавливает требования к средствам защиты информации в целях сохранения жизни и здоровья человека, имущества, а также предупреждения действий, вводящих в заблуждение потребителей (пользователей) относительно безопасности и качества средств защиты информации.

5.7. Обеспечение безопасности в компьютерных сетях

Информационный ресурс корпоративного уровня особенно уязвим и требует качественной и надежной защиты, так как информационная структура разнородна и состоит из распределенных систем, технологий, баз и банков данных. Мероприятия по защите

корпоративной информации должны обеспечивать выполнение следующих задач:

– защиту от проникновения в компьютерную сеть и утечки информации из сети по каналам связи;

– разграничение потоков информации между сегментами сети;

– защиту наиболее критичных ресурсов сети от вмешательства в нормальный процесс функционирования;

– защиту важных рабочих мест и ресурсов от несанкционированного доступа;

– криптографическую защиту наиболее важных информационных ресурсов.

Работа с сервисами сети Интернет существенно увеличивает диапазон угроз информации, обрабатываемой в корпорации.

Для поддержания режима информационной безопасности в компьютерных сетях особенно важны программно-технические меры и средства. Ключевыми механизмами являются: идентификация и аутентификация; управление доступом; технологии обнаружения атак; протоколирование и регистрация; криптография и сетевая защита; экранирование.

Использование в корпоративных сетях межсетевых экранов предотвращает возможность нарушения пользователями установленных администраторами правил безопасности информации, позволяет решать ряд задач:

– безопасное взаимодействие пользователей и информационных ресурсов, расположенных в экстранет- и интранет-сетях, с внешними сетями;

– создание технологически единого комплекса мер защиты для распределенных и сегментированных локальных сетей подразделений предприятия;

– построение иерархической системы защиты, предоставляющей адекватные средства обеспечения безопасности для различных по степени закрытости сегментов корпоративной сети.

Для защиты информации, передаваемой по открытым каналам связи, поддерживающим протоколы TCP/IP, существует ряд программных продуктов, предназначенных для построения VPN на основе международных стандартов IPSec. Виртуальные сети создаются чаще всего на базе арендуемых и коммутируемых каналов связи

в сетях общего пользования (сеть Интернет). Для небольших и средних компаний они являются хорошей альтернативой изолированным корпоративным сетям, так как обладают преимуществами: высокая гарантированная надежность, изменяемая топология, простота конфигурирования и масштабирования, контроль всех событий и действий в сети, относительно невысокая стоимость аренды каналов и коммуникационного оборудования.

Системы шифрования с открытым криптографическим интерфейсом реализуют различные криптоалгоритмы. Это дает возможность использовать продукты в любой стране мира в соответствии с принятыми национальными стандартами. В настоящее время информационные продукты, предназначенные для шифрования в корпоративных сетях, устанавливаются только на тех рабочих местах, на которых хранится очень важная информация.

Наличие разнообразных модификаций (для клиентских, серверных платформ, сети масштаба офиса, генерации ключевой информации) позволяет подбирать оптимальное по стоимости и надежности решение с возможностью постепенного наращивания мощности системы защиты.

В связи с постоянными изменениями сети важно своевременное выявление новых уязвимых мест, угроз и атак на информационные ресурсы и внесение изменений в соответствующие настройки информационного комплекса и его подсистем, и в том числе в подсистему защиты. Для этого рабочее место администратора системы должно быть укомплектовано специализированными программными средствами обследования сетей и выявления уязвимых мест (наличия «дыр») для проведения атак «извне» и «снаружи», а также комплексной оценки степени защищенности от атак нарушителей. Например, в состав продуктов ЭЛВИС+, Net Pro VPN входят наиболее мощные среди обширного семейства коммерческих пакетов продукты компании Internet Security Systems (Internet Scanner и System Security Scanner), а также продукты компании Cisco: система обнаружения несанкционированного доступа NetRanger и сканер уязвимости системы безопасности NetSonar.

Поддержка процесса идентификации и аутентификации пользователей и реализация механизма интеграции существующих приложений и всех компонент подсистемы безопасности осуществляется на основе технологии построения инфраструктуры открытых ключей (Public Key Infrastruture – PKI). Основными функциями PKI являются поддержка

жизненного цикла цифровых ключей и сертификатов (генерация, распределение, отзыв и пр.). Несмотря на существующие международные стандарты, определяющие функционирование системы PKI и способствующие ее взаимодействию с различными средствами защиты информации, к сожалению, не каждое средство информационной защиты, даже если его производитель декларирует соответствие стандартам, может работать с любой системой PKI.

Наиболее критичными с точки зрения безопасности ресурсами в корпоративных сетях являются серверы. Основным способом вмешательства в нормальный процесс их функционирования является проведение атак с использованием уязвимых мест сетевого аппаратного и программного обеспечения. Основная задача – своевременное обнаружение атаки и противодействие ей.

Защита важных рабочих мест и ресурсов от несанкционированного доступа как средство обеспечения информационной безопасности в компьютерных сетях проводится с помощью дополнительных средств защиты: криптографических, регламентирования и протоколирования действий пользователей, разграничения прав пользователей по доступу к локальным ресурсам.

Одним из важнейших методов повышения безопасности сетей является использование средств, предусмотренных в стандартных протоколах их построения (модель OSI). Обязательно применение следующих методов: организация доменов безопасности на уровне транспортной сети и на уровне сервиса электронной почты; регистрация попыток пользователей установления соединений вне заданных доменов безопасности; протокольная аутентификация устанавливаемых соединений; определение доступного сервиса для каждого конкретного абонента и др.

Проблема защиты информации сети Интернет сегодня самая актуальная и в ней выделяются две категории: общая безопасность и надежность финансовых операций. Безопасность сети можно обеспечивать на различных уровнях сетевого взаимодействия (рис. 5.6, табл. 5.3) с использованием различных протоколов защиты.

В качестве протокола сетевого уровня в настоящее время используется протокол IPv6, который обеспечивает расширенное адресное пространство; улучшенные возможности маршрутизации; управление доставкой информации; средства обеспечения безопасности, использующие алгоритмы аутентификации и шифрования.

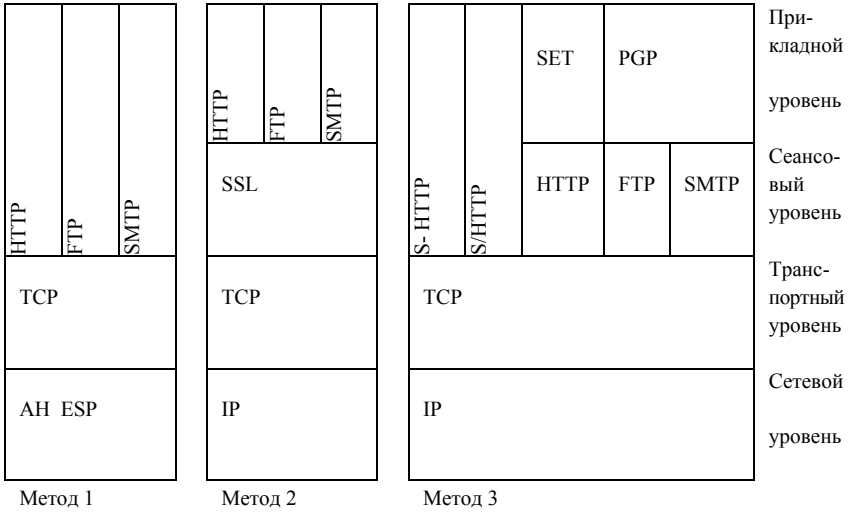


Рис. 5.6. Методы защиты данных в сетях

Таблица 5.3

Протоколы защиты данных в сети Интернет

Стандарт	Функция	Применение
Secure HTTP (S-HTTP)	Защита транзакций в *сети*	Браузеры, web-серверы, приложения для сети Интернет
Secure Sockets Layer (SSL)	Защита пакетов данных на сетевом уровне	Браузеры, web-серверы, приложения для сети Интернет
Secure MIME (S/MIME)	Защита вложений в электронные сообщения на различных платформах	Почтовые программы с поддержкой шифрования и цифровой подписи RSA

Окончание табл. 5.3

Стандарт	Функция	Применение
Secure Wide Area Networks (S/WAN)	Шифрование одноранговых соединений между брандмауэрами и маршрутизаторами	Виртуальные частные сети
Secure Electronic Transaction (SET)	Защита транзакций с кредитными картами	Smart-карты, серверы транзакций, электронная коммерция

Спецификация IPsec, входящая в стандарт IPv6, предусматривает стандартный способ шифрования трафика на сетевом уровне IP и обеспечивает защиту на основе сквозного шифрования (шифрует каждый проходящий по каналу пакет независимо от приложения, что позволяет создавать в сети Интернет виртуальные частные сети). IPsec использует различные методы обеспечения комплексной информационной безопасности (применение цифровой подписи с использованием открытого ключа; алгоритм шифрования, подобный DES, для шифрования передаваемых данных; использование хэш-алгоритма для определения подлинности пакетов и др.)

Преимущества применения IPsec на сетевом уровне: поддержка отличных от TCP протоколов; поддержка виртуальных сетей в незащищенных сетях; более надежная защита от анализа трафика; защита от атак типа «отказ в обслуживании» и др.

Ядро IPSec составляют три протокола: протокол аутентификации (Authenti-cation Header – AH), который гарантирует целостность и аутентичность данных; протокол шифрования (Encapsulation Sectirity Payload – ESP), который шифрует передаваемые данные, гарантируя конфиденциальность, может также поддерживать аутентификацию и целостность данных; протокол обмена ключами (Internet Key Exchange – IKE), где решает вспомогательную задачу автоматического предоставления конечным

точкам канала секретных ключей, необходимых для работы протоколов аутентификации и шифрования данных.

Протокол IPsec создавался в рамках разработок средств защищенной передачи пакетов в сети Интернет-2.

Контрольные вопросы

1. Перечислите основные методы и средства защиты информации.
2. Дайте определения кодирования и декодирования.
3. Как осуществляется защита от несанкционированного доступа к данным?
4. Охарактеризуйте классы безопасности компьютерных систем.
5. Сформулируйте определение электронной цифровой подписи.

6. МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ И ЧИСЛЕННЫЕ МЕТОДЫ

6.1. Математические модели и численные методы решения задач в различных предметных областях

В современном мире математика все больше и больше становится одним из важных инструментов познания человеком окружающего мира. Математика является основным методом теоретического исследования и практическим орудием в естествознании и технике, без математики совершенно невозможно проводить серьезные научные и инженерные расчеты. Недаром родоначальник немецкой классической философии Иммануил Кант (1742–1804 гг.) утверждал, что «в каждой отдельной естественной науке можно найти собственно науку лишь постольку, поскольку в ней можно найти математику». Математика, как наука, возникла в связи с необходимостью решения практических задач: измерений на местности, навигации и т. д. Вследствие этого математика всегда была численной математикой, ее целью являлось получение решения задач в виде числа. Создание ЭВМ дало новый толчок развитию математики, появились новые дисциплины «математическая экономика», «математическая химия», «математическая лингвистика» и т. д. Возникло понятие «математическое моделирование». Слово «*модель*» происходит от латинского *modus* (копия, образ, очертание). Моделирование – это замещение некоторого объекта А (оригинала) другим объектом Б (моделью).

Математическая модель – это упрощенное описание реальности с помощью математических понятий. Математическое моделирование – процесс построения и изучения математических моделей реальных процессов и явлений, т. е. метод исследования объектов и процессов реального мира с помощью их приближенных описаний на языке математики – математических моделей. Крупнейшие ученые прошлого сочетали в своих трудах как построение математического описания явлений природы (математические модели), так и его исследования. Анализ усложненных моделей требовал создания новых, как правило, численных методов решения задач.

Основоположником отечественного математического моделирования справедливо считают академика А. А. Самарского. Он выразил методологию математического моделирования знаменитой триадой *«модель – алгоритм – программа»*.

I этап. *Модель*. Выбирается или строится модель исследуемого объекта, которая в математической форме отражает его важнейшие свойства. Обычно математические модели реальных процессов достаточно сложны и включают в себя системы нелинейных функционально-дифференциальных уравнений. Ядром математической модели, как правило, являются уравнения с частными производными. Для получения предварительных знаний об объекте построенная модель исследуется традиционными аналитическими средствами прикладной математики.

II этап. *Алгоритм*. Выбирается или разрабатывается вычислительный алгоритм для реализации построенной модели на компьютере, который не должен искажать основные свойства модели, должен быть адаптирующимся к особенностям решаемых задач и используемым вычислительным средствам. Проводится изучение построенной математической модели методами вычислительной математики.

III этап. *Программа*. Создается программное обеспечение для реализации модели и алгоритма на компьютере. Создаваемый программный продукт должен учитывать важнейшую специфику математического моделирования, связанную необходимостью использования набора математических моделей и многовариантностью расчетов. В результате исследователь получает в руки универсальный, гибкий и недорогой инструмент, который сначала отлаживается, тестируется и калибруется на решении набора пробных задач. Затем проводится широкомасштабное исследование математической модели для получения необходимых качественных и количественных свойств и характеристик исследуемого объекта. Предложенная методология получила свое развитие в виде технологии «*вычислительного эксперимента*». Вычислительный эксперимент – это информационная технология, предназначенная для изучения явлений окружающего мира, когда натурный эксперимент оказывается либо невозможен (например, при изучении здоровья человека), либо слишком опасен (например, при изучении экологических явлений), либо слишком дорог и сложен (например, при изучении

астрофизических явлений). Широкое применение ЭВМ в математическом моделировании, разработанная теория и значительные практические результаты позволяют говорить о вычислительном эксперименте как о новой технологии и методологии научных и практических исследований. Серьезное внедрение вычислительного эксперимента в инженерную деятельность еще не очень широко, но там, где оно происходит реально (в авиационной и космической промышленности) его плоды весьма весомы. Отметим некоторые достоинства вычислительного эксперимента по сравнению с натурным. Вычислительный эксперимент, как правило, дешевле физического. В этот эксперимент можно легко и безопасно вмешиваться. Его можно повторить еще раз, если это необходимо, и прервать в любой момент. В ходе этого эксперимента можно смоделировать условия, которые нельзя создать в лаборатории. В ряде случаев проведение натурного эксперимента затруднено, а иногда и невозможно. Часто проведение полномасштабного натурного эксперимента сопряжено с губительными или непредсказуемыми последствиями (ядерная война, поворот сибирских рек) или с опасностью для жизни или здоровья людей. Нередко требуется исследование и прогнозирование катастрофических явлений (авария ядерного реактора АЭС, глобальное потепление или похолодание климата, цунами, землетрясение). В этих случаях вычислительный эксперимент может стать основным средством исследования. С его помощью оказывается возможным прогнозировать свойства новых, еще не созданных конструкций и материалов на стадии их проектирования. В то же время нужно помнить, что применимость результатов вычислительного эксперимента ограничена рамками принятой математической модели. В отличие от натурных исследований вычислительный эксперимент позволяет накапливать результаты, полученные при исследовании какого-либо круга задач, а затем эффективно применять их к решению задач в других областях. Например, уравнение нелинейной теплопроводности описывает не только тепловые процессы, но и диффузии вещества, движения грунтовых вод, фильтрации газа в пористых средах. Изменяется только физический смысл величин, входящих в это уравнение. После проведения первого этапа вычислительного эксперимента может возникнуть необходимость в уточнении модели. На втором этапе учитываются дополнительные эффекты и связи в изучаемом

явлении, либо возникает необходимость пренебречь некоторыми закономерностями и связями. Затем этот процесс повторяют до тех пор, пока не убеждаются, что модель адекватна изучаемому объекту. Обычно в процессе математического моделирования и вычислительного эксперимента участвуют помимо профессиональных математиков и программистов специалисты в конкретной предметной области (биологии, химии, медицине и др.). Первый серьезный вычислительный эксперимент был проведен в СССР в 1968 г. группой ученых под руководством академиков А. Н. Тихонова и А. А. Самарского. Это было открытие, так называемого, эффекта Т-слоя (температурного токового слоя в плазме, которая образуется в МГД-генераторах) – явления, которого на самом деле никто не наблюдал. И только через несколько лет Т-слой был зарегистрирован в экспериментальных физических лабораториях и технологам и инженерам окончательно стал ясен принцип работы МГД-генератора с Т-слоем. В последние годы ряд Нобелевских премий по химии, медицине, экономике, физике элементарных частиц были присуждены работам, методологическую основу которых составляло именно математическое моделирование. Математические модели для описания изучаемых явлений в механике, физике и других точных науках естествознания использовались достаточно давно. 3 – 4 тысячи лет назад решали задачи прикладной математики, связанные с вычислением площадей и объемов, расчетами простейших механизмов, т. е. с несложными задачами арифметики, алгебры и геометрии. Вычислительными средствами служили собственные пальцы, а затем – счеты. Большинство вычислений выполнялось точно, без округлений. В XVII в. Исаак Ньютон полностью описал закономерности движения планет вокруг Солнца, решал задачи геодезии, проводил расчеты механических конструкций. Задачи сводились к обыкновенным дифференциальным уравнениям, либо к алгебраическим системам с большим числом неизвестных, вычисления проводились с достаточно высокой точностью до восьми значащих цифр. При вычислениях использовались таблицы элементарных функций, арифмометр, логарифмическая линейка; к концу этого периода появились неплохие клавишные машины с электромотором. В это время были разработаны алгоритмы численных методов, которые до сих пор занимают почетное место в арсенале вычислительной математики. Так Ньютон предложил эффективный

численный метод решения алгебраических уравнений, а Эйлер – численный метод решения обыкновенных дифференциальных уравнений. Классическим примером применения численных методов является открытие планеты Нептун, а так же планеты Уран, следующей за Сатурном, который много веков считался самой из далеких планет. К 40-м годам XIX в. точные наблюдения показали, что Уран едва заметно уклоняется от того пути, по которому он должен следовать с учетом возмущений со стороны всех известных планет. Леверье (во Франции) и Адамс (в Англии) высказали предположение, что если возмущения со стороны известных планет не объясняют отклонение в движении Урана, значит на него действует притяжение еще неизвестного тела. Они почти одновременно рассчитали, где за Ураном должно быть неизвестное тело, производящее своим притяжением эти отклонения. Они вычислили орбиту неизвестной планеты, ее массу и указали место на небе, где в данное время она должна была находиться. Эта планета и была найдена в телескоп на указанном ими месте в 1846 г. Ее назвали Нептуном. Для расчета траектории Нептуна Леверье понадобилось полгода. Численное решение прикладных задач всегда интересовало математиков. Разработкой численных методов занимались крупнейшие ученые своего времени: Ньютон, Эйлер, Лобачевский, Гаусс, Эрмит, Чебышев и др. Численные методы, разработанные ими, носят их имена. Развитие численных методов способствовало постоянному расширению сферы применения математики в других научных дисциплинах и прикладных разработках. Появление ЭВМ дало мощный импульс еще более широкому внедрению численных методов в практику научных и технических расчетов. Скорость выполнения вычислительных операций выросла в миллионы раз, что позволило решить широкий круг математических задач, бывших до этого практически не решаемыми. Разработка и исследование вычислительных алгоритмов, их применение к решению конкретных задач составляет содержание огромного раздела современной математики – вычислительной математики. Вычислительная математика как самостоятельная математическая дисциплина сформировалась в начале XX в. Ее определяют в широком смысле как раздел математики, исследующий широкий круг вопросов, связанных с использованием ЭВМ. В узком смысле вычислительную математику определяют как теорию численных методов и алгоритмов

решения поставленных математических задач. Современные компьютерно-ориентированные численные методы должны удовлетворять многообразным и зачастую противоречивым требованиям. Обычно построение численного метода для заданной математической модели разбивается на два этапа: дискретизацию исходной математической задачи и разработку вычислительного алгоритма, позволяющего отыскать решение дискретной задачи. Выделяют две группы требований к численным методам. Первая группа связана с адекватностью дискретной модели исходной математической задаче, вторая – с реализуемостью численного метода на имеющейся вычислительной технике. К первой группе относятся такие требования, как сходимость численного метода, выполнение дискретных аналогов законов сохранения, качественно правильное поведение решения дискретной задачи. Предположим, что дискретная модель математической задачи представляет собой систему большого числа алгебраических уравнений. Обычно, чем точнее мы хотим получить решение, тем больше уравнений приходится брать. Говорят, что численный метод сходится, если при неограниченном увеличении числа уравнений решение дискретной задачи стремится к решению исходной задачи. Поскольку реальный компьютер может оперировать лишь с конечным числом уравнений, на практике сходимость, как правило, не достигается. Следовательно, очень важно уметь оценивать погрешность метода в зависимости от числа уравнений, составляющих дискретную модель. По этой же причине стараются строить дискретную модель так, чтобы она правильно отражала качественное поведение решения исходной задачи даже при сравнительно небольшом числе уравнений. При дискретизации задач математической физики приходят к разностным схемам, представляющим собой системы линейных или нелинейных алгебраических уравнений. Дифференциальные уравнения математической физики являются следствиями интегральных законов сохранения. Поэтому естественно требовать, чтобы для разностной схемы выполнялись аналоги таких законов сохранения. Разностные схемы, удовлетворяющие этому требованию, называются консервативными. Оказалось, что при одном и том же числе уравнений в дискретной задаче консервативные разностные схемы более правильно отражают поведение решения исходной разностной задачи, чем неконсервативные схемы. Сходимость численного метода тесно связана с его корректностью. Пусть исходная математическая задача поставлена корректно, т. е. ее решение существует, единственно и непрерывно зависит от входных данных. Тогда дискретная модель этой задачи должна быть построена так, чтобы свойство корректности сохранялось. Следовательно, в понятие корректности численного метода включаются свойства однозначной разрешимости соответствующей системы уравнений и ее устойчивости. Под устойчивостью понимается непрерывная зависимость от входных данных. Вторая группа требований, предъявляемых к численным методам, связана с возможностью реализации данной дискретной модели на данном компьютере, т. е. с возможностью получить численное решение за приемлемое время. Обычно сложные вычислительные задачи, возникающие при исследовании физических и технических проблем, разбиваются на ряд элементарных. Многие элементарные задачи являются несложными, они хорошо изучены, для них уже разработаны методы численного решения и имеются стандартные программы решения. Целью данной главы является знакомство с методологией построения и исследования основных численных методов алгебры и математического анализа и проблемами, возникающими при численном решении задач.

Построение модели объекта, явления начинается с выделения его наиболее существенных черт и свойств и описания их с помощью математических соотношений. Затем, после создания математической модели, ее исследуют математическими методами, т. е. решают сформулированную математическую задачу.

Построение математической модели является одним из наиболее сложных и ответственных этапов исследования объекта. Математическая модель никогда не бывает тождественна рассматриваемому объекту, не передает всех его свойств и особенностей. Она основывается на упрощении, идеализации и является приблизительным описанием объекта. Поэтому результаты, получаемые на основе этой модели, имеют всегда приближенный характер. Их точность определяется степенью соответствия, адекватности модели и объекта. Вопрос о точности является важнейшим в прикладной математике. Однако он не является чисто математическим вопросом и не может быть решен математическими методами. Основным критерием истины является эксперимент, т.е. сопоставление результатов, получаемых на основе математической

модели, с рассматриваемым объектом. Только практика позволяет сравнить различные гипотетические модели и выбрать из них наиболее простую и достоверную, указать области применимости различных моделей и направление их совершенствования. Рассмотрим развитие модели на примере известной задачи баллистики об определении траектории тела, выпущенного с начальной скоростью ν_0 под углом α_0 к горизонту. Предположим, что скорость ν_0 и дальность полета тела небольшие. Тогда для данной задачи будет справедлива математическая модель Галилея, основанная на следующих допущениях:

- Земля – инерциальная система;
- ускорение свободного падения g = const;
- Земля – плоское тело;
- сопротивление воздуха отсутствует.

В этом случае составляющие скорости движения тела по осям x и y равны:

$$\nu_x = \nu_0 \cos(\alpha_0),\ \nu_y = \nu_0 \sin(\alpha_0) - gt, \tag{6.1}$$

а их пути:

$$x = tv_0 \cos(\alpha_0);$$

$$y = tv_0 \sin(\alpha_0) - \frac{gt^2}{2}, \tag{6.2}$$

где t – время движения.

Определяя t из первого уравнения и подставляя его во второе, получаем уравнение траектории тела, представляющее собой параболу:

$$y = x\mathrm{tg}(\alpha_0) - \frac{x^2 g}{2v_0 \cos(\alpha_0)}, \tag{6.3}$$

из условия $y = 0$ получаем дальность полета тела:

$$l = \frac{v_0^2}{g}\sin(2\alpha_0). \tag{6.4}$$

Однако как показывает практика, результаты, получаемые на основе этой модели, оказываются справедливыми лишь при малых начальных скоростях движения тела $v < 30$ м/с. С увеличением скорости v_0 дальность полета становится меньше величины, даваемой формулой (6.1).

Такое расхождение эксперимента с расчетной формулой (6.1) говорит о неточности модели Галилея, не учитывающей сопротивление воздуха.

Дальнейшее уточнение модели баллистической задачи в части учета сопротивления воздуха было сделано Ньютоном. Это позволило с достаточной точностью рассчитывать траектории движения пушечных ядер, выстреливаемых со значительными начальными скоростями.

Переход от гладкоствольного к нарезному оружию позволил увеличить скорость, дальность и высоту полета снарядов, что вызвало дальнейшее уточнение математической модели задачи. В новой математической модели были пересмотрены все допущения, принятые в модели Галилея, т.е. Земля уже не считалась плоской и инерциальной системой, и сила земного притяжения не принималась постоянной.

Последующее совершенствование математической модели задачи связано с использованием методов теории вероятности. Это было вызвано тем, что параметры снарядов, орудий, зарядов и окружающей среды в силу допусков и других причин не остаются неизменными, а подчиняются случайным колебаниям.

В результате последовательных уточнений и усовершенствований была создана математическая модель наиболее полно и точно описывающая задачу внешней баллистики. Сопоставление ее данных с результатами стрельб показало хорошее их совпадение.

На этом примере показаны этапы создания, развития и уточнения математической модели объекта, которые сопровождаются постоянно сопоставлением и проверкой на практике, т. е. с самим реальным объектом или явлением. Именно недостаточно хорошее совпадение результатов, предоставляемых моделью, с объектом вызывает дальнейшее совершенствование модели.

6.2. Численное дифференцирование и интегрирование

6.2.1. Особенность задачи численного дифференцирования

Когда производную аналитически заданной функции по причине ее сложности искать затруднительно, либо выражение для производной приобретает неудобную для применения форму, используется приближенное или численное дифференцирование. Этот метод тем более необходим, если исходная функция задана таблично. Один из способов решения задачи дифференцирования – использование интерполяционных многочленов.

Пусть $f(x)$ – функция, для которой нужно найти производную в заданной точке отрезка $[a; b]$, $F_n(x)$ – интерполяционный многочлен для $f(x)$, построенный на отрезке $[a; b]$. Заменяя $f(x)$ интерполяционным многочленом $F_n(x)$, получим значение производной $f(x)$ на отрезке $[a; b]$ как значение производной $F_n'(x)$ интерполяционного многочлена, т.е. примем приближенно

$$f'(x) = F_n'(x). \tag{6.5}$$

Аналогичным путем можно поступать при нахождении значений производных высших порядков функции $f(x)$.

Полагая, что погрешность интерполирования определяется формулой:

$$R_n(x) = f(x) - F_n(x). \tag{6.6}$$

Получаем подход к оценке погрешности производной $F_n'(x)$:

$$r_n(x) = f'(x) - F_n'(x) = R_n'(x), \tag{6.7}$$

т. е. погрешность производной интерполирующей функции равна производной от погрешности этой функции.

6.2.2. Интерполяционная формула Лагранжа для равноотстоящих узлов

Рассмотрим методы численного дифференцирования на основе интерполяционных многочленов Лагранжа и Ньютона.

Применяя для численного дифференцирования на отрезке $[a; b]$ интерполяционный многочлен, естественно строить на этом отрезке систему равноотстоящих узлов:

$$a = x_0, x_1, x_2, \ldots, x_n = b,$$

которыми отрезок делится на n равных частей:

$$x_{i+1} - x_i = h = \text{const}; \tag{6.8}$$
$$(i = 0, 1, 2, n - 1).$$

В этом случае шаг интерполирования равен $h = \frac{b-a}{n}$, а интерполяционный многочлен Лагранжа строится на равноотстоящих узлах и имеет более удобный вид.

$$\frac{x - x_0}{h} = t. \tag{6.9}$$

С учетом формулы Лагранжа:

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{\Pi_{n+1}(x)}{(x - x_i) \cdot \Pi'_{n+1}(x)} = \Pi_{n+1}(x) \sum_{i=0}^{n} \frac{y_i}{(x - x_i) \cdot \Pi'_{n+1}(x)}, \tag{6.10}$$

получим новые выражения для $\Pi_{n+1}(x)$. Учитывая, что:

$$\Pi_{n+1}(x) = (x - x_0)(x - x_i) \ldots (x - x_n),$$

и используя (6.9), последовательно находим:

$$x - x_0 = ht;$$

$$x - x_1 = x - x_0 - h = h(t - 1);$$

$$x - x_2 = x - x_0 - 2h = h(t - 2);$$

т.е. в общем случае:

$$x - x_i = x - x_0 - ih = h(t - i), \tag{6.11}$$

$$i = 0, 1, \ldots, n.$$

$$\Pi_{n+1}(x)=h^{n+1}t(t-1)(t-2)\cdot\ldots(t-n).$$

Обозначим:

$$t(t-1)(t-2)\cdot\ldots(t-n)=t^{[n+1]},$$

тогда выражение $\Pi_{n+1}(x)$ примет вид:

$$\Pi_{n+1}(x)=h^{n+1}t^{[n+1]}. \qquad (6.12)$$

Учитывая, что при постоянном шаге имеет место:

$$x_i=x_0+ih,\ i=0,\ 1,\ \ldots,\ n,$$

последовательно находим:

$$\begin{aligned} x_i-x_0&=hi;\\ x_i-x_1=x_i-x_0-h&=h(i-1);\\ x_i-x_n=x_i-x_0-nh&=h(i-n). \end{aligned} \qquad (6.13)$$

Заметим, что в (6.13) ровно n строк (i-тая отсутствует), причем значения разностей из первых i строк положительны, а остальных – отрицательны. Используя (6.12), получаем:

$$\Pi_{n+1}(x_i)=(x_i-x_0)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)=h^n i(i-1)\cdot\ldots\cdot1\cdot(-1)\cdot\ldots\cdot(-(n-i)),$$

то есть

$$\Pi_{n+1}(x_i)=h^n i!(n-i)!(-1)^{n-i}. \qquad (6.14)$$

С учетом представлений (6.12) и (6.14) формула Лагранжа (6.10) для равноотстоящих узлов принимает вид:

$$L_n(x_0+th)=\sum_{i=0}^{n}y_i\frac{(-1)^{n-i}t^{[n+1]}}{i!(n-i)!(t-i)}. \qquad (6.15)$$

Пример 6.1.

Составить интерполяционный многочлен Лагранжа для функции, заданной своими значениями на равноотстоящих узлах ($n=2$, $h=1$):

Таблица 6.1

x	2	3	4
$f(x)$	4	–2	6

Используя формулу (6.15), запишем:

$$L_3(2+t)=4\frac{t(t-1)(t-2)}{2!t}+(-2)\frac{(-1)(t-1)(t-2)}{t-1}+6\frac{t(t-1)(t-2)}{2!(t-2)}=$$

$$=2(t-1)(t-2)+2t(t-2)+3t(t-1)=7t^2-13t+4.$$

6.2.3. Численное дифференцирование на основе интерполяционной формулы Лагранжа

Следуя (6.5), будем дифференцировать многочлен Лагранжа (6.15) по x как функцию от t:

$$f'(x)\approx L_n'(x)\,\frac{dx}{dt}=\sum_{i=0}^{n}y_i\frac{(-1)^{n-i}}{i!(n-i)!}\frac{d}{dt}\left[\frac{t^{[n+1]}}{t-i}\right].$$

Учитывая, что согласно (6.9) $x=x_0+th$ а также $\frac{dx}{dt}=h$, получим окончательно:

$$f'(x)=f'(x_0+th)\approx\frac{1}{h}\sum_{i=0}^{n}y_i\frac{(-1)^{n-i}}{i!(n-i)!}\frac{d}{dt}\left[\frac{t^{[n+1]}}{t-i}\right]. \qquad (6.16)$$

Пользуясь формулой (6.16), можно вычислять приближенные значения производной функции $f(x)$, если она задана на отрезке $[a;\ b]$ значениями в равноотстоящих узлах $a=x_0,x_1,x_2,\ \ldots,\ x_n=b$. Анналогично могут быть найдены производные функции $f(x)$ высших порядков.

Пример 6.2.

Вычислить приближенное значение производной функции, заданной таблицей 6.2 в точке x=4.

Таблица 6.2

x	3	4	5
$f(x)$	2	–1	6

Используя формулу (6.16), получим (n = 2, h = 1):

$$f'(x)\approx 2\cdot\frac{1}{2!}\cdot\frac{d\left[(t-1)(t-2)\right]}{dt}-\frac{(-1)}{1}\cdot\frac{d\left[(t-2)\right]}{dt}+6\cdot\frac{1}{2!}\cdot\frac{d\left[t(t-1)\right]}{dt}=$$

$$=(t-2+t-1)=(t-2+t)+3(t-1+t)=2t-3+2t-2+6t-3=10t-8.$$

Учитывая, что узел x=4 соответствует значению t=1, т. е.

$$t=\frac{x-x_0}{h}\to f'(4)\approx 2.$$

Если известно аналитическое выражение функции $f(x)$, то формулу для оценки погрешности численного дифференцирования можно при этом же условии получить на основе формулы погрешности интерполирования:

$$R_n(x)=f(x)-F_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\Pi_{n+1}(x), \qquad (6.17)$$

где $\xi=\xi(x)$ – значение из отрезка [a; b], отличное от узлов и x.

Учитывая (6.7) и допуская, что $f(x)$ дифференцируема n + 1 раз, запишем:

$$r_n(x)=R_n'(x)=\frac{1}{(n+1)!}\left(f^{(n+1)}(\xi)\cdot\Pi_{n+1}(x)\cdot\frac{d}{dx}\left[f^{(n+1)}(\xi)\right]\right), \qquad (16.18)$$

Формула (6.18) значительно упрощается, если оценка находится для значения производной $f'(x)$ в узле x_i таблицы. В этом случае, учитывая (6.14), получаем:

$$r_n(x_i)=R_n'(x)=(-1)^{n-i}\cdot h^n\frac{i!(n-i)1}{(n+1)!}\cdot f^{(n+1)}(\xi), \qquad (6.19)$$

где ξ – промежуточное значение между $x_0,x_1,x_2,..,x_n$.

Обозначив:

$$M_{n+1}=\max_{x_0\le x\le x_n}\left|f^{(n+1)}(x)\right|,$$

получим верхнюю оценку абсолютной ошибки численного дифференцирования в узлах:

$$\left|r_n(x_i)\right|\le\frac{M_{n+1}}{(n+1)!}h^n i!(n-i)!. \qquad (6.20)$$

6.2.4. Численное дифференцирование на основе интерполяционной формулы Ньютона

Запишем для функции $f(x)$, заданной своими значениями в равноотстоящих узлах $x_0,x_1,x_2,\ \dots\ ,x_n$ первый интерполяционный многочлен Ньютона:

$$P_n(x)=P_n(x_0+th)=y_0+t\Delta y_0+\frac{t(t-1)}{2!}\Delta^2y_0-\frac{t(t-1)(t-2)}{3!}\Delta^3y_0+\ldots+$$

$$+\frac{t(t-1)\cdot\ldots\cdot(t-n+1)}{n!}\Delta^ny_0.$$

Перепишем этот полином, производя перемножение скобок:

$$P_n(x_0+th)=$$

$$=y_0+t\Delta y_0+\frac{t^2-t}{2}\Delta^2y_0-\frac{t^3-3t^2+2t}{3!}\Delta^3y_0+\frac{t^4-6t^3+11t^2-6t}{24}\Delta^4y_0+\ldots$$

Дифференцируя $P_n(x_0+th)$ по t, получим аналогично формуле (6.16):

$$f'(x)\approx P_n'(x_0+th)=$$

$$=\frac{1}{h}\left(\Delta y_0+\frac{2t-1}{2!}\Delta^2 y_0-\frac{3t^2-6t+2}{6}\Delta^3 y_0+\frac{2t^3-9t^2+11t-3}{12}\Delta^4 y_0+\ldots\right). \quad (6.21)$$

Подобным путем можно получить и производные функции $f(x)$ более высоких порядков. Однако каждый раз вычисляя значение производной функции $f(x)$ в фиксированной точке x, в качестве x_0 следует брать ближайшее слева узловое значение аргумента.

Формула 6.21 существенно упрощается, если исходным значением x оказывается один из узлов таблицы. Так как в этом случае каждый узел можно считать начальным, то принимая $x = x_0$, $t = 0$, получаем:

$$f'(x_0)=\frac{1}{h}\left(\Delta y_0-\frac{\Delta^2 y_0}{2}+\frac{\Delta^3 y_0}{3}-\frac{\Delta^4 y_0}{4}+\frac{\Delta^5 y_0}{5}\ldots\right). \quad (6.22)$$

Эта формула позволяет точно получать значения производных функций, заданных таблично.

Выведем формулу погрешности дифференцирования. Используя формулу (6.17) применительно к первому интерполяционному многочлену Ньютона, запишем:

$$R_n(x)=h^{n+1}\frac{t(t-1)(t-2)\ldots(t-n)}{(n+1)!}f^{(n+1)}(\xi),$$

где ξ – промежуточное значение между $x_0, x_1, x_2, \ldots, x_n$ и заданной точкой x. Предполагая, что $f(x)$ дифференцируема $n + 1$ раз, получим для оценки погрешности дифференцирования $r_n(x_i)$ (по аналогии с формулой (6.18)):

$$r_n(x)=R_n'(x)=\frac{h^n}{(n+1)!}\left(f^{(n+1)}(\xi)\frac{d}{dt}t^{(n+1)}+t^{(n+1)}\frac{d}{dt}\left[f^{(n+1)}(\xi)\right]\right). \quad (6.23)$$

Для случая оценки погрешности в узле таблицы получим:

$$R_n'(x)=(-1)^n\frac{h^n}{n+1}f^{(n+1)}(\xi).$$

На практике $f^{(n+1)}(\xi)$ оценивать непросто, поэтому при малых h приближенно полагают:

$$f^{(n+1)}(\xi)\approx\frac{\Delta^{n+1}y_0}{h^{n+1}},$$

что позволяет использовать приближенную формулу:

$$r_n(x_0)\approx\frac{(-1)^n\cdot\Delta^{n+1}y_0}{h(n+1)}. \quad (6.24)$$

6.2.5. Постановка задачи численного интегрирования

При вычислении определенного интеграла:

$$I1=\int_a^b f(x)dx,$$

где $f(x)$ – непрерывная на отрезке $[a; b]$ функция, иногда удается воспользоваться формулой Ньютона–Лейбница:

$$\int_a^b f(x)dx=F(b)-F(a). \quad (6.25)$$

Здесь $F(x)$ – одна из первообразных функции $f(x)$. Однако даже в тех редких случаях, когда первообразную удается явно найти в аналитической форме, не всегда удается довести до числового ответа значение определенного интеграла. Если учесть, что подынтегральная функция задается таблицей или графиком, то интегрирование по формуле (6.25) не получает широкого применения на практике.

В подобных случаях применяют различные методы численного интегрирования. Формулы, используемые для вычисления однократных интегралов, называют квадратурными формулами.

Прием построения квадратурных формул состоит в том, что подынтегральная функция *f*(*x*) заменяется на отрезке [*a; b*] интерполяционным многочленом, например, многочленом Лагранжа $L_n(x)$, и получается приближенное равенство

$$\int_a^b f(x)dx \approx \int_a^b L_n(x)dx. \qquad (6.26)$$

Предполагается, что отрезок [*a; b*] разбит на *n* частей точками $a = x_0,\ x_1, x_2, \ldots, x_n = b$, наличие которых подразумевается при построении многочлена $L_n(x)$.

Подставляя вместо $L_n(x)$ (6.10), получим:

$$\int_a^b f(x)dx \approx \int_a^b \sum_{i=0}^{n} y_i \frac{\Pi_{n+1}(x)}{(x-x_i)\Pi_{n+1}(x)}dx = \sum_{i=0}^{n} y_i \int_a^b \frac{\Pi_{n+1}(x)}{(x-x_i)\Pi_{n+1}(x)}dx.$$

Таким образом,

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} y_i A_i, \qquad (6.27)$$

где

$$A_i = \int_a^b \frac{\Pi_{n+1}(x)}{(x-x_i)\Pi'_{n+1}(x)}dx. \qquad (6.28)$$

В формуле 6.27 коэффициенты A_i не зависят от функции *f*(*x*), так как они составлены только с учетом узлов интерполяции. Если *f*(*x*) – полином степени *n*, то формула (6.27) точная, так как $L_n(x) \equiv f(x)$.

6.2.6. Квадратурные формулы Ньютона – Котеса

Применение формулы (6.26) предполагает построение на отрезке интегрирования [*a*; *b*] системы узлов интерполяции $a = x_0, x_1, x_2, \ldots, x_n = b$, которыми отрезок делится на *n* частей.

Длина $x_{i+1} - x_i = h,\ i = 1,2,\ldots,n-1$ называется шагом интегрирования. Естественно считать, что шаг *h* постоянен, т.е.:

$$h = \frac{b-a}{n}.$$

В этом случае можно применить интерполяционную формулу Лагранжа для равноотстоящих узлов. Итак, с учетом (6.12) и (6.14) формула (6.28) для весовых коэффициентов A_i примет вид:

$$A_i = \int_{x_0}^{x_n} \frac{(-1)^{n-i}\, t^{[n+1]}}{i!(n-i)!(t-i)}dx, \quad i = 1,2,\ldots,n-1\ . \qquad (6.29)$$

Перейдем в этом интеграле к переменной *t*. Из подстановки (6.9) получаем:

$$dt = \frac{dx}{h}, \text{ т.е. } dx = hdt = \frac{b-a}{n}dt.$$

При $x = x_0$ имеем $t = 0$, а при $x = x_n$ будет:

$$t = \frac{x_n - x_0}{h} = n.$$

Тогда:

$$A_i = \frac{b-a}{n}\int_{0_0}^{n} \frac{(-1)^{n-i}\, t^{[n+1]}}{i!(n-i)!(t-i)}dt = (b-a)H_i, \qquad (6.30)$$

где

$$H_i = \frac{1}{n}\int_{0_0}^{n} \frac{(-1)^{n-i}\, t^{[n+1]}}{i!(n-i)!(t-i)}dt,\ i = 0,1,2,\ldots,n \qquad (6.31)$$

Числа (6.29) называют коэффициентами Котеса. Они не зависят от функции *f(x)*, а только от числа точек разбиения. Окончательно, с учетом формул (6.27) и (6.30) получаем следующий вид квадратурных формул Ньютона - Котеса:

$$\int_a^b f(x)dx \approx (b-a)\sum_{i=0}^{n} y_i H_i, \qquad (6.32)$$

дающих на одном участке интегрирования различные представления различного числа *n* отрезков разбиения.

6.2.7. Формула трапеций

При *n*=1 из формулы (6.31) имеем: ($i = 0; 1$):

$$H_0 = -\int_0^1 \frac{t(t-1)}{t}dt = -\int_0^1 (t-1)dt = \frac{1}{2};$$

$$H_1 = \int_0^1 tdt = \frac{1}{2}.$$

Тогда по формуле (6.32) на отрезке $[x_0; x_1]$ получаем интеграл:

$$\int_{x_0}^{x_1} f(x)dx = (x_1 - x_0)(H_0 y_0 + H_1 y_1) = \frac{h}{2}(y_0 + y_1), \qquad (6.33)$$

Формула (6.33) дает один из простейших способов вычисления определенного интеграла и называется формулой трапеций. Действительно, при *n*=1 подынтегральная функция заменяется интерполяционным многочленом Лагранжа первой степени (т.е. линейной функцией), а геометрически это означает, что площадь криволинейной фигуры заменяется площадью трапеции.

Распространяя формулу (6.33) на все отрезки разбиения, получим общую формулу трапеций для отрезка $[a; b]$:

$$\int_a^b f(x)dx = h\left(\frac{y_0}{2} + y_1 + y_2 + \ldots + y_{n-1} + \frac{y_n}{2}\right). \qquad (6.34)$$

Если аналитическое выражение для подынтегральной функции известно, может быть поставлен вопрос об оценке погрешности численного интегрирования по формуле (6.34) (погрешность метода).

В этом случае имеется ввиду, что:

$$\int_a^b f(x)dx = \int_a^b L_n(x)dx + R_n(f),$$

где $R_n(f)$ – остаточный член квадратурной формулы (6.34). Формулу остаточного члена получим вначале для отрезка $[x_0; x_1]$.

Имеем:

$$R = \int_{x_0}^{x_1} f(x)dx - \frac{h}{2}(y_0 + y_1) = \int_{x_0}^{x_0+h} f(x)dx - \frac{h}{2}(f(x_0) + f(x_0 + h)),$$

откуда следует, что естественно рассматривать *R* как функцию шага *h*: *R*=*R*(*h*). Заметим, что *R*(0)=0.

Продифференцируем *R*(*h*) по *h*:

$$R'(h) = \left(\int_{x_0}^{x_0+h} f(x)dx\right)' - \frac{1}{2}(f(x_0) + f(x_0 + h)) - \frac{h}{2}(f'(x_0 + h)) =$$

$$= f(x_0 + h) - \frac{1}{2}f(x_0 + h) - \frac{1}{2}f(x_0) - \frac{h}{2}f'(x_0 + h) =$$

$$= \frac{1}{2}(f(x_0 + h) - f(x_0)) - \frac{h}{2}f'(x_0 + h).$$

Заметим, что *R*`(0)=0. Далее:

$$R''(h) = \frac{1}{2}f'(x_0 + h) - \frac{1}{2}f'(x_0 + h) - \frac{h}{2}f''(x_0 + h) = -\frac{h}{2}f''(x_0 + h) \qquad (6.35)$$

Определим R, последовательно интегрируя $R''(h)$ на отрезке $[0; h]$:

$$\int_0^h R''(z)dz = R'(h) - R'(0) = R'(h),$$

откуда с учетом (6.35) имеем:

$$R'(h) = \int_0^h R''(z)dz = -\frac{1}{2}\int_0^h zf''(x_0 + z)dz. \qquad (6.36)$$

Применяя к (6.36) обобщенную теорему о среднем, получаем:

$$R'(h) = -\frac{1}{2} f''(\xi_1) \int_0^h z\,dz = -\frac{h^2}{4} f''(\xi_1), \qquad (6.37)$$

где $\xi_1 \in [x_0; x_0 + h]$ и ξ_1 зависит от h.

Далее $\int_0^h R'(z)dz = R(h) - R(0) = R(h)$, откуда с учетом (6.37) и обобщенной теоремы о среднем имеем:

$$R(h) = \int_0^h R'(z)dz = \frac{1}{4}\int_0^h z^2 f''(\xi_1)dz = -\frac{h^3}{12} f''(\xi),$$

где $\xi_1 \in [x_0; x_0 + h]$.

Таким образом, погрешность метода при интегрировании функции на отрезке $[x_0; x_1]$ по формуле (6.34) имеет величину:

$$R = -\frac{h^3}{12} f''(\xi), \quad \xi \in [x_0; x_1]. \qquad (6.38)$$

Из формулы (6.38) видно, что при $f''(\xi) > 0$ формула (6.34) дает значение интеграла с избытком, а при $f''(\xi) < 0$ – с недостатком. Можно показать, что при распространении оценки (6.38) на весь отрезок интегрирования $[a; b]$ получается формула:

$$R_n = \frac{h^3 n}{12} f''(\xi), \quad \xi \in [a; b].$$

С учетом $hn = b - a$ найден следующий окончательный вид для оценки погрешности метода интегрирования по формуле трапеций:

$$|R_n| \le M \frac{|b-a|h^2}{12}, \qquad (6.39)$$

где $M = \max_{x \in [a,b]} |f''(x)|$.

6.2.8. Формула Симпсона

При $n = 2$ из формулы (6.31) последовательно имеем ($i = 0, 1, 2$):

$$H_0 = -\frac{1}{2}\int_0^2 \frac{t(t-1)(t-2)}{t} dt = \frac{1}{6};$$

$$H_1 = -\frac{1}{2}\int_0^2 t(t-2)dt = \frac{2}{3};$$

$$H_2 = \frac{1}{4}\int_0^2 t(t-1)dt = \frac{1}{6}.$$

Тогда с учетом (6.32) получим на отрезке $[x_0; x_2]$:

$$\int_{x_0}^{x_2} f(x)dx \approx (x_2 - x_0)\sum_{i=0}^{2} H_i y_i = 2h\left(\frac{1}{6}y_0 + \frac{2}{3}y_1 + \frac{1}{6}y_2\right),$$

т.е.

$$\int_{x_0}^{x_2} f(x)dx \approx \frac{h}{3}\left(\frac{1}{6}y_0 + 4y_1 + y_2\right). \qquad (6.40)$$

Геометрически, в соответствии со смыслом интерполяционной формулы Лагранжа при $n = 2$, использование формулы (6.40) означает замену подынтегральной функции $f(x)$ параболой $L_2(x)$, проходящей через точки $M_i(x_i, y_i)$ ($i = 0, 1, 2$).

Если считать, что n – четное число ($n = 2m$), то применяя формулу (6.40) последовательно к каждой паре частичных отрезков $[x_{2i-2}; x_{2i}]$ ($i = 1, 2, \ldots, m$) получим:

$$\int_a^b f(x)dx \approx \frac{2h}{3}\left(\frac{y_0 + y_{2m}}{2} + 2y_1 + y_2 + \ldots + 2y_{2m-1}\right). \qquad (6.41)$$

Формула (6.41) называется формулой Симпсона.

Оценка остаточного члена формулы Симпсона дается формулой:

$$R_n = \frac{(b-a)h^4}{180} f^{IV}(\xi), \quad \xi \in [a; b].$$

или

$$|R_n| \le M\frac{(b-a)h^4}{180}, \qquad (6.42)$$

где

$$M = \max_{x\in[a,b]_n} \left|f^{IV}(x)\right|.$$

Как следует из оценки, формула Симпсона, оказывается точной для полиномов до третьей степени включительно (т.к. для этих случаев производная четвертого порядка равна нулю). Формула Симпсона обладает повышенной точностью по сравнению с формулой трапеций. Это означает, что для достижения той же точности, что и по формуле трапеций, можно брать меньшее число отрезков разбиения.

Укажем простой практический прием позволяющий прогнозировать требуемое число отрезков разбиения по заданной точности ε.

Пусть задана предельная допустимая погрешность интегрирования ε. Желая иметь $|R| \le \varepsilon$ с учетом оценки (6.42) достаточно потребовать:

$$M\frac{(b-a)h^4}{180} \le \varepsilon,$$

Откуда:

$$h^4 \le \frac{180\varepsilon}{|b-a|\cdot M}, \text{ т. е. } h \le \sqrt[4]{\frac{180\varepsilon}{|b-a|\cdot M}}. \qquad (6.43)$$

Формула (6.43) позволяет оценить величину шага, необходимую для достижения заданной точности.

6.2.9. Оценка точности квадратурных формул

Как следует из оценочных формул (6.38) и (6.43), оценка погрешности метода интегрирования по формулам трапеций и Симпсона возможна лишь тогда, когда подынтегральная функция задана аналитически. Однако даже и в этом случае на практике применяется следующий прием, пригодный для каждого из рассмотренных методов интегрирования.

Искомый интеграл вычисляется дважды: при делении отрезка $[a;b]$ на n частей и на $2n$ частей (при интегрировании по формуле Симпсона n должно быть четным). Вслед за этим полученные значения интеграла I_n и I_{2n} сравниваются, и совпадающие первые десятичные знаки считаются верными.

Пусть R_n и R_{2n} – погрешности интегрирования по формуле Симпсона, соответственно при n и $2n$ отрезках разбиения. Учитывая оценку (6.43), можно составить равенство:

$$\frac{R_n}{R_{2n}} = \frac{h_n^4}{h_{2n}^4}, \qquad (6.44)$$

где h_n и h_{2n} – длина отрезков разбиения (шаг интегрирования) в первом и втором случае. Так как $h_{2n} = \frac{h_n}{2}$, тогда из (6.44) получаем:

$$R_n = 16R_{2n}. \qquad (6.45)$$

Если I – истинное значение интеграла, то $I = I_n + R_n$ и $I = I_{2n} + R_{2n}$, откуда $I_n + 16R_{2n} = I_{2n} + R_{2n}$, то есть:

$$|R_{2n}| = \frac{|I_n - I_{2n}|}{15}. \qquad (6.46)$$

Формула (6.46) удобна для практической оценки погрешности метода Симпсона, но требует двойного счета.

Из оценочных формул (6.39) и (6.42) следует, что ошибка интегрирования по методу трапеций и методу Симпсона уменьшается с уменьшением шага интегрирования. При последовательном увеличении числа отрезков разбиения будем получать значение интеграла, все более и более близкое к истинному. Этот вывод имеет теоретическое значение. В процессе практических вычислений при последовательном удвоении числа отрезков разбиения начинает сильно прогрессировать удельный вес ошибки округления, значение которой с некоторого момента ставит предел достижимой точности результата интегрирования.

6.3. Методы решения обыкновенных дифференциальных уравнений

Инженерные и научные задачи часто связаны с решением дифференциальных уравнений, так как с помощью последних описываются многие физические явления. Соответственно, процессы в технических устройствах так же описываются дифференциальными уравнениями.

Природа этих процессов различна. При анализе тепловых режимов аппаратуры рассчитывают тепловые потоки, при изучении электромагнитных процессов – электрические и магнитные поля, при оценке прочности изделий вычисляют механические напряжения и деформации.

К сожалению, для многих практически важных случаев задачи, описываемые дифференциальными уравнениями, весьма сложны и получить их точное решение оказывается затруднительно или невозможно. Эти трудности могут быть связаны с видом уравнения, например, с его нелинейным характером. Однако решить подобные сложные задачи также как и более простые можно с помощью компьютера. Поэтому методы решения дифференциальных уравнений на ЭВМ широко применяются в инженерной практике.

6.3.1. Задача Коши и краевая задача

Методы решения задач, содержащих обыкновенные дифференциальныения, зависят от их математической формулировки. Рассмотрим их.

6.3.1.1. Классификация уравнений

Дифференциальные уравнения принято делить на две группы: *обыкновенные дифференциальные уравнения* и *уравнения в частных производных*.

В данном разделе рассматриваются методы решения задач, описываемых обыкновенными дифференциальными уравнениями. Эти уравнения содержат только одну независимую переменную, в качестве которой может выступать время или пространственная координата. Иначе говоря, в таких уравнениях все функции зависят только от одной переменной и их производные по этой переменной являются полными.

Уравнения в частных производных содержат более одной независимой переменной. Этими переменными могут быть, например, одновременно пространственные координаты и время или только пространственные координаты для статической задачи. В таких уравнениях производные от функций по любой из независимых переменных являются частными. Кроме того, уравнение может содержать смешанные производные.

6.3.1.2. Задача Коши

Важным элементом задач, содержащих дифференциальные уравнения, являются дополнительные условия, которые необходимы для получения количественного решения.

Применительно к обыкновенным дифференциальным уравнениям различают два простых вида задач: *задачу с начальными условиями (задачу Коши) и задачу с краевыми условиями (краевую задачу)*.

Задачу Коши можно сформулировать следующим образом. Дано обыкновенное дифференциальное уравнение:

$$\frac{du(x)}{dx} = f(x,u) \tag{6.47}$$

и начальное условие:

$$u(x_0) = u_0 . \tag{6.48}$$

Требуется найти функцию $u(x)$, удовлетворяющую уравнению (6.47) и начальному условию (6.48).

На практике подобные задачи обычно связаны с расчетом переходных электрических нестационарных тепловых или механических процессов при заданном в некоторый начальный момент времени исходном состоянии системы. Формулировка краевой задачи будет рассмотрена ниже.

6.3.2. Одношаговые методы решения задачи Коши

К одношаговым относится метод Эйлера (первого порядка), его модификация (второго порядка) и методы Рунге – Кутта (более высоких порядков).

6.3.2.1. Метод Эйлера

Метод Эйлера является простейшим численным методом решения задачи Коши. Рассмотрим его на примере решения обыкновенного дифференциального уравнения первого порядка (6.47) с соответствующим начальным условием (6.48).

Расчетную формулу метода Эйлера можно получить, используя разложение функции $u(x)$ в ряд Тейлора в окрестности некоторой точки x_i:

$$u(x_i + h) = u(x_i) + h\frac{du(x_i)}{dx} + \frac{h^2}{2!}\frac{d^2u(x_i)}{dx^2} + \frac{h^3}{3!}\frac{d^3u(x_i)}{dx^3} + \dots \quad (6.49)$$

Если приращение h мало (то есть $h<<x_i$), то члены ряда, начиная со слагаемого, включающего h во второй степени, могут быть отброшены как малые величины. Тогда из (6.49) в первом приближении получим:

$$u(x_i + h) = u(x_i) + h\frac{du(x_i)}{dx}. \quad (6.50)$$

Воспользуемся формулой (6.50), применив ее к единственной известной из условия задачи точке x_0. Найдем в x_0 производную $du(x_0)/dx$, подставив (6.48) в (6.47):

$$\frac{du(x_0)}{dx} = f(x_0, u(x_0)) = f(x_0, u_0).$$

Подставив последнее выражение в (6.50) и полагая, что $x_i = x_0$, получим:

$$u(x_0 + h) = u(x_0) + hf(x_0, u(x_0, u(x_0))),$$

или, сокращая обозначения, в окончательном виде:

$$u_1 = u_0 + hf(x_0, u_0).$$

Таким образом (6.50) при известном значении функции $u_0 = u(x_0)$ в начальной точке x_0 позволяет найти приближенное значение $u_1 = u(x_1)$ при малом смещении h от x_0. На рис. 6.2 графически показан начальный шаг решения методом Эйлера.

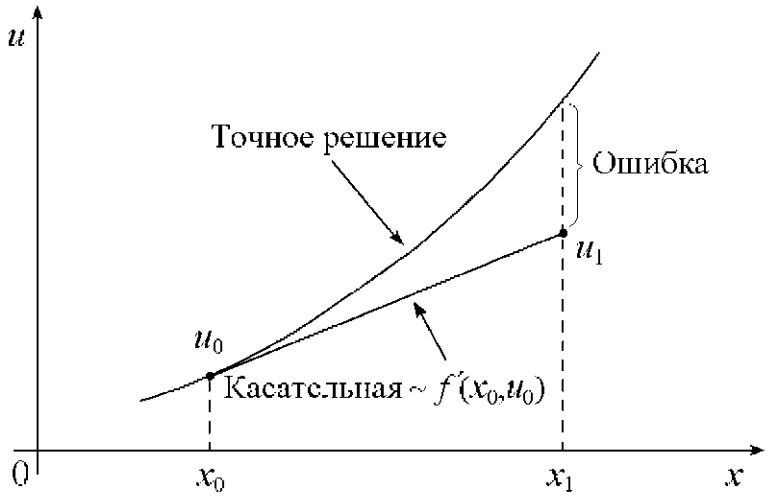


Рис. 6.2. Метод Эйлера

Решение можно продолжить, используя найденное значение функции u_1 для вычисления следующего значения – u_2. Распространяя эти рассуждения на последующие точки, запишем расчетную формулу метода Эйлера в виде:

$$u_{i+1} = u_i + hf(x_i, u_i),\ i = 0,1,2,\dots . \quad (6.51)$$

Из рис. 6.1 видно, что ошибка метода Эйлера на шаге связана с используемой линейной аппроксимацией $u(x)$. Хотя тангенс угла наклона касательной к кривой точного решения в точке (x_0,u_0) известен и равен $du(x_0)/dx$, он изменяется при смещении от x_0 до x_i. Следовательно, при сохранении начального наклона касательной на всем интервале h расчет u_1 выполняется с погрешностью.

Ошибка метода Эйлера на каждом шаге имеет порядок h^2, так как члены, содержащие h во второй и более высоких степенях, отбрасываются – см. (6.49) и (6.50). Уменьшая h можно снизить *локальную ошибку* на шаге.

6.3.2.2. Модифицированный метод Эйлера

Точность метода Эйлера можно существенно повысить, улучшив аппроксимацию $u(x)$ на рассчитываемом шаге. Для этого при разложении $u(x)$ в ряд Тейлора учтем дополнительно слагаемое, содержащее h^2 и $d^2u(x_i)/dx^2$ в (6.49). Определим вторую производную, аппроксимировав ее конечной разностью:

$$\frac{d^2u(x_i)}{dx^2} = \frac{d}{dx}\left(\frac{du(x_i)}{dx}\right) \approx \frac{\Delta u'}{\Delta x} = \frac{u'(x_i+h)-u'(x_i)}{h}. \qquad (6.52)$$

где $\Delta x = h$, $u'(x_i + h) = du(x_i + h)/dx$ и $u'(x_i) = du(x_i)/dx$.

Подставляя полученное выражение в (6.49) и отбрасывая члены ряда, начиная со слагаемого, содержащего h^3, запишем:

$$u(x_i+h) = u(x_i) + \frac{h}{2}\left(\frac{du(x_i)}{dx} + \frac{du(x_i+h)}{dx}\right).$$

Заменяя в последнем выражении производные так же, как это было сделано ранее, и используя сокращенные обозначения, получим расчетную формулу модифицированного метода Эйлера:

$$u_{i+1} = u_i + \frac{h}{2}\left(f(x_i,u_i) + f(x_{i+1},u_{i+1})\right). \qquad (6.53)$$

Соотношение (6.53) дает решение для u_{i+1} в неявном виде, поскольку u_{i+1} присутствует одновременно в левой и правой его частях. Следует отметить, что использование неявных методов оправдано тем, что они, как правило, более устойчивы, чем явные.

Формула (6.53) может рассматриваться и как явное решение, если в ее правую часть подставить значение u^*_{i+1}, рассчитав его предварительно методом Эйлера по формуле (6.51). При этом значение u_{i+1}^* является *прогнозом,* а уточнение результата по формуле

(6.53) – его *коррекцией.* Непосредственная подстановка формулы Эйлера (6.51) в правую часть (6.53) дает расчетное соотношение *метода Эйлера - Коши* (или *метода Хьюна).*

Графически модифицированный метод Эйлера представлен на рис. 6.3. Из рисунке видно, что поправка, учитывающая изменение наклона кривой u(x) заметно уменьшает ошибку на шаге h. Модифицированный метод Эйлера обеспечивает второй порядок точности. Ошибка на каждом шаге при использовании этого метода пропорциональна $h3$. Повышение точности дости гается за счет дополнительных затрат машинного времени при расчете каждого шага.

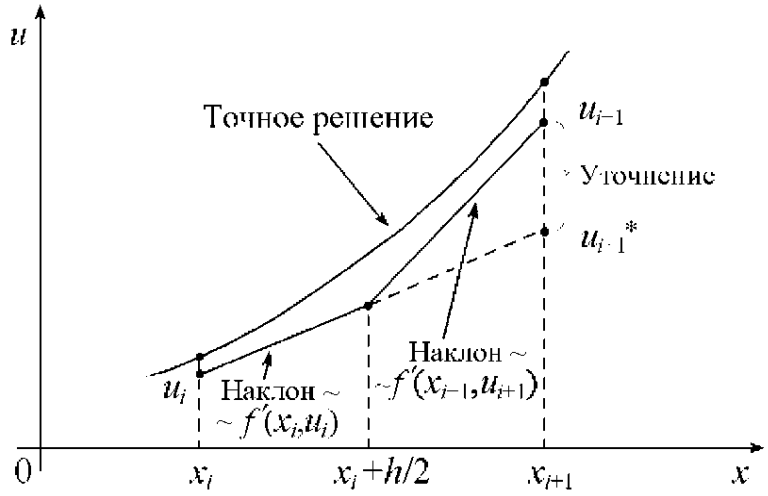


Рис. 6.3. Модифицированный метод Эйлера

Дальнейшее снижение погрешности решения можно получить за счет использования лучшей аппроксимации $u(x)$, учитывающей слагаемые высоких порядков. Эта идея положена в основу методов Рунге – Кутта.

6.3.2.3. Метод Рунге – Кутта четвертого порядка

В модифицированном методе Эйлера для получения второй производной $d^2u(x_i)/dx^2$ используется конечно-разностная формула (6.52), включающая значения первой производной $u'(x)$ и $u'(x_i+h)$ в начальной и конечной точках шага. Если подобным же образом

вычислить третью производную, рассчитав предварительно вторую производную в двух точках шага, то можно с помощью (6.49) построить расчетную формулу метода третьего порядка точности. Для этого потребуется определить первую производную $u'(x)$ в дополнительной промежуточной точке между x_i и $x_i + h$.

Аналогичные рассуждения позволяют вывести расчетные формулы методов более высоких порядков, обеспечивающих заметное снижение погрешности решения. Однако на практике их реализация требует существенного повышения объема вычислений с использованием дополнительных промежуточных точек на каждом шаге.

Существуют и другие способы построения численных методов с высоким порядком точности. Один из них, применяемый при построении группы методов Рунге – Кутта, заключается в аппроксимации решения дифференциального уравнения суммой:

$$u(x_i + h) \approx \xi(x_i, h) = u(x_i) + \sum_{n=1}^{p} A_n k_n(h), \qquad (6.54)$$

где A_n – коэффициенты разложения, k_n – последовательность функций:

$$\begin{gathered} k_1 = hf(x_i, u_i); \\ k_2 = hf(x_i + \alpha_2 h, \quad u_i + \beta_{21} k_1); \\ k_3 = hf(x_i + \alpha_3 h, \quad u_i + \beta_{31} k_1 + \beta_{32} k_2); \\ \dots \\ k_3 = hf(x_i + \alpha_3 h, \quad u_i + \beta_{31} k_1 + \beta_{32} k_2), \end{gathered} \qquad (6.55)$$

где α_n, β_{nm}, $0 < m < n < p$ – некоторые параметры.

Неизвестные параметры A_n, α_n, β_{nm} можно выбрать из условия:

$$\psi(0) = \psi'(0) = \psi''(0) = \dots \psi^{(K)}(0) = 0, \qquad (6.56)$$

где функция $\psi(h) = u(x_i + h) - \xi(x_i, h)$ показывает отклонение приближенного решения $\xi(x_i, h)$ от точного $u(x_i + h)$. Увеличение параметра p в (6.54) позволяет сделать погрешность, связанную с заменой точного решения приближенным, как угодно малой.

Предположим, что $p = 1$. Тогда, подставляя (6.54) в (6.56), из условия $\psi(0) = \psi'(0) = 0$ получим $A_1 = 1$ и $\psi''(0) = 0$, откуда:

$$u(x_i + h) \approx u(x_i) + \sum_{n=1}^{p} A_n k_n(h) = u_i + k_1 = u_i + hf(x_i, u_i),$$

что соответствует формуле Эйлера (6.51). Таким же образом можно получить формулы более высоких порядков точности, которые называют методами Рунге – Кутта.

Одним из наиболее известных является вариант метода Рунге – Кутта, соответствующий $p = 4$. Это метод четвертого порядка точности, для которого ошибка на шаге имеет порядок h^5. Его расчетные формулы имеют следующий вид:

$$u_{i+1} = u_i + \frac{k_1 + 2k_2 + 2k_3 + k_4}{6},$$

где

$$\begin{gathered} k_1 = hf(x_i, u_i); \\ k_2 = hf\left(x_i + \frac{h}{2}, u_i + \frac{k_1}{2}\right); \\ k_3 = hf\left(x_i + \frac{h}{2}, u_i + \frac{k_2}{2}\right); \\ k_4 = hf(x_i + h, u_i + k_3). \end{gathered}$$

Рассмотренные выше метод Эйлера и его модификация по сути дела являются методами Рунге – Кутта первого и второго порядка соответственно. Несмотря на увеличение объема

вычислений метод четвертого порядка имеет преимущество перед методами первого и второго порядков, так как он обеспечивает малую локальную ошибку. Это позволяет увеличивать шаг интегрирования h и, следовательно, сокращать время расчета.

6.3.2.4. Погрешность решения и выбор шага

Как было показано выше, порядок точности метода p определяет ошибку дискретизации $\sim h^{p+1}$. Знание порядка ошибки не обеспечивает ее прямую оценку. Получить такую оценку позволяет правило Рунге (формула двойного пересчета).

Пусть одношаговый метод имеет порядок точности p. Тогда погрешность, равная разности точного решения u_T и приближенного $u_{i+1,h}$, полученного численно с использованием шага h, имеет порядок $p + 1$:

$$u_T - u_{i+1,h} \approx Ch^{p+1},$$

где C – константа, не зависящая от h.

При расчете с уменьшенным вдвое шагом, равным $h/2$, погрешность изменится:

$$u_T - u_{i+1,h/2} \approx C\left(\frac{h}{2}\right)^{p+1}.$$

Вычитая последнее выражение из предыдущего, определим изменение:

$$u_{i+1,h/2} - u_{i+1,h} \approx Ch^{p+1} - C\left(\frac{h}{2}\right)^{p+1} = C\left(\frac{h}{2}\right)^{p+1}\left(2^{p+1}-1\right).$$

Выражая из последнего соотношения постоянную C и подставляя в предыдущую формулу, получим оценку погрешности по правилу Рунге:

$$u_T - u_{i+1,h/2} \approx \frac{u_{i+1,h/2} - u_{i+1,h}}{2^{p+1}-1}. \qquad (6.57)$$

Ошибка дискретизации стремится к нулю при стремлении h к нулю. Следовательно, уменьшая шаг h можно сделать *локальную ошибку* (на шаге) сколь угодно малой. Однако при уменьшении h необходимо увеличить количество шагов. Поэтому сокращение h не приводит к такому же снижению *глобальной* (накапливаемой от шага к шагу) ошибки.

Малые ошибки, появившиеся в начале вычислений, могут совершенно исказить решение, если только не подобрать подходящий численный метод. Это явление иногда называют *«неустойчивостью»*. Неустойчивость проявляется в катастрофическом нарастании погрешности решения вплоть до возникновения паразитной осцилляции кривой решения.

На практике уменьшению h препятствуют и ошибки округления, вызванные неточностью представления чисел в компьютере. При уменьшении шага, начиная с некоторого h_0, вклад ошибок округления преобладает, что приводит к возрастанию погрешности решения.

Обычно алгоритмы обеспечивают автоматический выбор шага. Для этого выполняется два пробных расчета – с заданным шагом h и с уменьшенным вдвое $h/2$.

В простейшем случае ограничиваются сравнением результатов решений в одной и той же точке:

$$\left|u_{i+1,h/2} - u_{i+1,h}\right| < \delta,$$

где δ – некоторое малое положительное число, определяющее требования к точности. Более сложные оценки основываются на формулах подобных правилу Рунге. Если оценка показывает большую ошибку, алгоритм переходит на уменьшенный вдвое шаг.

6.3.3. Многошаговые методы решения задачи Коши

6.3.3.1. Многошаговые методы

Из вышеизложенного видно, что снижение погрешности решения задачи Коши может быть обеспечено использованием одношаговых методов высоких порядков точности. При этом в пределах каждого шага интегрирования приходится вводить промежуточные точки и увеличивать объем вычислений.

Снизить вычислительные затраты без ухудшения погрешности можно, если на очередном шаге уточняющую информацию получать не за счет дополнительных точек, а из предыдущих шагов. Действительно, если в расчете использовать не только последнюю из известных точек решения, а еще и ряд предыдущих, можно более точно предсказать дальнейший ход кривой. Методы, реализующие эту идею, получили название *многошаговых.*

6.3.3.2. Метод Адамса

В простейшем случае многошаговый метод опирается только на две последние точки решения – (x_i, u_i) и (x_i, u_i). Вычисление следующей точки строится на двух интервалах – от x_{i-1} до x_i и от x_i до x_{i+1}. В данном случае говорят, что метод является *двухшаговым.*

Для получения расчетной формулы двухшагового метода проинтегрируем обе части дифференциального уравнения (1) на интервале от x_i до x_{i+1}. Интегрирование левой части дает:

$$\int_{x_i}^{x_{i+1}} \frac{du(x)}{dx}dx = u(x_{i+1}) - u(x_i) \approx u_{i+1} - u_i, \qquad (6.58)$$

Для интегрирования правой части (6.47) заменим $f(x, u) = f(x, u(x))$ на интерполяционный многочлен $F(x)$. Для двух известных точек x_{i-1} и x_i может быть построен линейный многочлен, совпадающий с кривой решения в точках (x_{i-1}, u_i) и (x_i, u_i):

$$F(x) = \frac{x - x_{i-1}}{h} f(x_i, u_i) - \frac{x - x_i}{h} f(x_{i-1}, u_{i-1}).$$

Тогда интегрирование правой части дает:

$$\int_{x_i}^{x_{i+1}} F(x)dx = \frac{h}{2}\left(3f(x_i, u_i) - f(x_{i-1}, u_{i-1})\right). \qquad (6.59)$$

Приравнивая правые части (6.58) и (6.59) и применяя сокращенные обозначения $f_i = f(x_i, u_i)$, $f_{i+1} = f(x_{i+1}, u_{i+1})$, запишем формулу двухшагового метода

$$u_{i+1} = u_i + \frac{h}{2}(3f_i - f_{i-1}).$$

Аналогично, учитывая большее число предыдущих точек решения можно построить формулы экстраполяционного метода Адамса – Башфорта:

$$u_{i+1} = u_i + \frac{h}{12}(23f_i - 16f_{i-1} + 5f_{i-2});$$

$$u_{i+1} = u_i + 24(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}).$$

Первая формула соответствует *трехшаговому,* а вторая – *четырехшаговому* методу.

6.3.3.3. Методы прогноза и коррекции (предиктор-корректор)

Аналогичные вышеприведенным выражения можно получить, включая в интерполяционный многочлен рассчитываемую точку x_{i+1} как известную. Этот подход дает формулы метода Адамса – Моултона:

$$u_{i+1} = u_i + \frac{h}{2}(f_{i+1} + f_{i-1});$$

$$u_{i+1} = u_i + \frac{h}{12}(5f_{i+1} + 8f_i - f_{i-1});$$

$$u_{i+1} = u_i + \frac{h}{12}(9f_{i+1} + 19f_i - 5f_{i-1} + f_{i-2}).$$

Данный метод является неявным, то есть требует решения уравнения относительно u_{i+1}, что представляется неудобным. Тем не менее, неявные формулы применяются на практике, так как позволяют повысить устойчивость решения и существенно увеличить шаг.

Обычно решение строится в два этапа. Сначала по явной схеме определяют *прогноз* u_{i+1}, например, по формуле Адамса – Башфорта. На втором этапе производится *коррекция* u_{i+1} по неявной формуле. Далее, многократно используя неявную формулу, можно дополнительно уточнять u_{i+1} подобно тому, как это делается в методе простой итерации. Но обычно ограничиваются единственной итерацией. Описанный алгоритм многошаговых методов получил название *метода прогноза и коррекции.*

6.3.3.4. Общая характеристика многошаговых методов

По сравнению с одношаговыми методами многошаговые методы требуют меньшего объема вычислений (сравните формулы Рунге – Кутта и Адамса – Башфорта). К недостаткам многошаговых методов относится трудность смены шага, так как расчетные формулы не учитывают эту возможность. Данная проблема решена в многозначных методах, использующих иную экстраполяцию решения.

Кроме того, многошаговые методы не обладают свойством *самостартования,* так как требуют задания ряда предыдущих точек. В то же время любой из одношаговых методов способен стартовать из единственной начальной точки.

6.3.4. Краевая задача и метод стрельбы

6.3.4.1. Краевая задача

В краевой задаче требуется найти решение обыкновенного дифференциального уравнения с дополнительными условиями, заданными при нескольких различных значениях независимой переменной.

Например, для обыкновенного дифференциального уравнения второго порядка:

$$\frac{d^2u(x)}{dx^2} = f\left(x, u, \frac{du}{dx}\right), \qquad (6.60)$$

эти условия задаются в двух разных точках x = a и x = b:

$$u(a) = A, \text{ и } u(b) = B. \qquad (6.61)$$

Поскольку эти точки определяют границы области a < x < b, в которой обычно и отыскивается решение, поставленные в них дополнительные условия называют граничными или краевыми. В инженерной практике, например, такая задача может быть связана с расчетом прогиба стержня при заданном способе закрепления его концов.

Одним из методов решения краевой задачи является метод конечных разностей. Этот метод будет рассмотрен в следующей лабораторной работе. Другим способом решения краевой задачи является ее приведение к задаче Коши путем замены дополнительных условий. Рассмотрим метод стрельбы, использующий этот подход.

6.3.4.2. Метод стрельбы

Итак, если дана краевая задача, например, в вышеприведенной формулировке, то в методе стрельбы она заменяется задачей Коши для того же уравнения (6.60) но с начальными условиями:

$$u(a)=A, \quad \frac{du}{dx} = k = tg\theta \qquad (6.62)$$

Здесь $u(a)$ – точка, которая является началом кривой решения $u(x)$ дифференциального уравнения, θ – угол наклона касательной к этой кривой в начальной точке.

Считая решение задачи Коши зависящим от начального условия $\frac{du}{dx} = tg\theta$, будем подбирать такое значение θ, при котором кривая решения $u(x)$ в точке b даст совпадающий с (6.61) результат $u(b) = B$. Если это условие будет выполнено, то решение задачи Коши совпадет с решением краевой задачи.

Применительно к описанному подходу называние «метод стрельбы» вполне оправдано, поскольку в нем производится как бы «пристрелка» по углу наклона кривой $u(x)$ в начальной точке.

Чтобы сократить количество попыток при поиске решения $u(x)$, применяют различные стратегии подбора параметра θ. Например, при использовании метода половинного деления действуют следующим образом. Вначале выполняют два пробных расчета при значе-

ниях параметра θ равных θ_1 и θ_2. Эти значения выбирают таким образом, чтобы при $\theta = \theta_1$ решение давало в точке $x = b$ «перелет», то есть $u(b) > B$, а при $\theta = \theta_2$ – «недолет», то есть $u(b) < B$.

Далее, используя в начальном условии (6.61) значение $\theta_1 = \frac{\theta_1 + \theta_2}{2}$, вновь численно решают задачу Коши. Из трех полученных решений отбрасывают то, которое дает в точке $x = b$ наибольшее отклонение от B. Затем от двух оставшихся значений параметра θ находят среднее θ_4 и вновь выполняют с этим значением расчет.

Повторение описанного процесса прекращают, когда разность двух последовательно найденных значений θ станет меньше некоторого заданного малого числа или достаточно малым будет отклонение $u(b)$ от B. Подобный алгоритм может быть построен и с использованием метода Ньютона.

6.3.4.3. Метод стрельбы для линейного дифференциального уравнения

Если обыкновенное дифференциальное уравнение второго порядка является линейным, при граничных условиях:

$$\frac{d^2u(x)}{dx^2} = f_1(x)\frac{du}{dx} + f_2(x)u + f_3(x), \qquad (6.63)$$

то поиск решения методом стрельбы существенно упрощается.

Выполнив два «пристрелочных» расчета при θ_1 и θ_2, как это было описано ранее, получим два решения $u_1(x)$ и $u_2(x)$. Если $u_1(b) = B_1$ и $u_2(b) = B_2$, причем $B_1 \neq B_2$, то решением краевой задачи будет линейная комбинация двух решений:

$$u(x) = \frac{1}{B_1 - B_2}\left((B - B_2)u_1(x) + (B_1 - B)u_2(x)\right). \qquad (6.64)$$

Подставляя в это выражение при $x = a$ значения $u_1(a) = A$ и при $x = b$ значения $u_1(b) = B_1$, $u_2(b) = B_2$ нетрудно убедиться, что оно удовлетворяет обоим исходным граничным условиям задачи.

6.4. Решение дифференциальных уравнений в частных производных

6.4.1. Краткие теоретические сведения

Большое число задач, связанных с анализом физических (и не только физических) полей описываются дифференциальными уравнениями в частных производных. К сожалению, во многих случаях, представляющих практический интерес, найти аналитическое решение таких задач трудно или практически невозможно. Это обычно обусловлено сложной формой или неоднородностью свойств области, в которой отыскивается решение.

Однако результат можно получить численно с помощью компьютера. Подходы к решению дифференциальных уравнений с частными производными определяются их математической формой. Поэтому рассмотрим классификацию уравнений с этой точки зрения.

6.4.2. Классификация уравнений по математической форме

Во многих случаях для описания физических процессов используют уравнения с частными производными до второго порядка включительно.

Так, например, изучение свободных колебаний различной природы приводит к *волновым уравнениям* вида:

$$\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) - \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = 0, \qquad (6.65)$$

где $u(x,y,z,t)$ – функция, описывающая волновой процесс;
x, y, z – координаты;
c – скорость распространения волны в данной среде;
t – время.

Оператор $\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}+\frac{\partial^2}{\partial z^2}\right)$ принято обозначать знаком ∇, который в этом случае носит название оператора Лапласа.

Процессы распространения тепловой энергии описываются *уравнением теплопроводности*

$$\rho C\frac{\partial T}{\partial t}-k\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right)=Q, \tag{6.66}$$

где ρ и C – плотность и теплоемкость вещества;
T – температура;
k – коэффициент теплопроводности;
Q – плотность источников тепла.

Анализ стационарных состояний, например, статических тепловых, электрических, магнитных полей или деформаций при статических нагрузках проводят, используя *уравнение Пуассона*:

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=-f(x,u,z), \tag{6.67}$$

где $u(x,y,z)$ – функция, описывающая статическое поле;
(x,y,z) – распределенные источники.

Если $(x,y,z) = 0$, то (6.67) обращается в *уравнение Лапласа:*

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=0. \tag{6.68}$$

Известны и другие виды задач и соответствующие им дифференциальные уравнения в частных производных, например, уравнение диффузии или уравнение Гельмгольца.

Несмотря на различие процессов, описываемых рассмотренными уравнениями, и форм их записи, все они с математической точки зрения могут быть представлены как частные случаи обобщенной формы дифференциального уравнения второго порядка.

Рассмотрим уравнение второго порядка с двумя независимыми переменными *x* и *y:*

$$A\frac{\partial^2 u}{\partial x^2}+2B\frac{\partial^2 u}{\partial y^2}+C\frac{\partial^2 u}{\partial z^2}+D=0, \tag{6.69}$$

где A, B, C и D – некоторые функции, зависящие в общем случае от x, y, u, *∂u/∂x* и *∂u/∂y,* причем A, B и C одновременно не обращаются в ноль.

Дифференциальные уравнения, описывающие физические поля, могут быть нелинейными. Однако на практике многие задачи рассматриваются в линейном приближении, когда уравнение с частными производными линейно относительно неизвестной функции *u* и ее частных производных.

На основании того, что уравнению (6.69) можно поставить в соответствие квадратичную форму $A\zeta_1^2+B\zeta_1\zeta_2+C\zeta_2^2=0$, по математической природе различают следующие типы квазилинейных уравнений:

– *гиперболический,* если $B^2 - 4AC > 0$ – его аналогом является волновое уравнение (6.65);

– *параболический,* если $B^2 - 4AC = 0$ – его аналог уравнение теплопроводности (6.66);

– *эллиптический,* если $B^2 - 4AC < 0$ – аналог уравнение Пуассона (6.67) или Лапласа (6.68).

В задачах, описываемых дифференциальными уравнениями в частных производных, другой важной составляющей помимо самого уравнения является формулировка *дополнительных условий.*

Для задач с уравнениями гиперболического или параболического типа, содержащих в качестве независимой переменной время *t,* условия по *t* обычно формулируются как *начальные,* описывающие исходное состояние системы. По координатам *x, y* и *z* задают *граничные условия.* В тепловых задачах они, например, описывают распределение температуры на границе расчетной области. В задачах с уравнениями эллиптического типа, не содержащими переменную *t*, используют только граничные условия по координатам *x, y* и *z*, а саму задачу называют *краевой.*

Если краевое условие задает распределение функции *u* на границе, то его принято называть *условием Дирихле.* Условие, определяющее производную $\vec{n}\overrightarrow{grad(u)}\equiv\vec{n}\vec{\nabla}u$ на границе расчетной области,

называют *условием Неймана.* Здесь $\vec{n}$ – единичная нормаль к границе. Условия, представляющие собой комбинацию двух вышеназванных, называют смешанными.

С помощью дифференциальных уравнений формулируют и другой вид задач – задачи на собственные значения, связанные, например, с определением собственных волн (частот) колебательных систем или волноведущих структур.

Приведенная классификация позволяет определить общие подходы к решению дифференциальных уравнений в задачах различных по физической сути, но сходных с математической точки зрения. В настоящее время широкое распространение получил *метод конечных разностей* и *метод конечных элементов,* основы которых и будут рассмотрены ниже.

6.4.3. Основы метода конечных разностей

Метод конечных разностей заключается в том, что дифференциальное уравнение в частных производных заменяется соответствующей ему системой алгебраических уравнений. Решение этой системы дает приближенное решение для искомой функции $u(x, y, z, t)$.

Метод включает следующие основные этапы:

1. построение *сетки,* охватывающей рассматриваемую область, например, элемент конструкции какого-нибудь устройства;
2. построение на полученной сетке *конечно-разностной аппроксимации,* эквивалентной исходному дифференциальному уравнению и дополнительным условиям;
3. формирование на основе конечно-разностной аппроксимации системы алгебраических уравнений и ее решение.

Рассмотрим перечисленные этапы на примере двухмерных задач.

6.4.3.1. Построение сетки

Формирование сетки производится с учетом геометрии задачи, например, формы детали, для которой выполняется расчет. Обычно для деталей, имеющих прямоугольную форму, используют декартову систему координат и соответственно прямоугольную сетку.

На рис. 6.4 приведен пример такой двухмерной сетки, нанесенной на прямоугольную пластину.

В методе конечных разностей применяют и другие виды сеток. Например, если исследуемая конструкция содержит элементы с осевой симметрией, используют полярную сетку.

В дальнейшем решение задачи строят, опираясь на *узлы сетки,* то есть на точки пересечения ее линий (рис. 6.4).

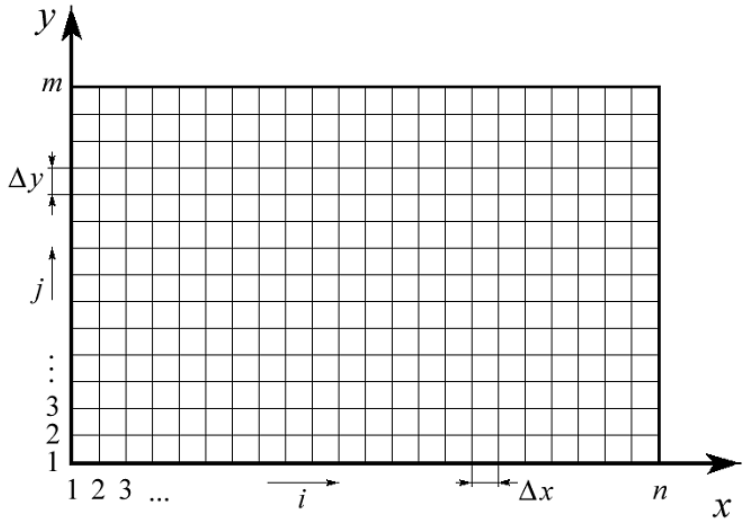


Рис. 6.4. Прямоугольная сетка

Конечно-разностная аппроксимация производных в дифференциальном уравнении строится путем замены этих производных на их приближенные аналоги с помощью сетки. Так, например, частную производную $\frac{\partial u}{\partial x} = \lim_{\Delta x \to 0} \frac{u(x+\Delta x) - u(x)}{\Delta x}$ в точке (x_i, y_i) можно заменить приближенным значением *правой производной:*

$$\frac{\partial u}{\partial x} \approx \frac{\Delta u}{\Delta x} = \frac{u_{i+1} - u_i}{x_{i+1} - x_i} = \frac{u_{i+1} - u_i}{\Delta x}, \qquad (6.70)$$

или левой производной:

$$\frac{\partial u}{\partial x} \approx \frac{\Delta u}{\Delta x} = \frac{u_i - u_{i-1}}{x_i - x_{i-1}} = \frac{u_i - u_{i-1}}{\Delta x}, \qquad (6.71)$$

где Δu и $\Delta x -$ приращения функции и аргумента, u_i, x_i и u_{i+1}, x_{i+1} – значения функции и аргумента в узлах i и i+1, причем Δx – шаг сетки по координате x. Аналогично получается формула для второй производной $\partial^2 u/\partial x^2$:

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) = \frac{\dfrac{u_{i+1} - u_i}{\Delta x} - \dfrac{u_i - u_{i-1}}{\Delta x}}{\Delta x} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}. \qquad (6.72)$$

В полученных выражениях в отличие от точных производных используются малые, но не бесконечно малые разности Δu и Δx. Поэтому сам метод и получил название *метода конечных разностей*. Формулы для производных по независимым переменным y, *z, t* получают аналогично.

6.4.3.2. Аппроксимация уравнения эллиптического типа

Преобразование уравнения эллиптического типа (3) для двухмерной задачи (когда $\dfrac{\partial^2 u}{\partial z^2} \equiv 0$ производится путем замены в нем производных $\dfrac{\partial^2 u}{\partial x^2}$ и $\dfrac{\partial^2 u}{\partial y^2}$ конечно-разностными формулами. Заменив в (6.37) $\dfrac{\partial^2 u}{\partial x^2}$ с помощью (6.72) и используя аналогичное выражение для $\dfrac{\partial^2 u}{\partial y^2}$, получим:

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = -\frac{P}{W}, \qquad (6.73)$$

где индексы i и j отсчитываются соответственно по осям X и Y.

Для упрощения анализа предположим, что в сетке используются квадратные ячейки, то есть $\Delta x = \Delta y = h \neq 0$, тогда:

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j+1} = -\frac{P}{W}h^2. \qquad (6.74)$$

Уравнение (10) связывает между собой неизвестное значение функции $u_{i,j}$ с ее значениями в четырех соседних узлах. На сетке эти узлы образуют пятиточечный *шаблон* (рис. 6.5), позволяющий легко определить индексы в (6.74) для любого произвольно выбранного на сетке узла i, j .

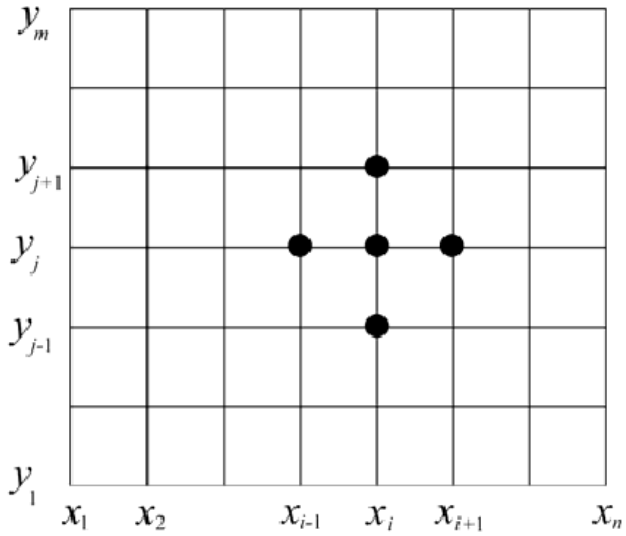


Рис. 6.5. Шаблон «крест» для уравнения эллиптического типа

Записывая (6.74) для каждого узла $2<i<n-1$, $2<j<m-1$ и подставляя вместо i и j соответствующие номера, получим систему связанных уравнений. Количество уравнений будет равно количеству узлов, в которых необходимо найти неизвестные $u_{i,i}$.Число неизвестных равно числу уравнений и система будет замкнутой. Значения функции u в узлах сетки, лежащих на границе рассматриваемой области, определяются заданными граничными условиями.

Решение системы алгебраических уравнений, получаемой в результате конечно-разностной аппроксимации уравнения

эллиптического типа, является одним из наиболее тяжелых по вычислительным затратам этапов расчета. Для повышения точности решения приходится использовать сетки с большим числом узлов, на которых формируются и довольно большие системы − нередко до нескольких тысяч алгебраических уравнений. Одним из способов уменьшения числа узлов иявляется использование сеток с неравномерным шагом. При этом сетку сгущают в наиболее важных с точки зрения точности участках, например, вблизи углов или отверстий.

В то же время решение задачи облегчается тем, что каждое из алгебраических уравнений содержит небольшое количество неизвестных. В качестве примера ниже приведена система с разреженной матрицей ленточного типа, полученной из (6.74) для прямоугольной области (рис. 6.5) при $n = m = 5$. В правой части записаны u_{ij} относящиеся к узлам, лежащим на границах.

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_{22} \\ u_{23} \\ u_{24} \\ u_{32} \\ u_{33} \\ u_{34} \\ u_{42} \\ u_{43} \\ u_{44} \end{bmatrix} = \begin{bmatrix} -Ph^2/W - u_{21} - u_{12} \\ -Ph^2/W - u_{13} \\ -Ph^2/W - u_{14} - u_{25} \\ -Ph^2/W - u_{31} \\ -Ph^2/W \\ -Ph^2/W - u_{35} \\ -Ph^2/W - u_{41} - u_{52} \\ -Ph^2/W - u_{53} \\ -Ph^2/W - u_{45} - u_{54} \end{bmatrix}$$

Для решения подобных систем используют специальные методы, учитывающие *разреженность матрицы* коэффициентов. К специальным прямым относятся некоторые матричные методы и метод прогонки (аналог метода Гаусса). Из итерационных применяют метод Якоби (одновременных смещений) и метод Гаусса – Зейделя (последовательных смещений), а также модификации последнего, например, метод верхней релаксации.

6.4.3.3. Аппроксимация уравнения гиперболического типа

Построение алгебраических уравнений на основе дифференциального уравнения гиперболического типа выполняется так же, как и в предыдущем случае, то есть заменой производных конечно-разностными аналогами. В качестве примера рассмотрим задачу о продольных колебаниях тонкого однородного стержня длиной L, когда его деформация u зависит только от продольной (вдоль оси стержня) координаты x и времени t.

Колебания стержня описываются дифференциальным уравнением

$$\frac{\mathrm{d}^2u}{\mathrm{d}x^2} - \frac{1}{a^2}\frac{\mathrm{d}^2u}{\mathrm{d}t^2} = 0, \qquad (6.75)$$

где $a = \sqrt{\dfrac{E}{\rho}}$, E и ρ – модуль упругости и плотность материала стержня.

Аппроксимация уравнения производится на сетке в координатах t и x. Примерный вид сетки показан на рис. 6.6. Данная задача не имеет верхней границы по координате t. Это объясняется тем, что, с формальной точки зрения, колебания в стержне могут продолжаться неопределенно долгое время, даже если будут учтены потери, приводящие к их затуханию.

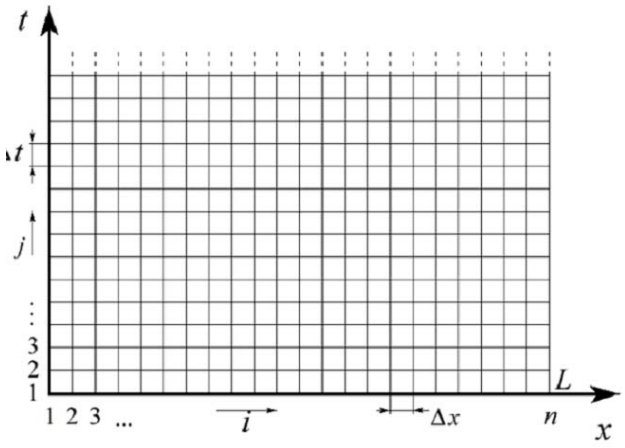


Рис. 6.6. Сетка в координатах t и x

Используя сетку, запишем в конечных разностях уравнение, эквивалентное (6.75):

$$\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{\Delta t^2}=a^2\,\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}, \qquad (6.76)$$

или:

$$u_{i,j+1}=2\left(1-\beta^2\right)u_{i,j}+\beta^2\left(u_{i+1,j}+u_{i-1,j}\right)-u_{i,j-1}, \qquad (6.77)$$

где $\beta=a\Delta t/\Delta x$.

Из (6.76) и (6.77) видно, что форма шаблона уравнения гиперболического типа подобна форме шаблона уравнения эллиптического типа.

Аналогично предыдущей задаче запишем уравнение (6.77) для каждого узла сетки и, подставляя в него вместо *i* и *j* соответствующие этим узлам номера, получим систему связанных алгебраических уравнений.

В качестве граничных условий по *x* в данной задаче могут использоваться любые условия, описывающие способ закрепления стержня. Например, жесткое закрепление предполагает нулевой сдвиг на концах стержня. Это соответствует условию $u(x=0,t)=0$ и $u(x=L,t)=0$, где $x=0$ и $x=L$ – координаты концов стержня.

По времени *t* в качестве начальных условий зададим при $t=0$ исходную деформацию стержня и начальную скорость его колебаний:

$$u\left(x,t=0\right)=f_{Д}\left(x\right). \qquad (6.78)$$

Решение системы уравнений для рассматриваемой задачи можно получить с помощью сравнительно простой процедуры, называемой *явной схемой.* Эта схема строится на том, что все уравнения системы последовательно связаны между собой.

Расчет будем проводить в следующем порядке. Вначале определим деформацию стержня в моменты $t=0$ и $t=0+\Delta t$. Для $t=0$ деформация $u(x,0)\equiv u_{i,1}$ известна из заданных начальных условий (6.78). Для следующего момента времени $t=\Delta t$ деформацию

$u(x,\Delta t)\equiv u_{i,2}$ определим с помощью второго начального условия, задающего скорость $\frac{du}{dt}$ при $t=0$:

$$\frac{du}{dt}=v\left(x\right)\Rightarrow\frac{u_{i,2}-u_{i,1}}{\Delta t}=v_{i,1},\ \text{тогда}\ u_{i,2}=u_{i,1}+v_{i,1}\Delta t. \qquad (6.79)$$

При известных из (6.78) и (6.79) $u_{i,1}$ и $u_{i,2}$ начнем решение задачи следующим образом. Полагая, что $j=2$, то есть $u_{i,j-1}=u_{i,1}$ и $u_{t,j}=u_{i,2}$, подставим в (6.77) известную из (6.78), соответствующую $t=0$ начальную деформацию $u_{i,1}\equiv u(x,t=0)=f_{Д}(x)$ и соответствующую $t=\Delta t$ деформацию $u_{i,2}=u_{i,1}+v_{i,1}\Delta t$ (см. (6.78)). Вычисление правой части (6.77) позволяет определить $u_{i,j+1}=u_{i,3}$ в момент времени $t=2\Delta t$.

Далее действуя аналогично и сдвигая шаблон решения на одну линию сетки по координате *t,* вычисляются последовательно фазы колебаний $u_{i,4}$ – из $u_{i,2}$ и $u_{i,3}$, затем $u_{i,5}$ – из $u_{i,3}$ и $u_{i,4}$ и так далее. То есть очередной временной *слой* $j+1$ рассчитывается из предыдущих с индексами *j* и $j-1$.

При решении гиперболического уравнения следует обращать внимание на выбор шага сетки по *x* и *t.* Теоретически можно показать, что приближенное решение, получаемое с помощью (6.77), сходится к точному при $\Delta x\to 0$ и $\Delta t\to 0$ со скоростью $O(\Delta x^2+\Delta t^2)$ если $\beta=\frac{a\Delta t}{\Delta x}<1$. Иначе говоря, если выбран шаг сетки Δx по координате *x,* то появляется ограничение на шаг по времени Δt.

При $\beta>1$ метод становится неустойчивым как в абсолютном, так и в относительном смысле. Последнее означает, что по мере продолжения вычислений ошибки катастрофически нарастают. Теоретически показано, что при $\beta=1$ метод устойчив и конечно-разностное решение совпадает с точным. При $\beta<1$ решение хотя и устойчиво, но его точность с уменьшением β убывает.

6.4.3.4. Аппроксимация уравнения параболического типа

Решение двухмерной задачи с уравнением параболического типа (6.66) выполняется с помощью сетки, аналогичной приведенной на рис. 6.6.

Рассмотрим процесс теплопередачи по длинному однородному стержню длиной L, ось которого совпадает с осью x. Предположим, что в исходном состоянии стержень по всей длине имеет температуру $T = T_0$. Затем, начиная с момента времени $t = 0$ температура на его правом конце $x = L$ скачком возрастает до T_L, в то время как на левом конце $x = 0$ поддерживается температура $T = T_0$. Теплопередачей через боковую поверхность стержня будем пренебрегать.

Учитывая, что в стержне отсутствуют источники тепла ($Q = 0$), запишем в конечных разностях уравнение эквивалентное (6.66):

$$\frac{T_{i,j+1} - T_{i,j}}{\Delta t} = \frac{k}{\rho C}\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} \qquad (6.80)$$

или

$$T_{i,j+1} = \beta T_{i+1,j} + (1 - 2\beta)T_{i,j} + \beta T_{i-1,j}, \qquad (6.81)$$

где $\beta = \frac{k}{\rho C}\frac{\Delta t}{\Delta x^2}$.

Из (6.80) и (6.81) видно, что шаблон для уравнения параболического типа напоминает перевернутую букву Т.

Граничные условия по координате x в данной задаче включают температуру на концах стержня: $T_{1j} = 0$ при $x = 0$ и $T_{nj} = T_L$ при $x = L$. По времени t начальное условие задает исходное распределение температуры в стержне $T(x, t = 0) = T_0$.

Запишем уравнение (6.81) для каждого узла сетки и, подставляя в него вместо i и j соответствующие этим узлам номера, получим систему связанных уравнений.

Решение системы уравнений для данной задачи так же, как и в предыдущем случае вычисляется с использованием *явной схемы.*

При этом расчет упрощается за счет того, что распределение температуры в стержне для каждого последующего временного

слоя $j + 1$ определяется из известного распределения только в одном предыдущем слое j.

При решении уравнения параболического типа также важен выбор шага Δt. Для обеспечения сходимости и устойчивости метода желательно, чтобы параметр $\beta = \frac{k}{\rho C}\frac{\Delta t}{\Delta x^2}$ в (6.81) не превышал 0,5. Нарушение этого условия приводит к расходящемуся или колеблющемуся решению.

6.4.3.5. Погрешность решения

Погрешность решения методом конечных разностей в первую очередь определяется ошибкой, вносимой при замене исходного дифференциального уравнения на его конечно-разностный аналог.

Вначале оценим погрешность аппроксимации (6.70) для первой производной, используя разложение $u(x)$ в окрестностях точки x_i в ряд Тейлора:

$$u(x_i + \Delta x) = u(x_i) + \Delta x\frac{\partial u(x_i)}{\partial x} + \frac{\Delta^2 x}{2!}\frac{\partial^2 u(x_i)}{\partial x^2} + \frac{\Delta^3 x}{3!}\frac{\partial^3 u(x_i)}{\partial x^3} + \frac{\Delta^4 x}{4!}\frac{\partial^4 u(x_i)}{\partial x^4} + \ldots, \qquad (6.82)$$

откуда:

$$\frac{\partial u(x_i)}{\partial x} = \frac{u(x_i + \Delta x) - u(x_i)}{\Delta x} - \frac{\Delta^2 x}{2!}\frac{\partial^2 u(x_i)}{\partial x^2} - \frac{\Delta^3 x}{3!}\frac{\partial^3 u(x_i)}{\partial x^3} - \frac{\Delta^4 x}{4!}\frac{\partial^4 u(x_i)}{\partial x^4} - \ldots \qquad (6.83)$$

Согласно (6.82) погрешность конечно-разностной аппроксимация по формуле (6.70) обусловлена тем, что в ней не учитываются слагаемые высоких порядков, начиная с $\frac{\Delta^2 x}{2!}\frac{\partial^2 u(x_i)}{\partial x^2}$. Можно утверждать, что в (6.83) слагаемые убывают по мере увеличения их порядка. Поэтому ошибка (6.70) приближенно равна $\frac{\Delta^2 x}{2!}\frac{\partial^2 u(x_i)}{\partial x^2}$.

Аналогичную оценку нетрудно провести и для второй производной. Для этого необходимо воспользоваться (6.82) и аналогичным разложением, записанным для $u(x_i - \Delta x)$:

$$u(x_i - \Delta x) = u(x_i) - \Delta x\frac{\partial u(x_i)}{\partial x} + \frac{\Delta^2 x}{2!}\frac{\partial^2 u(x_i)}{\partial x^2} - \frac{\Delta^3 x}{3!}\frac{\partial^3 u(x_i)}{\partial x^3} + \frac{\Delta^4 x}{4!}\frac{\partial^4 u(x_i)}{\partial x^4} - \ldots \qquad (6.84)$$

Сложив (6.82) и (6.84) получим выражение для второй производной:

$$\frac{\partial^2 u(x_i)}{\partial x^2} = \frac{u(x_i+\Delta x)-2u(x_i)+u(x_i-\Delta x)}{\Delta x^2} - \frac{\Delta^2 x}{12}\frac{\partial^4 u(x_i)}{\partial x^4} - \cdots. \quad (6.85)$$

Из сравнения (6.85) и (6.72) видно, что погрешность (8) определяется не учтенными в ней слагаемыми высоких порядков, начиная с $\frac{\Delta^2 x}{12}\frac{\partial^4 u(x_i)}{\partial x^4}$. Поэтому ошибка (6.72) уменьшается пропорционально квадрату Δx. Данный результат полезно учитывать при выборе шага сетки. Так, например, уменьшение вдвое шага $\Delta x = \Delta y = h$ приводит к снижению ошибки аппроксимации для уравнения эллиптического типа в четыре раза.

Нельзя утверждать, что уменьшение шага сетки однозначно повышает точность решения методом конечных разностей. С увеличением количества узлов сетки возрастает объем вычислений и, следовательно, растут вычислительные погрешности. На практике для оценки погрешности решения можно провести ряд пробных расчетов с разными значениями шага сетки и выбрать вариант, обеспечивающий приемлемую точность при невысоких вычислительных затратах.

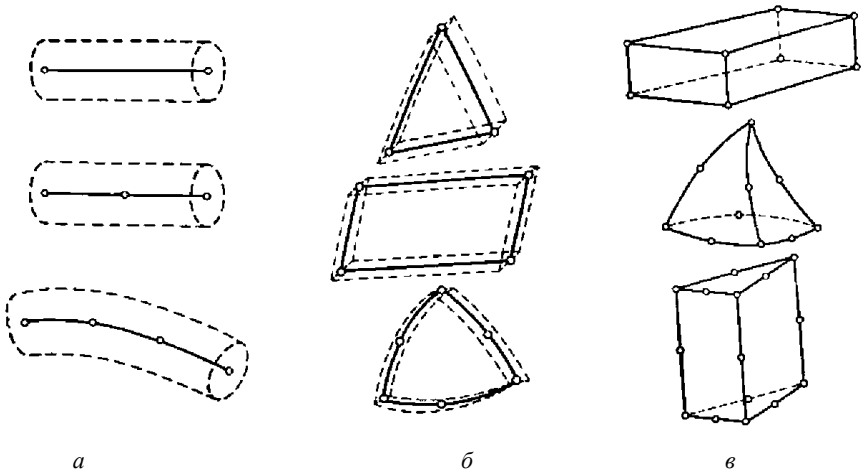
6.4.4. Основы метода конечных элементов

Существуют различные формулировки метода конечных элементов, различающиеся как в основных, так и в менее значительных деталях. Ограничимся кратким рассмотрением основных этапов решения задачи этим методом.

6.4.4.1. Формирование сетки

Метод конечных элементов основывается на том, что любое непрерывное распределение физической переменной $u(x,y,z,t)$ в расчетной области, например деформации или температурного поля, можно аппроксимировать набором кусочно-непрерывных функций, определенных на конечном числе подобластей *(конечных элементов)*. Данные элементы имеют общие узловые точки и в совокупности аппроксимируют форму области.

В зависимости от геометрии и размерности задачи используют различные виды конечных элементов (см. рис. 6.7). Чаще всего применяются простейшие элементы – *симплексы*.



а *б* *в*

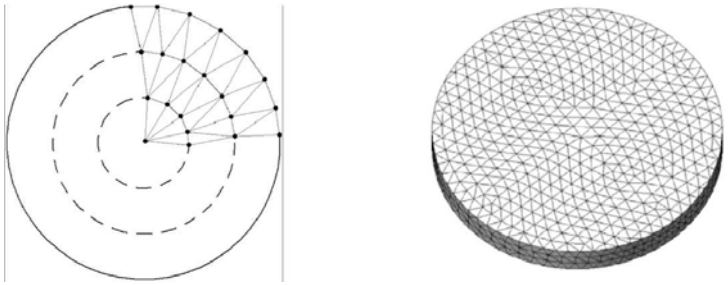
Рис. 6.7. Некоторые виды конечных элементов:
а – омерные; *б* – двумерные; *в* – трехмерные

Количество узлов в симплексе на единицу превышает размерность задачи. Для двумерной задачи симплекс-элементом будет являться прямолинейный трехузловой треугольник, а для трехмерной – прямолинейный четы-рехузловой тетраэдр. Широкое применение симплексов обусловлено тем, что они позволяют заполнять расчетную область произвольной формы полностью без разрывов, а также на них удобно использовать в качестве аппроксимирующих функций линейные полиномы.

Обычно для разбиения расчетной области на элементы используется специальный *алгоритм покрытия,* обеспечивающий автоматическую генерацию сетки.

Одна из таких процедур работает следующим образом (см. рис. 6.8, *а*). Вначале производится нанесение с некоторым шагом узлов на границу области. После этого внутри области строится вспомогательная кривая эквидистантная границе. На кривую также наносятся узлы. Поочередное соединение узлов на первом и втором контурах дает симплексы. Далее все операции повторяются до заполнения симплексами всей области.

Известны и другие алгоритмы формирования конечных элементов: «картографический», использующий наложение на расчетную область сетки, которая затем адаптируется к границам и неоднородностям геометрии, или методы, основанные на заполнении объекта набором фигур (тел) с использованием свойств симметрии или отражения.

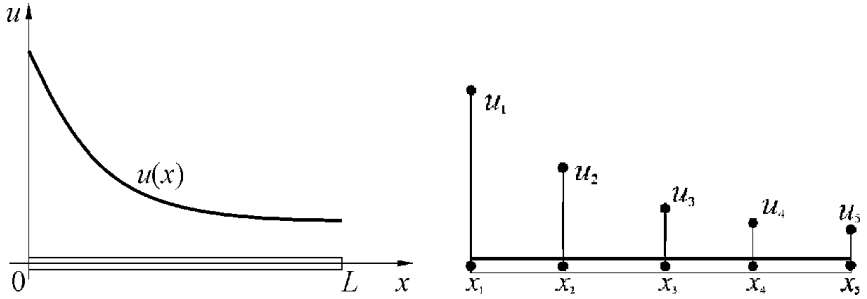


а *б*

Пример автоматически сгенерированной трехмерной сетки для круглого диска показан на рис. 6.8, *б*.

6.4.4.2. Конечно-элементная аппроксимация

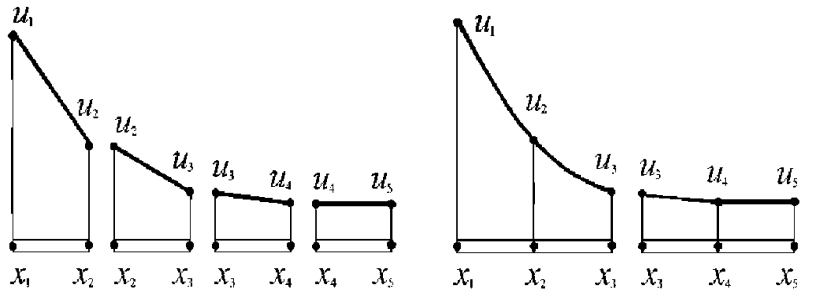
Рассмотрим построение аппроксимации на одномерном примере. Пусть требуется найти распределение некоторой непрерывной функции $u(x)$ вдоль стержня (см. рис. 6.9, *а*). На практике эта функция может описывать, например, распределение температуры или деформацию стержня.



а *б*

Рис. 6.9. Одномерное распределение

Выберем и пронумеруем ряд точек вдоль оси x – это узловые точки (рис. 6.9, *б*). В нашем примере взято всего пять точек. Вообще говоря, их может быть произвольное количество, и располагаться они могут не на равном расстоянии друг от друга. Предположим, что значения в узловых точках известны. Они обозначены на рис. 6.9, *б* в соответствии с номерами узлов – u_1 u_2, u_3, u_4.



а *б*

Рис. 6.10. Варианты разбиения стержня на элементы

Разбиение расчетной области, то есть стержня, на конечные элементы может быть проведено различными способами. Можно, например, выделить четыре элемента, включив в каждый из них по два соседних узла (рис. 6.10, *а)*. А можно выделить в стержне всего два элемента, содержащие по три узла (рис. 6.10, б).

При использовании четырех элементов, каждый из которых включает только два узла, аппроксимирующая функция в пределах элемента будет линейна по x, так как две точки однозначно определяют прямую линию. Общая аппроксимация зависимости $u(x)$ по всей длине стержня будет складываться из четырех отрезков прямых (рис. 6.10, *а*).

Зависимость $u(x)$ в пределах одного элемента, ограниченного двумя соседними узлами x_i и X_j ($j = i + 1$), можно представить линейным интерполяционным полиномом $u(x) \sim a + a_x x$. Определив параметры a и a_x по известным в точках x_i и x_j значениям функции u_i и U_j, запишем интерполяционный полином, то есть *функцию элемента* следующим образом:

$$u(x) \approx \frac{x_j - x}{x_j - x_i} u_i + \frac{x - x_i}{x_j - x_i} u_j = N_i u_i + N_j u_j = \left[N^{(e)}\right]\left[u^{(e)}\right], \quad (6.86)$$

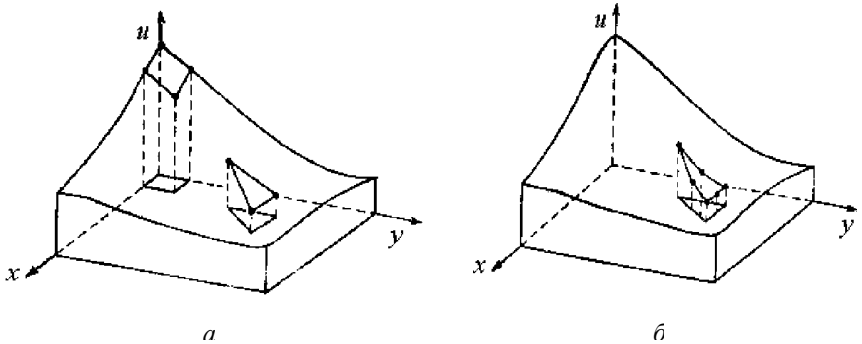
где N_i и N_j – *функции формы* конечного элемента, u_i и u_j – значения $u(x)$ в точках x_i и x_j, $\left[N^{(e)}\right] = \left[N_i, N_j\right]$ – матричная строка функций формы элемента:

$$\left[u^{(e)}\right] = \begin{bmatrix} u_i \\ u_j \end{bmatrix}.$$

Следует отметить, что ряд терминов метода конечных элементов получили название из механики, где они впервые начал активно использоваться.

В случае разбиения области на два элемента (рис. 6.10, *б)* три узловые точки в каждом из них позволяют однозначно определить функции элементов в виде полиномов второй степени. Соответственно, распределение *u(x)* на всей длине стержня будет аппроксимироваться кусочно-непрерывной квадратичной функцией. При этом общая аппроксимация для стержня может содержать излом из-за несовпадения углов наклона графиков полиномов (их первых производных) в третьем узле.

Для двумерной или трехмерной задачи аппроксимация строится аналогичным образом. В зависимости от вида элементов (количества используемых в них узлов) также применяется линейная или нелинейная аппроксимация. Примеры аппроксимации двухмерной непрерывной функции $u(x,y)$ приведены на рис. 6.11.



а *б*

Рис. 6.11. Моделирование двухмерной скалярной функции с помощью линейной (*а*) и нелинейной (*б*) аппроксимации

Функция формы элемента будет представлена плоскостью, если для него взято минимальное число узлов, которое для треугольного элемента равно трем, а для четырехугольного – четырем. В этом случае используют линейную аппроксимацию:

$$u(x, y) \approx \alpha + \alpha_x x + \alpha_y y .$$

По аналогии с одномерным случаем линейный интерполяционный многочлен для простейшего треугольного элемента, включающего только три узла, записывают в виде

$$u(x, y) \approx N_i u_i + N_j u_j + N_k u_k = \left[N^{(e)}\right]\left[u^{(e)}\right], \quad (6.87)$$

где N_i, N_j, N_k – функции формы элемента;
u_i, u_j, u_k – значения функции в узлах, принадлежащих элементу;
$[N^{(e)}]$ – матричная строка функций формы элемента;
$[u^{(e)}]$ – вектор-столбец значений функции $u(x,y)$ в его узлах.

Если элемент содержит большее количество узлов, то аппроксимирующая функция элемента будет отображаться криволинейной поверхностью.

Для всей расчетной области аппроксимацией распределения $u(x,y)$ является кусочно-линейная (или кусочно-нелинейная) поверхность, каждый из участков которой определяется на отдельном элементе с помощью значений $u(x,y)$ в принадлежащих ему узлах.

Для построения аппроксимации так, как это было показано выше, необходимо знать распределение $u(x,y)$ во всей расчетной области. Однако до решения задачи эта зависимость обычно как раз и не известна. Тем не менее, используя аппроксимирующие формулы (6.86) или (6.87), решение можно получить. Способы поиска решения рассмотрены ниже.

6.4.4.3. Построение решения

Вначале необходимо провести *объединение конечных элементов в ансамбль.* Значения u_1, u_2, u_3, ... в узлах теперь будем рассматривать как неизвестные переменные, которые необходимо найти. Сформируем из этих значений, взятых по всей расчетной

области, столбцовую матрицу, которую обозначим $\left\lfloor u^{(\Sigma)} \right\rfloor$. Каждой строке $\left\lfloor u^{(\Sigma)} \right\rfloor$соответствует узел сетки конечных элементов. Тогда аппроксимацией для всей расчетной области (в двухмерном случае) будет:

$$u(x,y) \approx \left\lfloor N^{(\Sigma)} \right\rfloor \left\lfloor u^{(\Sigma)} \right\rfloor,$$

где $\left\lfloor N^{(\Sigma)} \right\rfloor$ – матричная строка функций формы всех конечных элементов, входящих в расчетную область.

При составлении матриц $\left\lfloor N^{(\Sigma)} \right\rfloor$ и $\left\lfloor u^{(\Sigma)} \right\rfloor$ производится сквозная нумерация узлов. Для двух- и трехмерных задач эта процедура сложна и от нее в значительной степени зависит время расчета.

Следующий этап – *построение разрешающей системы алгебраических уравнений* на основе конечно-элементной аппроксимации. В результате решения задачи узловые значения u_1, u_2, u_3, ... должны быть «подобраны» так, чтобы они обеспечивали наилучшее приближение к истинному распределению $u(x,y)$. Этот «подбор» может осуществляться различными способами.

Существуют *вариационная* и *проекционная* формулировки метода конечных элементов. При вариационном подходе производится минимизация некоторого функционала, связанного с исходным дифференциальным уравнением. Например, в задачах механики может минимизироваться потенциальная энергия системы. Процесс минимизации приводит к решению системы алгебраических уравнений относительно узловых значений $u(x)$.

Проекционный вариант метода конечных элементов является частным случаем метода взвешенных невязок. Последний основан на минимизации невязки в дифференциальном уравнении при подстановке в него приближенного решения вместо точного. В методе конечных элементов оценка невязки производится по отдельным элементам и также сводится к решению системы алгебраических уравнений относительно узловых значений $u(x)$.

При построении решения функции формы N позволяют определять в пределах каждого элемента пространственные дифференциальные операторы первого порядка от скалярного или векторного поля.

В методе конечных элементов также как и в методе конечных разностей матрица коэффициентов системы уравнений включает большое число нулевых элементов, что облегчает решение задачи.

К достоинствам метода конечных элементов, благодаря которым он находит широкое применение, относятся гибкость и разнообразие сеток, четко формализованные алгоритмы построения дискретных задач для произвольных областей, простота учета естественных краевых условий. Кроме того, этот метод применим к широкому классу исходных задач, а оценки погрешностей приближенных решений, как правило, получаются при менее жестких ограничениях, чем в методе конечных разностей.

Несмотря на то, что метод конечных разностей на первый взгляд представляется наиболее легким в реализации и был разработан раньше метода конечных элементов, последний в настоящее время является доминирующим в современных расчетных программах.

6.5. Элементы математической статистики

Результаты наблюдений во многих случаях можно представить последовательностью действительных чисел $(x_1, x_2, ..., x_n) = x$.

Для того, чтобы из ряда наблюдений можно было извлечь полезную информацию, необходимо иметь модель явления. Вероятностные модели оказываются наиболее пригодными при анализе явлений, исходы $x \in X$ которых обладают некоторой степенью неопределенности. Понятие неопределенности в теории вероятностей формализуется путем введения распределения вероятностей на множестве X. В простейшем случае, когда X конечное или счетное множество, задаются вероятности $p(x)$ всех его элементов x, так что $0 \le p(x) \le 1$ и $\sum_{x \in X} p(x) = 1$.

Любое множество $A \subseteq X$ называют в этом случае ***событием***, а его ***вероятность*** определяют формулой $p(A) = \sum_{x \in A} p(X)$.

Взаимоотношение явления и его вероятностной модели имеет статистический характер, то есть обнаруживается при повторных наблюдениях за явлением. Частоты исходов в длинном ряду испытаний

стабилизируются, их колебания с ростом числа испытаний уменьшаются. Это эмпирический факт, называемый законом устойчивости частот, наблюдается в самых различных ситуациях. Уже выходя за пределы реального опыта, полагают, что при неограниченном повторении частоты стремятся к пределам, которые и принимают за вероятности соответствующих исходов или событий.

6.5.1. Генеральная совокупность. Выборка. Статистические ряды

Пусть задача состоит в том, чтобы исследовать заданный качественный или количественный признак, характеризующий элементы некоторой последовательности (совокупности) наблюдений. Все множество объектов, входящих в рассматриваемую совокупность называют *генеральной совокупностью*. Число элементов генеральной совокупности может быть достаточно большим (в теоретических рассмотрениях используется и совокупности, содержащие бесконечное множество элементов).

Часть генеральной совокупности, выбранную из нее некоторым (случайным) образом, называют *выборочной совокупностью* (выборкой). Объем выборки, обозначаемый n (по количеству элементов), может быть и сравнительно большим и малым, но не может содержать меньше двух единиц. Выборочный метод является основным при изучении статистических совокупностей. Его преимущество перед сплошным учетом всех членов генеральной совокупности заключается в том, что он сокращает время и затраты труда (за счет уменьшения числа наблюдений), а главное позволяет получать информацию о таких совокупностях, сплошное исследование которых практически невозможно или нецелесообразно.

Элементы совокупности называют *вариантами*. Существует два основных способа отбора вариант из генеральной совокупности: повторный и бесповторный. В практике обычно применяют бесповторный отбор.

Статистическими рядами называют ряды числовых значений некоторого признака, расположенного в определенном порядке.

Расположим варианты $x_1, x_2, ..., x_n$ (n – объем выборки) в порядке возрастания и перенумеруем их заново: $x_1 \le x_2 \le ... \le x_n$, где:

$$x_1 = \min(x_1, x_2, ..., x_n),\ ...,\ x_n = \max(x_1, x_2, ..., x_n). \qquad (6.88)$$

Такую перенумерованную последовательность часто называют *вариационным рядом*.

Числа, показывающие сколько раз отдельные варианты встречаются в данной совокупности, называются *частотами* или весами вариант и обозначаются p или f.

Общая сумма частот всегда равна объему выборки. Говорят еще об относительных частотах (выражаются в частях или % от n).

Интервальный вариационный ряд, такой статистический ряд, в котором частоты распределяются по отдельным интервалам или промежуткам (от–до), на которые разбивается вариация признака в пределах от минимальной до максимальной варианты совокупности. Эти промежутки или классовые интервалы могут быть равными и неравными по ширине. Чаще всего рассматриваются равные интервалы. Величина равных интервалов определяется делением размаха варьирования признака ($x_{\max} - x_{\min}$) на число групп или классов (K), намечаемых при построении вариационного ряда:

$$i = \frac{x_{\max} - x_{\min}}{K}, \qquad (6.89)$$

где i – величина классового интервала, а величина K определяется по формуле Стерджеса $K = 1 + 3{,}32 \lg n$.

В общем случае техника построения вариационного ряда сводится к следующему:

1. Составляем сводку исходных данных.
2. Отыскиваем $x_{\min}$ и $x_{\max}$ варианты.
3. Определяем величину классового интервала i по формуле данной выше. Если значения признака выражены целыми числами и классовый интервал i окажется равным 1 (или ≈1), выборка распределяется в безинтервальный вариационный ряд. Если $i \neq 1$, выборку следует распределять в интервальный вариационный ряд. Кроме того, следует соблюдать правило, согласно которому величина классового интервала должна соответствовать точности, при-

нятой при измерении учитываемого признака (если ε = 0,001, то классовый интервал тоже определяется с точностью 0,001).

4. При построении интервального вариационного ряда следует добиваться того, чтобы минимальная варианта $x_{\min}$ попадала в середину классового интервала, то есть, $l_1 = x_{\min} - \frac{i}{2}$, где l_1 – нижняя граница первого классового интервала. Определяем верхнюю границу первого классового интервала $m_1 = l_1 + i$, второго классового интервала $m_2 = m_1 + i$, K – того классового интервала $m_K = m_{K-1} + i$.

5. Наметив классовые интервалы, остается распределить по ним варианты выборки, то есть, определить частоты каждого класса.

6.5.2. Графическое изображение вариационных рядов. Эмпирическое распределение

При построении графика зависимости частот от значений вариант безинтервального вариационного ряда по оси абцисс откладываются значения классов (вариант) по оси ординат – частоты. В результате будет построена геометрическая фигура в виде многоугольника. Полученный график называют полигоном распределения частот.

При построении графика зависимости частот от значений вариант интервального вариацинного ряда по оси абцисс откладывают границы классовых интервалов, по оси ординат – частоты. В результате – гистограмма распределения частот. Можно поступить иначе: по оси абцисс отложить срединные значения классов $\left(\frac{l_k + m_k}{2}\right)$, по оси ординат частоты для указанного класса.

Вариационная кривая – это сглаженные значения полигона.

Совокупность значений x_i и соответствующих им частот называется эмпирическим распределением.

6.5.3. Средние величины и показатели вариации

Вариационные ряды и их графики дают наглядное представление о том, как варьирует тот или иной количественный

признак. Но они недостаточны для полной характеристики статистической совокупности. Количественные показатели, которые (логически и теоретически обоснованы) позволяют судить о качественном своеобразии варьирующих объектов и сравнивать их между собой, называются статистическими характеристиками.

В отличие от индивидуальных числовых характеристик средние величины обладают большей устойчивостью, способностью характеризовать группу однородных вариант одним (средним) значением. И хотя средние абстрагируют нас от конкретных вещей, они вполне понятны и ощутимы. Средний рост, средняя масса, и т. д. (то есть, здесь уравновешиваются все индивидуальные отклонения и появляется качественное своеобразие группового объекта).

По определению Гаусса, истинной средней служит такая величина, сумма квадратов отклонений от которой обладает нименьшим значением.

$$M = \sqrt[k]{\frac{\sum_i^n x_i^k}{n}}, \qquad (6.90)$$

где M – средняя величина;
x_i - варианта, n – объем выборки;
k – величина, определяющая вид средней.

Средние величины могут характеризовать только однородную массу вариант (если это не так, следует сгруппировать варианты в отдельные качественно однородные группы и вычислять групповые средние).

$k = -1$ – средняя гармоническая. В этом случае:

$$\overline{x}_h = \frac{n}{\sum_{i=1}^n x_i}.$$

В некоторых случаях для усреднения количественных признаков используется такой тип средней.

$k = 2$ – средняя квадратическая. При выражении количественных признаков вариант мерами площади более точной усредненной характеристикой будет средняя квадратическая:

$$\bar{x}_q = \sqrt{\frac{\sum_{i=1}^{n} x_i}{n}} .$$

$k = 3$ – средняя кубическая. Более точная средняя характеристика, в тех случаях, когда варьирующий признак выражен в объмных единицах:

$$\bar{x}_k = \sqrt[3]{\frac{\sum_{i=1}^{n} x_i}{n}} .$$

Средняя геометрическая является более точной характеристикой при определении средних прибавок или при увеличении линейных размеров тел, прироста численности популяции за определенный промежуток времени.

$$\bar{x}_g = \sqrt[n]{x_1 x_2 \ldots x_n} .$$

$k = 1$ – средняя арифметическая. Эту величину рассотрим подробнее.

6.5.4. Средняя арифметическая и ее свойства

$\bar{x}$ – средняя арифметическая является центром распределения, вокруг которого группируются все варианты статистической совокупности.

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i . \qquad (6.91)$$

Если рассматривается интервальный вариационный ряд, то средняя арифметическая, называемая взвешенной, вычисляется по формуле:

$$\bar{x} = \frac{\sum_{i=1}^{K} x_i f_i}{\sum_{i=1}^{K} f_i} , \qquad (6.92)$$

где f_i – частота i –го класса;

$\sum_{i=1}^{K} f_i = n$, K – количество классовых интервалов.

Рассмотрим свойства средней арифметической.

Свойство 1. Если каждую варианту совокупности уменьшить или увеличить на какое-то произвольное положительное число А, то и средняя арифметическая уменьшится или увеличится на столько же.

Свойство 2. Если каждую варианту разделить или умножить на одно и тоже число А, то и средняя арифметическая изменится во столько же раз.

Свойство 3. Сумма произведений отклонений вариант от их средней арифметической на соответствующие им частоты равна нулю.

Свойство 4. Сумма квадратов отклонений вариант от их средней арифметической меньше суммы квадратов отклонений тех же вариант от любой другой величины А, не равной средней арифметической.

Размах вариации $x_{\max} - x_{\min}$ характеризует варьирование признака в совокупности.

Рассмотрим еще две характеристики выборочной совокупности: дисперсию и среднее квадратическое отклонение. Эти величины характеризуют не только величину, но и специфику варьирования признака.

6.5.5. Дисперсия и ее свойства. Среднее квадратическое отклонение

Дисперсия (или варианса) – это средний квадрат отклонений вариант данной совокупности от их средней величины.

Дисперсия:

$$S_x^2 = \frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}\ , \qquad (6.93)$$

или, если используется интервальный вариационный ряд:

$$S_x^2 = \frac{\sum_{i=1}^{K}(x_i - \overline{x})^2 f_i}{n-1}. \qquad (6.94)$$

Свойства дисперсии:

Свойство 1. Если каждую варианту совокупности уменьшить или увеличить на одно и тоже постоянное число A, то дисперсия не изменится.

Свойство 2. Если каждую варианту разделить (или умножить) на одно и тоже постоянное число A, то дисперсия уменьшится (или увеличится в A^2 раз.

Среднее квадратическое отклонение определяется по следующей формуле:

$$S_x = \sqrt{S_x^2}\ . \qquad (6.95)$$

Чем сильнее варьирует признак, тем больше величина этого показателя и наоборот.

6.5.6. Коэффициент вариации

Дисперсия и среднее квадратическое отклонение – величины абсолютные и имеют размерность вариант совокупности. Если же хотим сравнивать изменчивость признаков, выраженных разными единицами, следует перейти к относительным показателям.

Один из этих показателей – показатель Пирсона (коэффициент вариации):

$$V = \frac{S_x}{\overline{x}} 100\%. \qquad (6.96)$$

Чем выше процент, тем более изменчив признак.

6.5.7. Структурные средние

Медиана (Me) эмпирического распределения – средняя, относительно которой ряд распределения делится на две половины: в обе стороны от медианы располагается определенное число вариант. Если число вариант нечетно – центральная варианта его медиана. При четном – определяется по полусумме соседних вариант, расположенных в центре ряда.

Мода (Mo) – величина, которая встречается в данной совокупности наиболее часто. Класс с наибольшей частотой называется модальным.

О чем можно судить по медиане выборки? Важна эта характеристика особенно в тех случаях, когда обнаруживается значительная или резкая асимметрия в распределении частот по классам вариационного ряда. Часто используется для установления границ тех или иных нормативов.

6.5.8. Законы распределения случайных величин

Между отдельными значениями варьирующих признаков и частотой их встречаемости в генеральной совокупности существует определенная связь (это наглядно можно увидеть на графике зависимости частот от значения вариат).

Реализация того или иного эначения варьирующего признака представляет собой случайное событие. Предсказать появление случайного события в отдельных испытаниях (наблюдениях) можно лишь с некоторой уверенностью, или вероятностью, которое имеет данное событие. Случайной называется переменная величина, способная в одних и тех же условиях испытания принимать различные числовые значения. Функция $P(x)$, связывающая значения вариант x_i с вероятностями p_i называется законом распределения случайной величины.

В природе широко распространена закономерность: в массе относительно однородных членов, составляющих статистическую совокупность, большинство их оказывается среднего или близкого к нему размера, и чем дальше они отстоят от среднего уровня варьирующего признака , тем реже встречаются в данной совокупности. Такое поведение может описано **законом нормального распределения** (формула Гаусса – Лапласа)

$$P(x_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_i-\mu}{\sigma}\right)^2}, \qquad (6.97)$$

где σ^2 – дисперсия генеральной совокупности;

μ – генеральная средняя арифметическая (математическое ожидание);

Величина $t = \frac{x_i - \mu}{\sigma}$ получила название нормированного отклонения.

Выборочные характеристики рассматриваются как приближенные значения или точечные оценки соответствующих генеральных параметров, которые, как правило, остаются неизвестными. Средняя арифметическая выборки $\bar{x}$ служит оценкой средней арифметической генеральной совокупности μ, выборочная дисперсия S_x^2 является оценкой генеральной дисперсии σ^2, S_X – в качестве точечной оценки стандартного отклонения σ генеральной совокупности.

Формально математическое ожидание соответствует средней арифметической эмпирических распределений. Однако отождествлять эти величины нельзя. Средняя арифметическая выражается отношением суммы всех членов ряда к их общему числу, а математическое ожидание представляет сумму произведений членов ряда на их вероятности. Эмпирическая средняя стремится к своей вероятной величине, то есть, к математическому ожиданию по мере увеличения числа испытаний: чем больше число испытаний, тем ближе эмпирическая средняя к математическому ожиданию.

6.5.9. Статистические гипотезы

Величина отклонения выборочного показателя от его генерального параметра называется статистической ошибкой этого показателя или ошибкой репрезентативности. Статистические ошибки – это не ошибки возникающие в результате измерений. Их появление обусловлено процессом отбора вариант из генеральной совокупности и к ошибкам измерений отношения не имеют. Чем сильнее варьирует признак, тем больше при прочих равных условиях будет ошибка выборочных показателей и наоборот.

По известным значениям выборочных характеристик можно установить интервал, в котором с той или иной вероятностью находится величина генерального параметра. Вероятности, признанные достаточными для уверенных суждений о генеральных параметрах на основании выборочных показателей, называются доверительными.

Решение той или иной задачи, как правило не обходится без сравнений. О преимуществе одной из сравниваемых групп судят обычно по разности между выборочными средними. Но эта оценка тоже может носить случайный характер. Чтобы решить вопрос об истинной значимости различий,наблюдаемых между выборочными средними исходят из статистических гипотез – предположений или допущений о неизвестных генеральных параметрах, выражаемых в терминах вероятности, которые могут быть проверены на основании выборочных показателей.

Применяется так называемая нулевая гипотеза (H_0), то есть, предположение о том, что между генеральными параметрами сравниваемых групп разница равна нулю и различия, наблюдаемые между выборочными показателями, носят исключительно случайный характер.

Противоположная или альтернативная гипотеза(H_1), наоборот, исходит из предположения, что между генеральными параметрами сравниваемых групп разница не равна нулю.Статистические гипотезы могут исходить и из других предположений.

Истинность принятой гипотезы проверяется с помощью критериев значимости, или достоверности, то есть, специально выработанных случайных величин, функции распределения которых известны. Обычно для каждого критерия составляется таблица, в которой содержатся критические точки, отвечающие определенным числам степеней свободы (k) и принятым уровням значимости α.

Уровни значимости – значение вероятности, при котором различия, наблюдаемые между выборочными показателями, можно считать несущественными, случайными. В исследовательской работе обычно принимается 5 % уровень значимости, который соответствует вероятности $P = 0.05$ и нормированное отклонение $t = 1.96$, если распределение критерия нормально. Если окажется, что $P \geq 0.05$, то нулевая гипотеза сохраняется, иначе отвергается.

Рассмотрим гипотезу о равенстве средних арифметических исходных генеральных совокупностей. В рассмотрении участвуют две выборки и их параметры: объем выборки и средняя арифметическая (m и $\bar{x}$ для первой выборки и n и $\bar{y}$ для второй). Нулевая гипотеза предполагает, что $\bar{x} = \bar{y}$.

Имеется ли различие между этими средними значениями? Чтобы определить какой характер носит это различие используют критерий Стьюдента. Вычисленное значение критерия будет определено по формуле:

$$t = \frac{\bar{x} - \bar{y}}{S}\sqrt{\frac{mn}{m+n}} \,. \tag{6.98}$$

$$S^2 = \frac{\sum_{i=1}^{m}(x_i - \bar{x})^2 + \sum_{i=1}^{n}(y_i - \bar{y})^2}{m+n-2}. \tag{6.99}$$

Вычисленное значение критерия сравниваем с критической точкой, взятой из таблицы распределения Стьюдента в соответствии с выбранным уровнем значимости и числом степеней свободы $m+n-2$. Если $|t|$ больше табличного значения, то гипотезу о равенстве средних следует отвергнуть. Это будет означать, что различие средних нельзя считать случайным.

Теперь рассмотрим гипотезу о равенстве дисперсий исходных генеральных совокупностей. В рассмотрении участвуют две выборки и их параметры: объем выборки и дисперсия (m и S_x^2 для первой выборки и n и S_y^2 для второй). Нулевая гипотеза предполагает, что $S_x^2 = S_y^2$. Воспользуемся критерием Фишера $F = \frac{S_x^2}{S_y^2}$ (отношение большей из дисперсий к меньшей). Вычисленное значение критерия Фишера сравниваем с критическим значением, взятым из таблицы распределения Фишера в соответствии с уровнем значимости α и степенями свободы m и n. Если вычисленное значение критерия больше табличного, то различие выборочных дисперсий следует признать значимым.

Чтобы проверить, распределен ли варьирующий признак по нормальному закону, поступают следующим образом. Пусть элементы выборки распределены по K – интервалам, причем j-му интервалу ($j = 1, 2, \ldots, K$) соответствует частота f_j. Для проверки гипотезы о каком-либо распределении случайной величины используют критерий χ^2 (критерий Пирсона).

Вычисленное значение критерия определяется по формуле:

$$\chi^2 = n\sum_{j=1}^{K}\frac{(f_j - F_j)^2}{F_j}, \tag{6.100}$$

где f_j – относительная частота соответствующая j - ому интервалу;

F_j – теоретическая частота, соответствующая j - ому интервалу.

Правило вычисления F_j и определение числа степеней свободы зависит от вида теоретического распределния и способа оценки его параметров.

Сравним эмпирическое распределение с нормальным.

$$F_j = \int_{l_j}^{m_j} P(x)dx \,, \tag{6.101}$$

где l_j и m_j – левая и правая границы j - ого интервала;

$P(x)$ – плотность нормального распределения.

Для упрощения вычислений можно заменить интеграл в правой части этого равенства произведением длины промежутка интегрирования и значения функции в средней точке интервала, то есть,

$$F_j = \frac{x_{\max} - x_{\min}}{K}\frac{1}{S_x\sqrt{2\pi}}e^{-\frac{(x_j - \bar{x})^2}{2S_x^2}} \,. \tag{6.102}$$

В таблице распределения χ^2 находим критическую точку, соответствующую выбранному уровню значимости α и числу степеней свободы $K-3$ (если μ и σ не определяются по имеющимся данным, а известны заранее, то число степеней свободы $K-1$).

Если вычисленное по формуле значение критерия больше табличного, то на уровне значимости α прверяемая гипотеза должна быть отвергнута.

Можно поступить еще и так. Пусть f_j – абсолюное значение частоты j-ого интервала. Можно сравнить частоты теоретические и эмпирические. В этом случае

$$F_j = n\frac{x_{\max} - x_{\min}}{K}\frac{1}{S_x\sqrt{2\pi}}e^{-\frac{(x_j-\bar{x})^2}{2S_x^2}}, \qquad (6.103)$$

где n – объем выборки.

Для нормального распределения характерно совпадение по абсолютной величине средней арифметической, медианы и моды. Для этого вида распределения характерно то, что на равные интервалы, измеряемые нормированным отклонением от центра распределения, приходится равное число вариант.

Кривую нормального распределения характеризуют величины асимметрия (As) и эксцесс (Ex). Эти величины для рассматриваемой выборки можно определить, зная выборочные характеристики: среднюю арифметическую и дисперсию.

$$As = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^3 f_i}{nS_x^3}, \quad Ex = \frac{\Sigma(x_i - \bar{x})^4 f_i}{nS_x^4} - 3. \qquad (6.104)$$

Можно оценить статистические ошибки выборочных характеристик. Для выборочной средней $S_{\bar{x}} = \sqrt{\frac{S_x^2}{n}}$, для асимметрии $S_{As} = \frac{6}{n+3}$, для эксцесса $S_{Ex} = 2\sqrt{\frac{6}{n+5}}$. И нулевая гипотеза о том, что эмпирическое распределение нормально будет отвергаться, если $t_{As} = \frac{As}{S_{As}} > 3$ и $t_{Ex} = \frac{Ex}{S_{Ex}} > 3$.

Контрольные вопросы

1. Сформулируйте определение математической модели.
2. Перечислите основные численные методы решения технических задач.
3. Приведите примеры моделей, приводящих к необходимости численного дифференцирования и интегрирования функций.
4. Приведите примеры моделей, описываемых обыкновенными дифференциальными уравнениями.
5. В чем состоит идея методов Рунге – Кутта и прогноза и коррекции?
6. Приведите примеры моделей, описываемых дифференциальными уравнениями в частных производных.

7. МЕТОДЫ ОПТИМИЗАЦИИ И СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ

7.1. Характеристика методов решения задач оптимизации

При решении конкретной задачи оптимизации исследователь прежде всего должен выбрать математический метод, который приводил бы к конечным результатам с наименьшими затратами на вычисления или же давал возможность получить наибольший объем информации об искомом решении. Выбор того или иного метода в значительной степени определяется постановкой оптимальной задачи, а также используемой математической моделью объекта оптимизации.

В настоящее время для решения оптимальных задач применяют в основном следующие методы:

- методы исследования функций классического анализа;
- методы, основанные на использовании неопределенных множителей Лагранжа;
- вариационное исчисление;
- динамическое программирование;
- принцип максимума;
- линейное программирование;
- нелинейное программирование.

В последнее время разработан и успешно применяется для решения определенного класса задач метод *геометрического программирования*.

Как правило, нельзя рекомендовать какой-либо один метод, который можно использовать для решения всех без исключения задач, возникающих на практике. Одни методы в этом отношении являются более общими, другие – менее общими. Наконец, целую группу методов (методы исследования функций классического анализа, метод множителей Лагранжа, методы нелинейного программирования) на определенных этапах решения оптимальной задачи можно применять в сочетании с другими методами, например динамическим программированием или принципом максимума.

Отметим также, что некоторые методы специально разработаны или наилучшим образом подходят для решения оптимальных задач с математическими моделями определенного вида. Так, математический аппарат линейного программирования, специально создан для решения задач с линейными критериями оптимальности и линейными ограничениями на переменные и позволяет решать большинство задач, сформулированных в такой постановке. Так же и геометрическое программирование предназначено для решения оптимальных задач, в которых критерий оптимальности и ограничения представляются специального вида функциями – позиномами.

Динамическое программирование хорошо приспособлено для решения задач оптимизации многостадийных процессов, особенно тех, в которых состояние каждой стадии характеризуется относительно небольшим числом переменных состояния. Однако при наличии значительного числа этих переменных, т. е. при высокой размерности каждой стадии, применение метода динамического программирования затруднительно вследствие ограниченного быстродействия и объема памяти.

Наилучшим путем при выборе метода оптимизации, наиболее пригодного для решения соответствующей задачи, следует признать исследование возможностей и опыта применения различных методов оптимизации. Ниже приводится краткий обзор математических методов решения оптимальных задач и примеры их использования. Здесь же дана лишь краткая характеристика указанных методов и областей их применения, что до некоторой степени может облегчить выбор того или иного метода для решения конкретной оптимальной задачи.

Методы исследования функций классического анализа представляют собой наиболее известные методы решения несложных оптимальных задач, которые известны из курса математического анализа. Обычной областью использования данных методов являются задачи с известным аналитическим выражением критерия оптимальности, что позволяет найти не очень сложное, также аналитическое выражение для производных. Полученные приравниванием к нулю производных уравнения, определяющие экстремальные решения оптимальной задачи, крайне редко удается решить аналитическим путем, поэтому, как правило, применяют вычислительные машины. При этом надо решить систему конечных уравнений, чаще всего нелинейных, для чего приходится использовать численные методы, аналогичные методам нелинейного программирования.

Дополнительные трудности при решении оптимальной задачи методами исследования функций классического анализа возникают вследствие того, что система уравнений, получаемая в результате их применения, обеспечивает лишь необходимые условия оптимальности. Поэтому все решения данной системы (а их может быть и несколько) должны быть проверены на достаточность. В результате такой проверки сначала отбрасывают решения, которые не определяют экстремальные значения критерия оптимальности, а затем среди остающихся экстремальных решений выбирают решение, удовлетворяющее условиям оптимальной задачи, т. е. наибольшему или наименьшему значению критерия оптимальности в зависимости от постановки задачи.

Методы исследования при наличии ограничений на область изменения независимых переменных можно использовать только для поиска экстремальных значений внутри указанной области. В особенности это относится к задачам с большим числом независимых переменных (практически больше двух), в которых анализ значений критерия оптимальности на границе допустимой области изменения переменных становится весьма сложным.

Метод множителей Лагранжа применяют для решения задач такого же класса сложности, как и при использовании обычных методов исследования функций, но при наличии ограничений типа равенств на независимые переменные. К требованию возможности получения аналитических выражений для производных от критерия оптимальности при этом добавляется аналогичное требование относительно аналитического вида уравнений ограничений.

В основном при использовании метода множителей Лагранжа приходится решать те же задачи, что и без ограничений. Некоторое усложнение в данном случае возникает лишь от введения дополнительных неопределенных множителей, вследствие чего порядок системы уравнений, решаемой для нахождения экстремумов критерия оптимальности, соответственно повышается на число ограничений. В остальном процедура поиска решений и проверки их на оптимальность отвечает процедуре решения задач без ограничений.

Множители Лагранжа можно применять для решения задач оптимизации объектов на основе уравнений с частными производными и задач динамической оптимизации. При этом вместо решения

системы конечных уравнений для отыскания оптимума необходимо интегрировать систему дифференциальных уравнений.

Следует отметить, что множители Лагранжа используют также в качестве вспомогательного средства и при решении специальными методами задач других классов с ограничениями типа равенств, например, в вариационном исчислении и динамическом программировании. Особенно эффективно применение множителей Лагранжа в методе динамического программирования, где с их помощью иногда удается снизить размерность решаемой задачи.

Методы вариационного исчисления обычно используют для решения задач, в которых критерии оптимальности представляются в виде функционалов и решениями которых служат неизвестные функции. Такие задачи возникают обычно при статической оптимизации процессов с распределенными параметрами или в задачах динамической оптимизации.

Вариационные методы позволяют в этом случае свести решение оптимальной задачи к интегрированию системы дифференциальных уравнений Эйлера, каждое из которых является нелинейным дифференциальным уравнением второго порядка с граничными условиями, заданными на обоих концах интервала интегрирования. Число уравнений указанной системы при этом равно числу неизвестных функций, определяемых при решении оптимальной задачи. Каждую функцию находят в результате интегрирования получаемой системы.

Уравнения Эйлера выводятся как необходимые условия экстремума функционала. Поэтому полученные интегрированием системы дифференциальных уравнений функции должны быть проверены на экстремум функционала.

При наличии ограничений типа равенств, имеющих вид функционалов, применяют множители Лагранжа, что дает возможность перейти от условной задачи к безусловной. Наиболее значительные трудности при использовании вариационных методов возникают в случае решения задач с ограничениями типа неравенств.

Заслуживают внимания прямые методы решения задач оптимизации функционалов, обычно позволяющие свести исходную вариационную задачу к задаче нелинейного программирования, решить которую иногда проще, чем краевую задачу для уравнений Эйлера.

Динамическое программирование служит эффективным методом решения задач оптимизации дискретных многостадийных процессов, для которых критерий оптимальности задается как аддитивная функция критериев оптимальности отдельных стадий. Без особых затруднений указанный метод можно распространить и на случай, когда критерий оптимальности задан в другой форме, однако при этом обычно увеличивается размерность отдельных стадий.

По существу метод динамического программирования представляет собой алгоритм определения оптимальной стратегии управления на всех стадиях процесса. При этом закон управления на каждой стадии находят путем решения частных задач оптимизации последовательно для всех стадий процесса с помощью методов исследования функций классического анализа или методов нелинейного программирования. Результаты решения обычно не могут быть выражены в аналитической форме, а получаются в виде таблиц.

Ограничения на переменные задачи не оказывают влияния на общий алгоритм решения, а учитываются при решении частных задач оптимизации на каждой стадии процесса. При наличии ограничений типа равенств иногда даже удается снизить размерность этих частных задач за счет использования множителей Лагранжа. Применение метода динамического программирования для оптимизации процессов с распределенными параметрами или в задачах динамической оптимизации приводит к решению дифференциальных уравнений в частных производных. Вместо решения таких уравнений зачастую значительно проще представить непрерывный процесс как дискретный с достаточно большим числом стадий. Подобный прием оправдан особенно в тех случаях, когда имеются ограничения на переменные задачи и прямое решение дифференциальных уравнений осложняется необходимостью учета указанных ограничений.

При решении задач методом динамического программирования, как правило, используют вычислительные машины, обладающие достаточным объемом памяти для хранения промежуточных результатов решения, которые обычно получаются в табличной форме.

Принцип максимума применяют для решения задач оптимизации процессов, описываемых системами дифференциальных уравнений. Достоинством математического аппарата принципа максимума является то, что решение может определяться в виде разрывных

функций; это свойственно многим задачам оптимизации, например, задачам оптимального управления объектами, описываемыми линейными дифференциальными уравнениями.

Нахождение оптимального решения при использовании принципа максимума сводится к задаче интегрирования системы дифференциальных уравнений процесса и сопряженной системы для вспомогательных функций при граничных условиях, заданных на обоих концах интервала интегрирования, т. е. к решению краевой задачи. На область изменения переменных могут быть наложены ограничения. Систему дифференциальных уравнений интегрируют, применяя обычные программы на цифровых вычислительных машинах.

Принцип максимума для процессов, описываемых дифференциальными уравнениями, при некоторых предположениях является достаточным условием оптимальности. Поэтому дополнительной проверки на оптимум получаемых решений обычно не требуется.

Для дискретных процессов принцип максимума в той же формулировке, что и для непрерывных, вообще говоря, несправедлив. Однако условия оптимальности, получаемые при его применении для многостадийных процессов, позволяют найти достаточно удобные алгоритмы оптимизации.

Линейное программирование представляет собой математический аппарат, разработанный для решения оптимальных задач с линейными выражениями для критерия оптимальности и линейными ограничениями на область изменения переменных. Такие задачи обычно встречаются при решении вопросов оптимального планирования производства с ограниченным количеством ресурсов, при определении оптимального плана перевозок (транспортные задачи) и т. д.

Для решения большого круга задач линейного программирования имеется практически универсальный алгоритм – *симплексный метод,* позволяющий за конечное число итераций находить оптимальное решение подавляющего большинства задач. Тип используемых ограничений (равенства или неравенства) не сказывается на возможности применения указанного алгоритма. Дополнительной проверки на оптимальность для получаемых решений не требуется. Как правило, практические задачи линейного программирования отличаются весьма значительным числом независимых

переменных. Поэтому для их решения обычно используют вычислительные машины, необходимая мощность которых определяется размерностью решаемой задачи.

Методы нелинейного программирования применяют для решения оптимальных задач с нелинейными функциями цели. На независимые переменные могут быть наложены ограничения также в виде нелинейных соотношений, имеющих вид равенств или неравенств. По существу методы нелинейного программирования используют, если ни один из перечисленных выше методов не позволяет сколько-нибудь продвинуться в решении оптимальной задачи. Поэтому указанные методы иногда называют также *прямыми методами* решения оптимальных задач.

Для получения численных результатов важное место отводится нелинейному программированию и в решении оптимальных задач такими методами, как динамическое программирование, принцип максимума и т. п. на определенных этапах их применения.

Названием «методы нелинейного программирования» объединяется большая группа численных методов, многие из которых приспособлены для решения оптимальных задач соответствующего класса. Выбор того или иного метода обусловлен сложностью вычисления критерия оптимальности и сложностью ограничивающих условий, необходимой точностью решения, мощностью имеющейся вычислительной машины и т. д. Ряд методов нелинейного программирования практически постоянно используется в сочетании с другими методами оптимизации, как например, *метод сканирования* в динамическом программировании. Кроме того, эти методы служат основой построения систем автоматической оптимизации – оптимизаторов, непосредственно применяющихся для управления производственными процессами.

Геометрическое программирование – есть метод решения одного специального класса задач нелинейного программирования, в которых критерий оптимальности и ограничения задаются в виде полиномов – выражений, представляющих собой сумму произведений степенных функций от независимых переменных. С подобными задачами иногда приходится сталкиваться в проектировании. Кроме того, некоторые задачи нелинейного программирования иногда можно свести к указанному представлению, используя аппроксимационное представление для целевых функций и ограничений.

Специфической особенностью методов решения оптимальных задач (за исключением методов нелинейного программирования) является то, что до некоторого этапа оптимальную задачу решают аналитически, т. е. находят определенные аналитические выражения, например, системы конечных или дифференциальных уравнений, откуда уже отыскивают оптимальное решение. В отличие от указанных методов при использовании методов нелинейного программирования, которые, как уже отмечалось выше, могут быть названы прямыми, применяют информацию, получаемую при вычислении критерия оптимальности, изменение которого служит оценкой эффективности того или иного действия.

Важной характеристикой любой оптимальной задачи является ее размерность *n,* равная числу переменных, задание значений которых необходимо для однозначного определения состояния оптимизируемого объекта. Как правило, решение задач высокой размерности связано с необходимостью выполнения большого объема вычислений. Ряд методов (например, динамическое программирование и дискретный принцип максимума) специально предназначен для решения задач оптимизации процессов высокой размерности, которые могут быть представлены как многостадийные процессы с относительно невысокой размерностью каждой стадии.

В табл. 7.1 дана характеристика областей применения различных методов оптимизации, при этом за основу положена сравнительная оценка эффективности использования каждого метода для решения различных типов оптимальных задач. Классификация задач проведена по следующим признакам:

- вид математического описания процесса;
- тип ограничений на переменные процесса;
- число переменных.

Предполагается, что решение оптимальной задачи для процессов, описываемых системами конечных уравнений, определяется как конечный набор значений управляющих воздействий (статическая оптимизация процессов с сосредоточенными параметрами), а для процессов, описываемых системами обыкновенных дифференциальных уравнений, управляющие воздействия характеризуются функциями времени (динамическая оптимизация процессов с сосредоточенными параметрами) или пространственных переменных (статическая оптимизация процессов с распределенными параметрами).

Таблица 7.1

Области применения методов оптимизации

Вид описания процесса		Конечное уравнение						Дифференциальное уравнение					
Тип ограничений на переменные		Нет		Равенство		Неравенство		Нет		Равенство		Неравенство	
Число переменных n		<3	>3	<3	>3	<3	>3	<3	>3	<3	>3	<3	>3
Тип метода	Методы классического анализа	1	2	4	4	4	4	3	4	4	4	4	4
	Множители Лагранжа	–	–	1	2	–	–	–	–	2	3	–	–
	Вариационное исчисление	–	–	–	–	–	–	2	3	2; 7	3; 7	–	–
	Динамическое программирование	1; 5	3; 5	1;5; 7	3;5; 7	1;5	3;5	2	3	3	3	3	3
	Принцип максимума	2; 5	1; 5	2; 5	2; 5	2; 5	2; 5	1	1	2	2	2	2
	Линейное программирование	–	–	–	2; 6	2; 6	1; 6	–	–	–	–	–	–
	Методы нелинейного программирование	2	1	2	1	2	1	4	4	4	4	4	4
	Геометрическое программирование	2; 8	2; 8	–	–	2; 8	2; 8	–	–	–	–	–	–

Окончание таблицы 7.1

Примечание:
1. Эффективное применение метода.
2. Используется.
3. Возможно применение.
4. Используется как вспомогательный метод.
5. Многостадийные процессы (размерность указывается для отдельной стадии).
6. Задачи с линейными критериями оптимальности и линейными ограничениями.
7. Используются множители Лагранжа.
8. Задачи с критериями и ограничениями в форме позиномов.

Классификация задач по группам с числом независимых переменных, большим и меньшим трех или равным трем как характеристика размерности задач с большим и малым числом переменных, разумеется, весьма условна и в данном случае выбрана скорее из соображений наглядности графического изображения пространства изменения переменных задачи – фазового пространства (при числе переменных большем трех графическое изображение фазового пространства обычными приемами отсутствует). Тем не менее, такая классификация до некоторой степени все же отражает действительные трудности, возникающие при решении задач с размерностью выше трех.

7.1.1. Численные методы безусловной оптимизации нулевого порядка

7.1.1.1. Основные определения

Решение многих теоретических и практических задач сводится к отысканию экстремума (наибольшего или наименьшего значения) скалярной функции $f(x)$ n-мерного векторного аргументах. В дальнейшем под x будем понимать вектор-столбец (точку в n-мерном пространстве):

$$x = \begin{vmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{vmatrix}. \qquad (7.1)$$

Вектор-строка получается путем применения операции транспонирования:

$$x^T = (x_1, x_2, \ldots, x_n). \qquad (7.2)$$

Оптимизируемую функцию $f(x)$ называют функцией или критерием оптимальности.

В дальнейшем без ограничения общности будем говорить о поиске минимального значения функции $f(x)$ и записывать эту задачу следующим образом: $f(x) \to \min$. Вектор x^*, определяющий минимум целевой функции, называют оптимальным.

Отметим, что задачу максимизации $f(x)$ можно заменить эквивалентной ей задачей минимизации или наоборот. Рассмотрим это на примере функции одной переменной (рис. 7.1). Если x^* – точка минимума функции $y = f(x)$, то для функции $y = -f(x)$ она является точкой максимума, так как графики функций $f(x)$ и $-f(x)$, симметричны относительно оси абсцисс. Итак, минимум функции $f(x)$ и максимум функции $-f(x)$ достигаются при одном и том же значении переменной. Минимальное же значение функции $f(x)$, равно максимальному значению функции $-f(x)$, взятому с противоположным знаком, т.е. $\min f(x) = -\max(f(x))$.

Рассуждая аналогично, этот вывод нетрудно распространить на случай функции многих переменных. Если требуется заменить задачу минимизации функции $f(x_1, \ldots, x_n)$ задачей максимизации, то достаточно вместо отыскания минимума этой функции найти максимум функции $f(x_1, \ldots, x_n)$. Экстремальные значения этих функций достигаются при одних и тех же значениях переменных. Минимальное значение функции $f(x_1, \ldots, x_n)$ равно максимальному значению функции $-f(x_1, \ldots, x_n)$, взятому с обратным знаком, т. е. $\min f(x_1, \ldots, x_n) = \max f(x_1, \ldots, x_n)$. Отмеченный факт позволяет в дальнейшем говорить только о задаче минимизации.

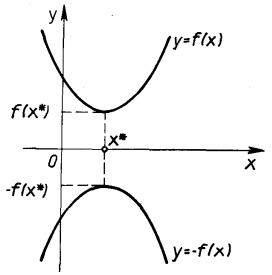


Рис. 7.1. Экстремум функции одной переменной

В реальных условиях на переменные x_i, $i=1, \ldots, n$, и некоторые функции $g_i(x)$, $h_i(x)$, характеризующие качественные свойства объекта, системы, процесса, могут быть наложены ограничения (условия) вида:

$$\begin{gathered} g_i(x) = 0,\ i=1, \ldots n; \\ h_i(x) <= 0,\ i=1, \ldots n; \\ a <= x <= b, \end{gathered} \qquad (7.3)$$

где

$$a = \begin{vmatrix} a_1 \\ a_2 \\ \ldots \\ a_n \end{vmatrix},\ b = \begin{vmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{vmatrix}. \qquad (7.4)$$

Такую задачу называют *задачей условной оптимизации.* При отсутствии ограничений имеет место *задача безусловной оптимизации.*

Каждая точка x в n-мерном пространстве переменных $x_1, \ldots, x_n$, в которой выполняются ограничения, называется *допустимой точкой задачи.* Множество всех допустимых точек называют *допустимой областью G. Решением задачи (оптимальной точкой)* называют допустимую точку x^*, в которой целевая функция $f(x)$ достигает своего минимального значения.

Точка x^* определяет *глобальный минимум* функции одной переменной $f(x)$, заданной на числовой прямой X, если $x^* \in X$ и $f(x^*) < f(x)$ для всех $x^* \in X$ (рис. 7.2, *а*). Точка x^* называется *точкой строгого глобального минимума,* если это неравенство выполняется как строгое. Если же в выражении $f(x^*) <= f(x)$ равенство возможно при x, *не равных* x^*, то реализуется *нестрогий минимум,* а под решением в этом случае понимают множество $x^* = [x^* \in X : f(x) = f(x^*)]$ (рис. 7.2, *б*).

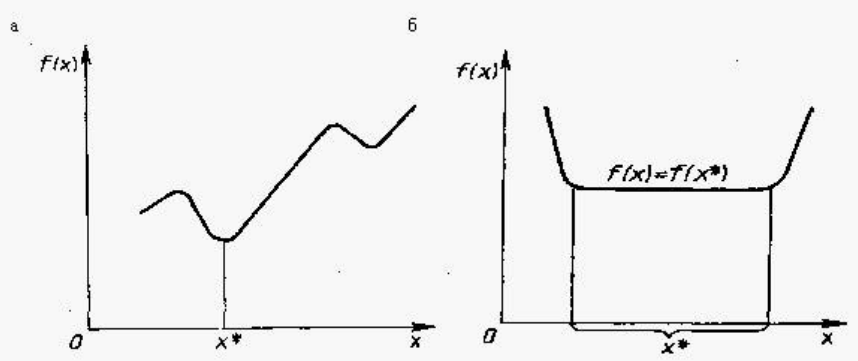


Рис. 7.2. Глобальный минимум: *а* – строгий, *б* – нестрогий

Точка $x^* \in X$ определяет *локальный минимум* функции $f(x)$ на множестве X, если при некотором достаточно малом $\varepsilon > 0$ для всех x, *не равных* x^*, $x \in X$, удовлетворяющих условию $|x - x^*| <= \varepsilon$, выполняется неравенство $f(x^*) < f(x)$. Если неравенство строгое, то x^* является точкой строгого локального минимума. Все определения для максимума функции получаются заменой знаков предыдущих неравенств на обратные. На рис. 7.3 показаны экстремумы функции одной переменной $f(x)$ на отрезке $[a, b]$. Здесь x_1, x_3, x_6 – точки локального максимума, а x_2, x_4 – локального минимума. В точке x_6 реализуется глобальный максимум, а в точке x_2 – глобальный минимум.

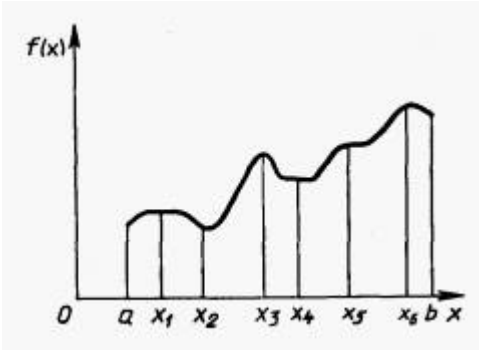


Рис. 7.3. Экстремумы функции

7.1.1.2. Классификация методов

Возможны два подхода к решению задачи отыскания минимума функции многих переменных $f(x) = f(x_1, ..., x_n)$ при отсутствии ограничений на диапазон изменения неизвестных. Первый подход лежит в основе *косвенных методов оптимизации* и сводит решение задачи оптимизации к решению системы нелинейных уравнений, являющихся следствием условий экстремума функции многих переменных. Как известно, эти условия определяют, что в точке экстремума x^* все первые производные функции по независимым переменным равны нулю:

$$\left.\frac{\partial f}{\partial x_i}\right|_{x=x^*} = 0,\ i=1,\ ...,\ n. \qquad (7.5)$$

Эти условия образуют систему n нелинейных уравнений, среди решений которой находятся точки минимума переменной.

$$f'(x) = \left(\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n}\right)^T, \qquad (7.6)$$

называют градиентом скалярной функции $f(x)$. Как видно, в точке минимума градиент равен нулю.

Решение систем нелинейных уравнений – задача весьма сложная и трудоемкая. Вследствие этого на практике используют второй подход к минимизации функций, составляющий основу *прямых методов.* Суть их состоит в построении последовательности векторов $x[0]$, $x[1]$, …, $x[n]$, таких, что $f(x[0]) > f(x[1]) > f(x[n]) > \ldots$ В качестве начальной точки $x[0]$ может быть выбрана произвольная точка, однако стремятся использовать всю имеющуюся информацию о поведении функции $f(x)$, чтобы точка $x[0]$ располагалась как можно ближе к точке минимума. Переход (итерация) от точки $x[k]$ к точке $x[k+1]$, $k = 0, 1, 2, ...$, состоит из двух этапов:

1. выбор направления движения из точки $x[k]$;
2. определение шага вдоль этого направления.

Методы построения таких последовательностей часто называют *методами спуска,* так как осуществляется переход от больших значений функций к меньшим.

Математически методы спуска описываются соотношением:

$$x[k+1] = x[k] + a_k p[k],\ k = 0, 1, 2, \ldots ,$$

где $p[k]$ – вектор, определяющий направление спуска;

a_k – длина шага.

В координатной форме:

$$\left| \begin{array}{l} x_1[k+1] = x_1[k] + a_k p_1[k] \\ x_2[k+1] = x_2[k] + a_k p_2[k] \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ x_n[k+1] = x_n[k] + a_k p_n[k] \end{array} \right| . \qquad (7.7)$$

Различные методы спуска отличаются друг от друга способами выбора двух параметров – направления спуска и длины шага вдоль этого направления. На практике применяются только методы, обладающие сходимостью. Они позволяют за конечное число шагов получить точку минимума или подойти к точке, достаточно близкой к точке минимума. Качество сходящихся итерационных методов оценивают по скорости сходимости.

В методах спуска решение задачи теоретически получается за бесконечное число итераций. На практике вычисления прекращаются при выполнении некоторых критериев (условий) останова итерационного процесса. Например, это может быть условие малости приращения аргумента:

$$x[k] - x[k-1] < \varepsilon, \qquad (7.8)$$

или функции:

$$f\left(x[k]\right) - f\left(x[k-1]\right) < \gamma, \qquad (7.9)$$

где k – номер итерации;

ε, γ – заданные величины точности решения задачи.

Методы поиска точки минимума называются *детерминированными,* если оба элемента перехода от $x[k]$ к $x[k + 1]$ (направление движения и величина шага) выбираются однозначно по доступной

в точке x $[k]$ информации. Если же при переходе используется какой-либо случайный механизм, то алгоритм поиска называется *случайным поиском минимума.*

Детерминированные алгоритмы безусловной минимизации делят на классы в зависимости от вида используемой информации. Если на каждой итерации используются лишь значения минимизируемых функций, то метод называется *методом нулевого порядка.* Если требуется вычисление первых производных минимизируемой функции, то имеют место методы *первого порядка,* при необходимости дополнительного вычисления вторых производных – *методы второго порядка.*

В настоящее время разработано множество численных методов для задач как безусловной, так и условной оптимизации. Естественным является стремление выбрать для решения конкретной задачи наилучший метод, позволяющий за наименьшее время использования ЭВМ получить решение с заданной точностью.

Качество численного метода характеризуется многими факторами: скоростью сходимости, временем выполнения одной итерации, классом решаемых задач и т. д. Решаемые задачи также весьма разнообразны: они могут иметь высокую и малую размерность, быть унимодальными (обладающими одним экстремумом) и многоэкстремальными и т. д. Один и тот же метод, эффективный для решения задач одного типа, может оказаться совершенно неприемлемым для задач другого типа. Очевидно, что разумное сочетание разнообразных методов, учет их свойств позволят с наибольшей эффективностью решать поставленные задачи. Многометодный способ решения весьма удобен в диалоговом режиме работы с ЭВМ. Для успешной работы в таком режиме очень полезно знать основные свойства, специфику методов оптимизации. Это обеспечивает способность правильно ориентироваться в различных ситуациях, возникающих в процессе расчетов, и наилучшим образом решить задачу.

7.1.1.3. Общая характеристика методов нулевого порядка

В этих методах для определения направления спуска не требуется вычислять производные целевой функции. Направление минимизации в данном случае полностью определяется последовательными вычислениями значений функции. Следует отметить, что при

решении задач безусловной минимизации методы первого и второго порядков обладают, как правило, более высокой скоростью сходимости, чем методы нулевого порядка. Однако на практике вычисление первых и вторых производных функции большого количества переменных весьма трудоемко. В ряде случаев они не могут быть получены в виде аналитических функций. Определение производных с помощью различных численных методов осуществляется с ошибками, которые могут ограничить применение таких методов. Кроме того, на практике встречаются задачи, решение которых возможно лишь с помощью методов нулевого порядка, например задачи минимизации функций с разрывными первыми производными. Критерий оптимальности может быть задан не в явном виде, а системой уравнений. В этом случае аналитическое или численное определение производных становится очень сложным, а иногда невозможным. Для решения таких практических задач оптимизации могут быть успешно применены методы нулевого порядка. Рассмотрим некоторые из них.

7.1.1.4. Метод прямого поиска (метод Хука – Дживса)

Суть этого метода состоит в следующем. Задаются некоторой начальной точкой $x[0]$. Изменяя компоненты вектора $x[0]$, обследуют окрестность данной точки, в результате чего находят направление, в котором происходит уменьшение минимизируемой функции $f(x)$. В выбранном направлении осуществляют спуск до тех пор, пока значение функции уменьшается. После того как в данном направлении не удается найти точку с меньшим значением функции, уменьшают величину шага спуска. Если последовательные дробления шага не приводят к уменьшению функции, от выбранного направления спуска отказываются и осуществляют новое обследование окрестности и т. д.

Алгоритм метода прямого поиска состоит в следующем.

1. Задаются значениями координат $x_i[0]$, $i = 1, ..., n$, начальной точки $x[0]$, вектором изменения координат Δx в процессе обследования окрестности, наименьшим допустимым значением e компонентов Δx.

2. Полагают, что $x[0]$ является *базисной точкой* $x^{б}$ и вычисляют значение $f(x^{б})$.

3. Циклически изменяют каждую координату $x^{б}_i$ $i = 1, ..., n$, базисной точки $x^{б}$ на величину Δx_i, $i = 1, ..., n$, т. е. $x_i[k] = x^{б} + \Delta x$; $x_i[k] = x^{б}_i - \Delta x_i$. При этом вычисляют значения $f(x[k])$ и сравнивают их со значением $f(x^{б})$. Если $f(x[k]) < f(x^{б})$, то соответствующая координата x_i, i = 1, ..., n , приобретает новое значение, вычисленное по одному из приведенных выражений. В противном случае значение этой координаты остается неизменным. Если после изменения последней n-й координаты $f(x[k]) < f(x^{б})$, то переходят к п. 4. В противном случае – к п. 7.

4. Полагают, что $x[k]$ является новой базисной точкой $x^{б}$, и вычисляют значение $f(x^{б})$.

5. Осуществляют спуск из точки $x[k] > x_i[k + 1] = 2x_i[k] - x^{б}$, i = 1, ..., n , где $x^{б}$ – координаты предыдущей базисной точки. Вычисляют значение $f(x[k + 1])$.

6. Как и в п. 3, циклически изменяют каждую координату точки $x[k + 1]$, осуществляя сравнение соответствующих значений функции f(x) со значением f $(x[k + 1])$, полученным в п. 5. После изменения последней координаты сравнивают соответствующее значение функции $f(x[k])$ со значением $f(x^{б})$, полученным в п. 4. Если $f(x[k]) < f(x^{б})$, то переходят к п. 4, в противном случае – к п. 3. При этом в качестве базисной используют последнюю из полученных базисных точек.

7. Сравнивают значения Δx и e. Если $\Delta x < e$, то вычисления прекращаются. В противном случае уменьшают значения Δx и переходят к п. 3.

Достоинством метода прямого поиска является простота его программирования на компьютере. Он не требует знания целевой функции в явном виде, а также легко учитывает ограничения на отдельные переменные, а также сложные ограничения на область поиска.

Недостаток метода прямого поиска состоит в том, что в случае сильно вытянутых, изогнутых или обладающих острыми углами линий уровня целевой функции он может оказаться неспособным обеспечить продвижение к точке минимума. Действительно, в случаях, изображенных на рис. 7.4, *а* и *б,* каким бы малым не был шаг в направлении x_1 или x_2 из точки x нельзя получить уменьшения значения целевой функции.

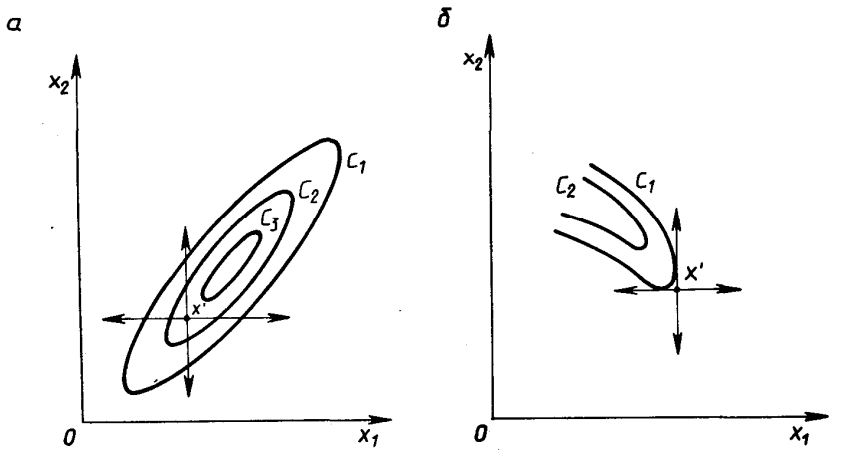


Рис. 7.4. Прямой поиск: невозможность продвижения к минимуму:
а – $C_1 > C_2 > C_3$; *б* – $C_1 > C_2$

Напомним, что *поверхностью уровня* (на плоскости – *линией уровня)* является поверхность, получаемая приравниванием выражения функции $f(x)$ некоторой постоянной величине C, т. е. $f(x) = C$. Во всех точках этой поверхности функция имеет одно и то же значение C. Давая величине C различные значения C_1, ..., C_n, получают ряд поверхностей, геометрически иллюстрирующих характер функции.

7.1.1.5. Метод деформируемого многогранника (метод Нелдера – Мида)

Данный метод состоит в том, что для минимизации функции n переменных $f(x)$ в n-мерном пространстве строится многогранник, содержащий $(n + 1)$ вершину. Очевидно, что каждая вершина соответствует некоторому вектору x. Вычисляются значения целевой функции $f(x)$ в каждой из вершин многогранника, определяются максимальное из этих значений и соответствующая ему вершина $x[h]$. Через эту вершину и центр тяжести остальных вершин проводится проецирующая прямая, на которой находится точка $x[q]$ с меньшим значением целевой функции, чем в вершине $x[h]$ (рис. 7.5). Затем исключается вершина $x[h]$. Из оставшихся вершин и точки $x[q]$ строится новый

многогранник, с которым повторяется описанная процедура. В процессе выполнения данных операций многогранник изменяет свои размеры, что и обусловило название метода.

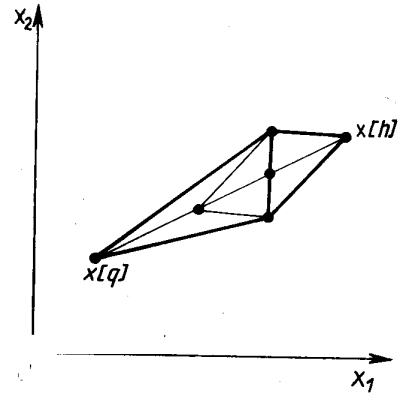


Рис. 7.5. Геометрическая интерпретация метода деформируемого многогранника

Введем следующие обозначения:

$$x[i, k] = (x_1[i, k], \ldots, x_j[i, k], \ldots, x_n[i, k])^T$$

где $i = 1, ..., n + 1$; $k = 0, 1, ...,$ – i-я вершина многогранника на k-м этапе поиска; $x[h, k]$ — вершина, в которой значение целевой функции максимально, т. е. $f(x[h, k] = \max\{f(x[1, k]), \ldots, f(x[n + 1, k])\}$; $x[l, k]$ – вершина, в которой значение целевой функции минимально, т. е. $f(x[l, k]) = \min\{f(x[1, k]), \ldots, f(x[n + 1, k])\}$; $x[n + 2, k]$- центр тяжести всех вершин, за исключением $x[h, k]$. Координаты центра тяжести вычисляются по формуле:

$$x_j[n+2,k] = \frac{1}{n}\left(\sum_{i=1}^{n+1} x_j[i,k] - x_j[h,k]\right), j = 1,\ldots,n$$

Алгоритм метода деформируемого многогранника состоит в следующем:

1. Осуществляют проецирование точки $x[h, k]$ через центр тяжести:

$$x[n + 3, k] = x[n + 2, k] + a(x[n + 2, k] - x[h, k]),$$

где $a > 0$ – некоторая константа. Обычно $a = 1$.

2. Выполняют операцию растяжения вектора $x[n+3, k] - x[n+2, k]$:

$$x[n + 4, k] = x[n + 2, k] + \gamma(x[n + 3, k] - x[n + 2, k]),$$

где $\gamma > 1$ – коэффициент растяжения. Наиболее удовлетворительные результаты получают при $2{,}8 <= \gamma <= 3$.

Если $f(x[n + 4, k]) < f(x[l, k])$, то $x[h, k]$ заменяют на $x[n + 4, k]$ и продолжают вычисления с п. 1 при $k = k + 1$. В противном случае $x[h, k]$ заменяют на $x[n + 3, k]$ и переходят к п. 1 при $k = k + 1$.

3. Если $f(x[n + 3, k]) > f(x[i, k])$ для всех i, не равных h, то сжимают вектор $x[h, k] - x[n+2, k]$:

$$x[n+ 5, k] = x[n + 2, k] + \beta\,(x[h, k] - x[n + 2, k]),$$

где $\beta > 0$ – коэффициент сжатия. Наиболее хорошие результаты получают при $0{,}4 <= \beta <= 0{,}6$.

Затем точку $x[h, k]$ заменяют на $x[n + 5, k]$ и переходят к п. 1 при $k = k + 1$.

4. Если $f(x[n + 3, k]) > f(x[h, k])$, то все векторы $x[i, k] - x[l, k]$. $i = 1,..., n + 1$, уменьшают в два раза:

$$x[i, k] = x[l, k] + 0{,}5(x[i, k] - x[l, k]).$$

Затем переходят к п. 1 при $k = k + 1$.

В диалоговой системе оптимизации выход из подпрограммы, реализующей метод деформируемого многогранника, осуществляется при предельном сжатии многогранника, т. е. при выполнении условия:

$$\max \sum_{j=1}^{n} \left(x_j[i,k] - x_j[n+2,k] \right)^2 < \sum_{j=1}^{n} e_j^2, \qquad (7.10)$$

где $e = (e_1, ..., e_n)$ – заданный вектор.

С помощью операции растяжения и сжатия размеры и форма деформируемого многогранника адаптируются к топографии целевой функции. В результате многогранник вытягивается вдоль длинных наклонных поверхностей, изменяет направление в изогнутых впадинах, сжимается в окрестности минимума, что определяет эффективность рассмотренного метода.

7.1.1.6. Метод вращающихся координат (метод Розенброка)

Суть метода состоит во вращении системы координат в соответствии с изменением скорости убывания целевой функции. Новые направления координатных осей определяются таким образом, чтобы одна из них соответствовала направлению наиболее быстрого убывания целевой функции, а остальные находятся из условия ортогональности. Идея метода состоит в следующем (рис. 7.6).

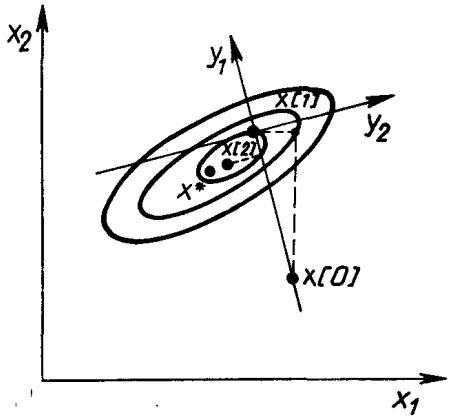


Рис. 7.6. Геометрическая интерпретация метода Розенброка

Из начальной точки $x[0]$ осуществляют спуск в точку $x[1]$ по направлениям, параллельным координатным осям. На следующей итерации одна из осей должна проходить в направлении $y_1 = x[1] - x[0]$, а другая – в направлении, перпендикулярном к y_1. Спуск вдоль этих осей приводит в точку $x[2]$, что дает возможность построить новый вектор $x[2] - x[1]$ и на его базе новую систему направлений поиска. В общем случае данный метод эффективен при минимизации овражных функций, так как результирующее направление поиска стремится расположиться вдоль оси оврага.

Алгоритм метода вращающихся координат состоит в следующем.

1. Обозначают через $p_1[k], ..., p_n[k]$ направления координатных осей в некоторой точке $x[k]$ (на k-й итерации). Выполняют пробный шаг h_1 вдоль оси $p_1[k]$, т. е.

$x[kl] = x[k] + h_1 p_1[k]$.

Если при этом $f(x[k1]) < f(x[k])$, то шаг h умножают на величину $\beta > 1$;

Если $f(x[k_1]) > f(x[k])$, то на величину $(-\beta)$, $0 < |\beta| < 1$;

$x[k_1] = x[k] + \beta\, h_1 p_1[k]$.

Полагая $\beta\, h_1 = a_1$, получают

$x[k_1] = x[k] + a_1 p_1[k]$.

2. Из точки $x[k1]$ выполняют шаг h_2 вдоль оси $p_2[k]$:

$x[k_2] = x[k] + a_1 p_1[k] + h_2 p_2[k]$.

Повторяют операцию п. 1, т. е.

$x[k_2] = x[k] + a_1 p_1[k] + a_2 p_2[k]$.

Эту процедуру выполняют для всех остальных координатных осей. На последнем шаге получают точку

$$x[kn] = x[k+1] = x[k] + \sum_{i=1}^{n} a_i p_i[k].$$

3. Выбирают новые оси координат $p_1[k+1]$, …, $p_n[k+1]$. В качестве первой оси принимается вектор

$p_1[k+1] = x[k+1] - x[k]$.

Остальные оси строят ортагональными к первой оси с помощью процедуры ортогонализации Грама – Шмидта. Повторяют вычисления с п. 1 до удовлетворения условий сходимости.

Коэффициенты β подбираются эмпирически. Хорошие результаты дают значения $\beta = -0{,}5$ при неудачных пробах ($f(x[ki]) > f(x[k])$) и $\beta = 3$ при удачных пробах ($f(x[ki]) < f(x[k])$).

В отличие от других методов нулевого порядка алгоритм Розенброка ориентирован на нахождение оптимальной точки в каждом направлении, а не просто на фиксированный сдвиг по всем направлениям. Величина шага в процессе поиска непрерывно изменяется в зависимости от рельефа поверхности уровня. Сочетание вращения координат с регулированием шага делает метод Розенброка эффективным при решении сложных задач оптимизации.

7.1.1.7. Метод параллельных касательных (метод Пауэлла)

Этот метод использует свойство квадратичной функции, заключающееся в том, что любая прямая, которая проходит через точку минимума

функции x^*, пересекает под равными углами касательные к поверхностям равного уровня функции в точках пересечения (рис. 7.7):

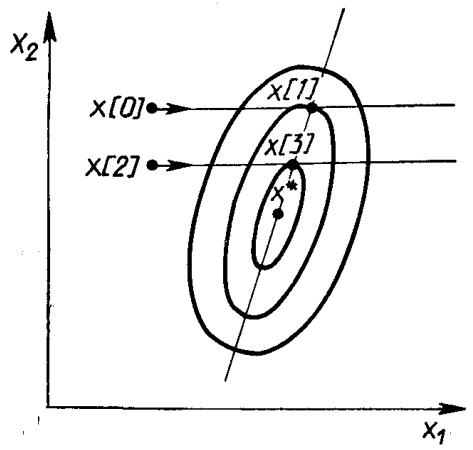


Рис. 7.7. Геометрическая интерпретация метода Пауэлла

Суть метода такова. Выбирается некоторая начальная точка $x[0]$ и выполняется одномерный поиск вдоль произвольного направления, приводящий в точку $x[1]$. Затем выбирается точка $x[2]$, не лежащая на прямой $x[0] - x[1]$, и осуществляется одномерный поиск вдоль прямой, параллельной $x[0] - x[1]$. Полученная в результате точка $x[3]$ вместе с точкой $x[1]$ определяет направление $x[1] - x[3]$ одномерного поиска, дающее точку минимума x^*. В случае квадратичной функции n переменных оптимальное значение находится за n итераций. Поиск минимума при этом в конечном счете осуществляется во взаимно сопряженных направлениях. В случае неквадратичной целевой функции направления поиска оказываются сопряженными относительно матрицы Гессе. Алгоритм метода параллельных касательных состоит в следующем.

1. Задаются начальной точкой $x[0]$. За начальные направления поиска $p[1]$, ... , $p[0]$ принимают направления осей координат, т. е. $p[i] = e[i]$, $i = 1, ..., n$ (здесь $e[i] = (0, ..., 0, 1, 0, \dots 0)^T$).

2. Выполняют n одномерных поисков вдоль ортогональных направлений $p[i]$, $i = 1, ..., n$. При этом каждый следующий поиск производится из точки минимума, полученной на предыдущем шаге. Величина шага a_k находится из условия:

$$f\left(x[k]+a_k p[k]\right)=\min_a f\left(x[k]+ap[k]\right).$$

Полученный шаг определяет точку:

$$x[k+1]=x[k]+a_k p[k]\ .$$

3. Выбирают новое направление $p = -x[n] - x[0]$ и заменяют направления $p[1]$, ..., $p[n]$ на $p[2]$, ..., $p\ [n]$, p. Последним присваивают обозначения $p[1]$, ..., $p[n]$.

4. Осуществляют одномерный поиск вдоль направления $p = p[n] = x[n] - x[0]$. Заменяют $x[0]$ на $x[n+1] = x[n] + a_n p[n]$ и принимают эту точку за начальную точку $x[0]$ для следующей итерации. Переходят к п. 1.

Таким образом в результате выполнения рассмотренной процедуры осуществляется поочередная замена принятых вначале направлений поиска. В итоге после n шагов они окажутся взаимно сопряженными.

7.1.2. Численные методы безусловной оптимизации первого порядка

7.1.2.1. Минимизация функций многих переменных. Основные положения

Градиентом дифференцируемой функции $f(x)$ в точке $x[0]$ называется n-мерный вектор $f(x[0])$, компоненты которого являются частными производными функции $f(x)$, вычисленными в точке $x[0]$, т. е.

$$f'(x[0]) = (\partial f(x[0])/\partial x_1, \ldots, \partial f(x[0])/\partial x_n)^T. \qquad (7.11)$$

Этот вектор перпендикулярен к плоскости, проведенной через точку $x[0]$ и касательной к поверхности уровня функции $f(x)$, проходящей через точку $x[0]$. В каждой точке такой поверхности функция $f(x)$ принимает одинаковое значение. Приравнивая функцию различным постоянным величинам C_0, C_1, ... , получим серию поверхностей, характеризующих ее топологию (рис. 7.8).

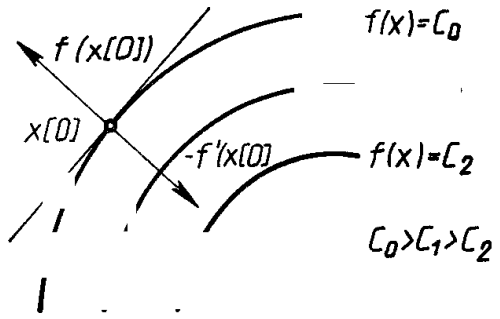


Рис. 7.8. Градиент

Вектор-градиент направлен в сторону наискорейшего возрастания функции в данной точке. Вектор, противоположный градиенту ($-f'(x[0])$), называется *антиградиентом* и направлен в сторону наискорейшего убывания функции. В точке минимума градиент функции равен нулю. На свойствах градиента основаны методы первого порядка, называемые также градиентными и методами минимизации. Использование этих методов в общем случае позволяет определить точку локального минимума функции.

Очевидно, что если нет дополнительной информации, то из начальной точки $x[0]$ разумно перейти в точку $x[1]$, лежащую в направлении антиградиента – наискорейшего убывания функции. Выбирая в качестве направления спуска $p[k]$ антиградиент $-f'(x[k])$ в точке $x[k]$, получаем итерационный процесс вида

$$x[k+1] = x[k] - a_k f'(x[k]),\ a_k > 0;\ k = 0, 1, 2, \ldots$$

В координатной форме этот процесс записывается следующим образом:

$$x_i\left[k+1\right] = x_i\left[k\right] - a_k \frac{df\left(x\left[k\right]\right)}{dx_i},$$

где $i = 1, \ldots, n$; $k = 0, 1, 2, \ldots$.

В качестве критерия останова итерационного процесса используют либо выполнение условия малости приращения аргумента $\| x[k+1] - x[k] \| <= \varepsilon$, либо выполнение условия малости градиента

$$\left\| f'\left(x\left[k+1\right]\right)\right\| \le \gamma,$$

где ε и γ – заданные малые величины.

Возможен и комбинированный критерий, состоящий в одновременном выполнении указанных условий. Градиентные методы отличаются друг от друга способами выбора величины шага a_k.

При методе с постоянным шагом для всех итераций выбирается некоторая постоянная величина шага. Достаточно малый шаг a_k обеспечит убывание функции, т. е. выполнение неравенства

$$f(x[k+1]) = f(x[k] - a_k f'(x[k])) < f(x[k]).$$

Однако это может привести к необходимости проводить неприемлемо большое количество итераций для достижения точки минимума. С другой стороны, слишком большой шаг может вызвать неожиданный рост функции либо привести к колебаниям около точки минимума (зацикливанию). Из-за сложности получения необходимой информации для выбора величины шага методы с постоянным шагом применяются на практике редко.

Более экономичны в смысле количества итераций и надежности градиентные *методы с переменным шагом,* когда в зависимости от результатов вычислений величина шага некоторым образом меняется. Рассмотрим применяемые на практике варианты таких методов.

7.1.2.2. Метод наискорейшего спуска

При использовании метода наискорейшего спуска на каждой итерации величина шага *ak* выбирается из условия минимума функции *f*(*x*) в направлении спуска, т.е.:

$$f\left(x\left[k\right] - a_k f'\left(x\left[k\right]\right)\right) = \min_{a>0} f\left(x\left[k\right] - a_k f'\left(x\left[k\right]\right)\right).$$

Это условие означает, что движение вдоль антиградиента происходит до тех пор, пока значение функции f(*x*) убывает. С математической точки зрения на каждой итерации необходимо решать задачу одномерной минимизации по a функции:

$$\varphi(a) = f\left(x\left[k\right] - a_k f'\left(x\left[k\right]\right)\right).$$

Алгоритм метода наискорейшего спуска состоит в следующем.

1. Задаются координаты начальной точки *x*[0].
2. В точке *x*[*k*], *k* = 0, 1, 2, ... вычисляется значение градиента *f*′(*x*[*k*]).
3. Определяется величина шага a_k, путем одномерной минимизации по *a* функции φ(*a*) = *f*(*x*[*k*] – *af*′(*x*[*k*])).
4. Определяются координаты точки *x*[*k* + 1]:
5. $x_i[k+1] = x_i[k] - a_k f'_i(x[k]),\ i = 1, ..., n.$
6. Проверяются условия останова процесса. Если они выполняются, то вычисления прекращаются. В противном случае осуществляется переход к п. 1.

В рассматриваемом методе направление движения из точки *x*[*k*] касается линии уровня в точке *x*[*k*+1] (рис. 7.9). Траектория спуска зигзагообразная, причем соседние звенья зигзага ортогональны друг другу. Действительно, шаг a_k выбирается путем минимизации по *a* функции *f*(*a*) = φ(*x*[*k*] – *af*′(*x*[*k*])). Необходимое условие минимума функции *dφ(a)/da* = 0. Вычислив производную сложной функции, получим условие ортогональности векторов направлений спуска в соседних точках:

$$d\varphi(a)/da = -f'(x[k+1]\, f'(x[k]) = 0.$$

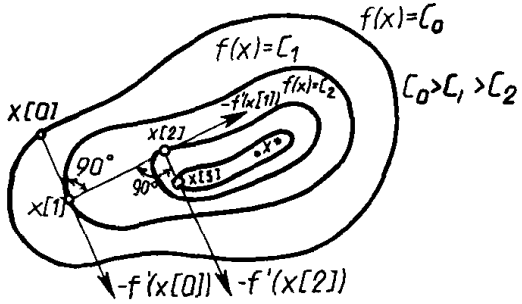


Рис. 7.9. Геометрическая интерпретация метода наискорейшего спуска

Градиентные методы сходятся к минимуму с высокой скоростью (со скоростью геометрической прогрессии) для гладких выпуклых функций. У таких функций наибольшее M и наименьшее m собственные значения матрицы вторых производных (матрицы Гессе) мало отличаются друг от друга, т. е. матрица $H(x)$ хорошо обусловлена. Напомним, что собственными значениями λ_i, $i = 1, \ldots, n$, матрицы являются корни характеристического уравнения

$$H(x) = \begin{vmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{vmatrix} \qquad (7.12)$$

$$\left\| \begin{matrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{matrix} \right\| = 0. \qquad (7.13)$$

Однако на практике, как правило, минимизируемые функции имеют плохо обусловленные матрицы вторых производных $(m/M << 1)$. Значения таких функций вдоль некоторых направлений изменяются гораздо быстрее (иногда на несколько порядков), чем в других направлениях. Их поверхности уровня в простейшем случае сильно вытягиваются (рис. 7.10), а в более сложных случаях изгибаются и представляют собой овраги. Функции, обладающие такими свойствами, называют *овражными.* Направление антиградиента этих функций (см. рис. 7.10) существенно отклоняется от направления в точку минимума, что приводит к замедлению скорости сходимости.

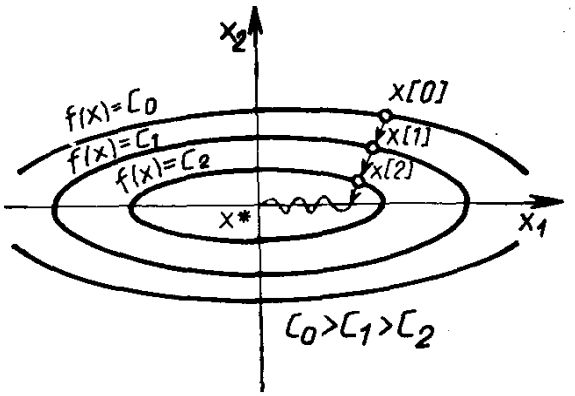


Рис. 7.10. Овражная функция

Скорость сходимости градиентных методов существенно зависит также от точности вычислений градиента. Потеря точности, а это обычно происходит в окрестности точек минимума или в овражной ситуации, может вообще нарушить сходимость процесса градиентного спуска. Вследствие перечисленных причин градиентные методы зачастую используются в комбинации с другими, более эффективными методами на начальной стадии решения задачи. В этом случае точка $x[0]$ находится далеко от точки минимума, и шаги в направлении антиградиента позволяют достичь существенного убывания функции.

7.1.2.3. Метод сопряженных градиентов

Рассмотренные выше градиентные методы отыскивают точку минимума функции в общем случае лишь за бесконечное число итераций. Метод сопряженных градиентов формирует направления поиска, в большей мере соответствующие геометрии минимизируемой функции. Это существенно увеличивает скорость их сходимости и позволяет, например, минимизировать квадратичную функцию

$$f(x) = (x, Hx) + (b, x) + a,$$

с симметрической положительно определенной матрицей H за конечное число шагов n, равное числу переменных функции. Любая

гладкая функция в окрестности точки минимума хорошо аппроксимируется квадратичной, поэтому методы сопряженных градиентов успешно применяют для минимизации и неквадратичных функций. В таком случае они перестают быть конечными и становятся итеративными.

По определению, два n-мерных вектора x и y называют *сопряженными* по отношению к матрице H (или H-сопряженными), если скалярное произведение $(x, Hy) = 0$. Здесь H – симметрическая положительно определенная матрица размером n на n.

Одной из наиболее существенных проблем в методах сопряженных градиентов является проблема эффективного построения направлений. Метод Флетчера – Ривса решает эту проблему путем преобразования на каждом шаге антиградиента $-f(x[k])$ в направление $p[k]$, H-сопряженное с ранее найденными направлениями $p[0]$, $p[1]$, ..., $p[k-1]$. Рассмотрим сначала этот метод применительно к задаче минимизации квадратичной функции.

Направления $p[k]$ вычисляют по формулам:

$$p[k] = -f(x[k]) + \beta_{k-1}p[k-1],\ k >= 1;$$
$$p[0] = -f'(x[0]).$$

Величины β_{k-1} выбираются так, чтобы направления $p[k]$, $p[k-1]$ были H-сопряженными:

$$(p[k], Hp[k-1]) = 0.$$

В результате для квадратичной функции:

$$\beta_{k-1} = \frac{\left(f'\left(x[k], f'\left(x[k]\right)\right)\right)}{\left(f'\left(x[k-1], f'\left(x[k-1]\right)\right)\right)},$$

итерационный процесс минимизации имеет вид:

$$x[k+1] = x[k] + a_k p[k],$$

где $p[k]$ – направление спуска на k-м шаге;

a_k – величина шага, которая выбирается из условия минимума функции $f(x)$ по a в направлении движения, т. е. в результате решения задачи одномерной минимизации:

$$f\left(x[k] + a_k p[k]\right) = \min_{a \geq 0} f\left(x[k] + ap[k]\right).$$

Для квадратичной функции:

$$a_k = -\frac{f'\left(x[k], p[k]\right)}{\left(p[k], Hp[k],\right)}. \quad (7.14)$$

Алгоритм метода сопряженных градиентов Флетчера – Ривса состоит в следующем:

1. В точке $x[0]$ вычисляется $p[0] = -f'(x[0])$.
2. На k-м шаге по приведенным выше формулам определяются шаг a_k и точка $x[k+1]$.
3. Вычисляются величины $f(x[k+1])$ и $f'(x[k+1])$.
4. Если $f'(x[k+1]) = 0$, то точка $x[k+1]$ является точкой минимума функции $f(x)$. В противном случае определяется новое направление $p[k+1]$ из соотношения:

$$p[k+1] = -f'\left(x[k+1]\right)\frac{\left(f'\left(x[k+1]\right), f'\left(x[k+1]\right)\right)}{\left(f'\left(x[k]\right), f'\left(x[k]\right)\right)}p[k], \quad (7.15)$$

и осуществляется переход к следующей итерации. Эта процедура найдет минимум квадратичной функции не более чем за n шагов. При минимизации неквадратичных функций метод Флетчера – Ривса из конечного становится итеративным. В таком случае после $(n+1)$-й итерации процедуры 1-4 циклически повторяются с заменой $x[0]$ на $x[n+1]$, а вычисления заканчиваются при $\left\|f'\left(x[k]\right)\right\| < \varepsilon$, где ε – заданное число. При этом применяют следующую модификацию метода:

$$x[k+1] = x[k] + a_k p[k],$$
$$p[k] = -f'(x[k]) + \beta_{k-1}p[k-1],\ k >= 1;$$
$$p[0] = -f'(x[0]);$$
$$f(x[k] + a_k p[k]) = \min_{a \geq 0} f(x[k] + ap[k]);$$

$$\beta_{k-1}=\begin{cases}\dfrac{\left(f'\left(x[k]\right),f'\left(x[k]\right)-f'\left(x[k-1]\right)\right)}{f'\left(x[k]\right),f'\left(x[k]\right)}, k\notin I.\\ 0, k\in I.\end{cases} \qquad (7.16)$$

Здесь I – множество индексов: $I = \{0, n, 2n, 3n, ...\}$, т. е. обновление метода происходит через каждые n шагов.

Геометрический смысл метода сопряженных градиентов состоит в следующем (рис. 7.11). Из заданной начальной точки $x[0]$ осуществляется спуск в направлении $p[0] = -f'(x[0])$. В точке $x[1]$ определяется вектор-градиент $f'(x\ [1])$. Поскольку $x[1]$ является точкой минимума функции в направлении $p[0]$, то $f'(x[1])$ ортогонален вектору $p[0]$. Затем отыскивается вектор $p[1]$, H-сопряженный к p [0]. Далее отыскивается минимум функции вдоль направления $p[1]$ и т. д.

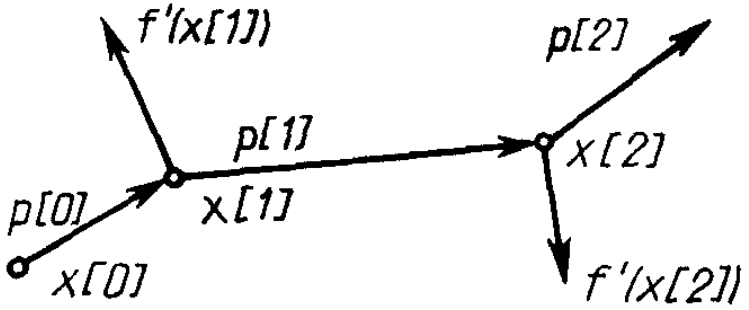


Рис. 7.11. Траектория спуска в методе сопряженных градиентов

Методы сопряженных направлений являются одними из наиболее эффективных для решения задач минимизации. Однако следует отметить, что они чувствительны к ошибкам, возникающим в процессе счета. При большом числе переменных погрешность может настолько возрасти, что процесс придется повторять даже для квадратичной функции, т. е. процесс для нее не всегда укладывается в n шагов.

7.1.3. Численные методы безусловной оптимизации второго порядка

7.1.3.1. Особенности методов второго порядка

Методы безусловной оптимизации второго порядка используют вторые частные производные минимизируемой функции $f(x)$. Суть этих методов состоит в следующем.

Необходимым условием экстремума функции многих переменных $f(x)$ в точке x^* является равенство нулю ее градиента в этой точке:

$$f'(x^*) = 0.$$

Разложение $f'(x)$ в окрестности точки $x[k]$ в ряд Тейлора с точностью до членов первого порядка позволяет переписать предыдущее уравнение в виде

$$f'(x) = f'(x[k]) + f''(x[k])\ (x - x[k]) = 0.$$

Здесь $f''(x[k]) = H(x[k])$ – матрица вторых производных (матрица Гессе) минимизируемой функции. Следовательно, итерационный процесс для построения последовательных приближений к решению задачи минимизации функции $f(x)$ описывается выражением:

$$x[k+1] = x[k] - H^{-1}(x[k])\, f'(x[k]),$$

где $H^{-1}(x[k])$ – обратная матрица для матрицы Гессе;
$H^{-1}(x[k])f'(x[k]) = p[k]$ – направление спуска.

Полученный метод минимизации называют *методом Ньютона.* Очевидно, что в данном методе величина шага вдоль направления $p[k]$ полагается равной единице. Последовательность точек $\{x[k]\}$, получаемая в результате применения итерационного процесса, при определенных предположениях сходится к некоторой стационарной точке x^* функции $f(x)$. Если матрица Гессе $H(x^*)$ положительно определена, точка x^* будет точкой строгого локального минимума функции $f(x)$. Последовательность $x[k]$ сходится к точке x^* только в том случае, когда матрица Гессе целевой функции положительно определена на каждой итерации.

Если функция $f(x)$ является квадратичной, то, независимо от начального приближения $x[0]$ и степени овражности, с помощью метода Ньютона ее минимум находится за один шаг. Это объясняется тем, что направление спуска $p[k] = H^{-1}(x[k])f'(x[k])$ в любых точках $x[0]$ всегда совпадает с направлением в точку минимума x^*. Если же функция $f(x)$ не квадратичная, но выпуклая, метод Ньютона гарантирует ее монотонное убывание от итерации к итерации. При минимизации овражных функций скорость сходимости метода Ньютона более высока по сравнению с градиентными методами. В таком случае вектор $p[k]$ не указывает направление в точку минимума функции $f(x)$, однако имеет большую составляющую вдоль оси оврага и значительно ближе к направлению на минимум, чем антиградиент.

Существенным недостатком метода Ньютона является зависимость сходимости для невыпуклых функций от начального приближения $x[0]$. Если $x[0]$ находится достаточно далеко от точки минимума, то метод может расходиться, т. е. при проведении итерации каждая следующая точка будет более удаленной от точки минимума, чем предыдущая. Сходимость метода, независимо от начального приближения, обеспечивается выбором не только направления спуска $p[k] = H^{-1}(x[k])f'(x[k])$, но и величины шага a вдоль этого направления. Соответствующий алгоритм называют *методом Ньютона с регулировкой шага.* Итерационный процесс в таком случае определяется выражением

$$x[k+1] = x[k] - a_k H^{-1}(x[k])f'(x[k]).$$

Величина шага a_k выбирается из условия минимума функции $f(x)$ по a в направлении движения, т. е. в результате решения задачи одномерной минимизации:

$$f\left(x[k] - a_k H^{-1}\left(x[k]\right) f'\left(x[k]\right)\right) = \min_{a \geq 0} f\left(x[k] - a_k H^{-1}\left(x[k]\right) f'\left(x[k]\right)\right).$$

Вследствие накопления ошибок в процессе счета матрица Гессе на некоторой итерации может оказаться отрицательно определенной или ее нельзя будет обратить. В таких случаях в подпрограммах оптимизации полагается $H^{-1}(x[k]) = E$, где E — единичная матрица. Очевидно, что итерация при этом осуществляется по методу наискорейшего спуска.

7.1.3.2. Метод Ньютона

Алгоритм метода Ньютона состоит в следующем.

1. В начальной точке $x[0]$ вычисляется вектор:

$$p[0] = -H^{-1}(x[0])f'([0]).$$

2. На k-той итерации определяются шаг a_k и точка $x[k+1]$.
3. Вычисляется величина $f(x[k+1])$.
4. Проверяются условия выхода из подпрограммы, реализующей данный алгоритм. Эти условия аналогичны условиям выхода из подпрограммы при методе наискорейшего спуска. Если эти условия выполняются, осуществляется прекращение вычислений. В противном случае вычисляется новое направление:

$$p[k+1] = -H^{-1}(x[k])f'([k]) .$$

и осуществляется переход к следующей итерации.

Количество вычислений на итерации методом Ньютона, как правило, значительно больше, чем в градиентных методах. Это объясняется необходимостью вычисления и обращения матрицы вторых производных целевой функции. Однако на получение решения с достаточно высокой степенью точности с помощью метода Ньютона обычно требуется намного меньше итераций, чем при использовании градиентных методов. В силу этого метод Ньютона существенно более эффективен. Он обладает сверхлинейной или квадратичной скоростью сходимости в зависимости от требований, которым удовлетворяет минимизируемая функция $f(x)$. Тем не менее в некоторых задачах трудоемкость итерации методом Ньютона может оказаться очень большой за счет необходимости вычисления матрицы вторых производных минимизируемой функции, что потребует затрат значительного количества машинного времени.

В ряде случаев целесообразно комбинированное использование градиентных методов и метода Ньютона. В начале процесса минимизации, когда точка $x[0]$ находится далеко от точки экстремума x^*, можно применять какой-либо вариант градиентных методов. Далее, при уменьшении скорости сходимости градиентного метода можно перейти к методу Ньютона.

7.2. Линейное программирование

Под *линейным программированием* понимается раздел теории экстремальных задач, в котором изучаются задачи минимизации (или максимизации) линейных функций на множествах, задаваемых системами линейных равенств и неравенств. В общем случае задача линейного программирования формулируется следующим образом. Найти вектор $x^* = (x^*_1, ..., x^*_n)$, определяющий максимум (минимум) линейной форме

$$f(x) = c_1x_1 + c_2x_2 + ... + c_nx_n \qquad (7.17)$$

при ограничениях (7.18):

$$\begin{gathered}
a_{11}x_1 + ... \; a_{1n}x_n \le b_1; \\
. \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \\
a_{m1}x_1 + ... \; a_{mn}x_n \le b_m; \\
a_{m+1}x_1 + ... \; a_{m+1,n}x_n \ge b_{m+1}; \\
. \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \\
a_kx_1 + ... \; a_{kn}x_n \ge b_m; \\
a_{k+1}x_1 + ... \; a_{k+1n}x_n \ge b_m; \qquad (7.18) \\
a_{m+1}x_1 + ... \; a_{m+1,n}x_n = b_{m+1}; \\
. \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \\
a_lx_1 + ... \; a_{ln}x_n = b_l; \\
x_i \ge 0; \; i = 1, \; ..., \; n.
\end{gathered}$$

Каждое из условий-неравенств определяет полупространство, ограниченное гиперплоскостью. Пересечение полупространств образует выпуклый n-мерный многогранник Q. Условия равенства выделяют из n-мерного пространства $(n–l)$-мерную плоскость, пересечение которой с областью Q дает выпуклый $(n–l)$-мерный многогранник G. Экстремальное значение линейной формы (если оно существует) достигается в некоторой вершине многогранника. При вырождении оно может достигаться во всех точках ребра или

грани многогранника. В силу изложенного для решения задачу линейного программирования теоретически достаточно вычислить значения функции в вершинах многогранника и найти среди этих значений наибольшее или наименьшее. Однако в практических задачах количество вершин области G настолько велико, что просмотр их даже с использованием ЭВМ невозможен. Поэтому разработаны специальные численные методы решения задач линейного программирования, которые ориентируются в основном на две формы записи задач. Каноническая форма задачи линейного программирования:

$$f(x) = c_1x_1 + ... + c_nx_n \to \max(\min); \qquad (7.19)$$

$$\begin{gathered}
a_{11}x_1 + ... \; a_{1n}x_n = b_1; \qquad (7.20) \\
. \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \\
a_mx_1 + ... \; a_{mn}x_n = b_m; \\
x_i \ge 0; \; i = 1, \; ..., \; n.
\end{gathered}$$

или в матричной форме:

$$(c, x) \to \max(\min); \qquad (7.21)$$

$$\begin{gathered}
Ax = b, \\
x \ge 0.
\end{gathered}$$

Здесь A = (a_{ij}) – (m на n) – матрица условий. Ее столбцы $(a_{1j}, ..., a_{mj})^T$, j = 1, ..., n, называются *векторами условий.* Вектор $b = (b_1, ..., b_m)^T$ носит название *вектора правых частей*, а $c = (c_1, \ldots, c_n)$ – *вектора коэффициентов линейной формы.*

Задача линейного программирования с однотипными условиями:

$$f(x) = c_1x_1 + ... + c_nx_n \to \max(\min); \qquad (7.22)$$

$$\begin{gathered}
a_{11}x_1 + \ldots \; a_{1n}x_n \le b_1; \\
. \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \; . \\
a_mx_1 + \ldots a_{mn}x_n \le b_m,
\end{gathered}$$

или в матричной форме:

$$(c, x) \to \max(\min); \qquad (7.23)$$

$$Ax \le b.$$

Любую задачу можно привести к каждой из приведенных форм с помощью приемов, описанных ниже.

Переход к эквивалентной системе неравенств

Знак неравенства можно поменять на обратный, меняя знаки свободного члена и коэффициентов. Например, ограничение:

$$a_{11}x_1 + \ldots a_{1n}x_n \le b_1;$$

можно заменить условием:

$$-a_{11}x_1 + \ldots - a_{1n}x_n \le -b_1.$$

Переход от ограничения-неравенства к равенству.

Для этого необходимо ввести дополнительную неотрицательную переменную. Так, условие:

$$a_1x_1 + \ldots a_nx_n \le b,$$

эквивалентно двум ограничениям:

$$-a_{11}x_1 + \ldots - a_{1n}x_n + x_{n+1} = b;\ x_{n+1} \ge b_1.$$

Представление ограничения-равенства парой неравенств.

Ограничение:

$$a_1x_1 + \ldots a_nx_n = b,$$

можно представить парой условий:

$$a_{11}x_1 + \ldots a_{1n}x_n \le b_1;$$
$$a_{11}x_1 + \ldots - a_{1n}x_n \le -b_1.$$

Переход к неотрицательным переменным

Если на знак переменной x_i не наложено ограничений, можно заменить ее разностью двух неотрицательных переменных:

$$x_i = x_{n+2} - x_{n+1},\ x_{n+1} \ge 0;\ x_{n+2} \ge 0.$$

Переход от переменных, ограниченных снизу, к неотрицательным переменным

Если переменная ограничена снизу $x_i \ge b_i$ то, заменив ее по формуле $x_i = y_i + b_i$ переходим к задаче с неотрицательной переменной $y_i > 0$.

Наиболее употребительным численным методом решения задач линейного программирования является *симплекс-метод.*

Идея этого метода состоит в следующем. Отыскивается некоторая вершина многогранника G и все ребра, выходящие из этой вершины. Далее перемещаются вдоль того из ребер, по которому функция убывает (при поиске минимума), и попадают в следующую вершину. Находят выходящие из нее ребра и повторяют процесс. Когда приходят в такую вершину, в которой вдоль всех выходящих из нее ребер функция возрастает, то минимум найден. Отметим, что, выбирая одно ребро, исключают из рассмотрения вершины, лежащие на остальных траекториях. В результате количество рассматриваемых вершин резко сокращается и оказывается посильным для ЭВМ. Симплекс-метод весьма эффективен и широко применяется для решения задач линейного программирования.

7.2.1. Транспортная задача линейного программирования

7.2.1.1. Постановка задачи

Транспортная задача является частным типом задачи линейного программирования и формулируется следующим образом. Имеется m пунктов отправления (или пунктов производства) A_i ..., A_m, в которых сосредоточены запасы однородных продуктов в количестве a_1, ..., a_m единиц. Имеется n пунктов назначения (или пунктов потребления) B_1, ..., B_m, потребность которых в указанных продуктах составляет b_1, ..., b_n единиц. Известны также транспортные расходы C_{ij}, связанные с перевозкой единицы продукта из пункта A_i в пункт B_j, $i = 1, \ldots, m$; $j = 1, \ldots, n$. Предположим, что

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j\ , \qquad (7.24)$$

т. е. общий объем производства равен общему объему потребления. Требуется составить такой план перевозок (откуда, куда и сколько единиц продукта везти), чтобы удовлетворить спрос всех пунктов потребления за счет реализации всего продукта, произведенного всеми пунктами производства, при минимальной общей стоимости всех перевозок. Приведенная формулировка транспортной задачи называется *замкнутой транспортной моделью.* Формализуем эту задачу.

Пусть x_{ij} – количество единиц продукта, поставляемого из пункта A_i в пункт B_j. Подлежащие минимизации суммарные затраты на перевозку продуктов из всех пунктов производства во все пункты потребления выражаются формулой:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} C_{ij}x_{ij}\,. \qquad (7.25)$$

Суммарное количество продукта, направляемого из каждого пункта отправления во все пункты назначения, должно быть равно запасу продукта в данном пункте. Формально это означает, что

$$\sum_{j=1}^{n} x_{ij} = a_i,\ \ i = 1,\ \ldots,\ m. \qquad (7.26)$$

Суммарное количество груза, доставляемого в каждый пункт назначения из всех пунктов отправления, должно быть равно потребности. Это условие полного удовлетворения спроса:

$$\sum_{j=1}^{n} x_{ij} = b_j,\ \ j = 1,\ \ldots,\ n. \qquad (7.27)$$

Объемы перевозок – неотрицательные числа, так как перевозки из пунктов потребления в пункты производства исключены:

$$x_{ij} \geq 0,\ i = 1,\ ...,\ m;\ j = 1,\ ...,\ n. \qquad (7.28)$$

Транспортная задача сводится к минимизации суммарных затрат при выполнении условий полного удовлетворения спроса и равенства вывозимого количества продукта запасам его в пунктах отправления.

В ряде случаев не требуется, чтобы весь произведенный продукт в каждом пункте производства был реализован. В таких случаях баланс производства и потребления может быть нарушен:

$$\sum_{j=1}^{n} x_{ij} \leq a_i,\ \ i = 1,\ \ldots,\ m. \qquad (7.29)$$

Введение этого условия приводит к открытой транспортной модели.

Задачи транспортного типа широко распространены в практике. Кроме того, к ним сводятся многие другие задачи линейного программирования – задачи о назначениях, сетевые, календарного планирования.

Как одна из задач линейного программирования транспортная задача принципиально может быть решена универсальным методом решения любой задачи линейного программирования, но этот метод не учитывает специфики условий транспортной задачи. Поэтому решение ее симплекс-методом оказывается слишком громоздким.

Структура ограничений задачи учитывается в ряде специальных вычислительных методов ее решения. Рассмотрим некоторые из них. Предварительно сделаем следующее замечание. Открытая транспортная модель может быть приведена к замкнутой модели добавлением фиктивного пункта отправления (потребления), от которого поступает весь недостающий продукт или в который свозится весь избыточный запас. Стоимость перевозок между реальными пунктами и фиктивным принимается равной нулю. Вследствие простоты перехода от открытой модели к замкнутой в дальнейшем рассматриваются методы решения замкнутой модели транспортной задачи.

7.2.1.2. Венгерский метод

Идея метода была высказана венгерским математиком Эгервари и состоит в следующем. Строится начальный план перевозок, не удовлетворяющий в общем случае всем условиям задачи (из некоторых пунктов производства не весь продукт вывозится, потребность части пунктов потребления не полностью удовлетворена). Далее осуществляется переход к новому плану, более близкому к оптимальному. Последовательное применение этого приема за конечное число итераций приводит к решению задачи.

Алгоритм венгерского метода состоит из подготовительного этапа и из конечного числа итераций. На подготовительном этапе строится матрица $X_0 = (x_{ij}[0])_{m,n}$, элементы которой неотрицательны и удовлетворяют неравенствам:

$$\sum_{j=1}^{n} x_{ij}[0] \le a_i,\ i = 1, \ldots, m. \qquad (7.30)$$

$$\sum_{i=1}^{m} x_{ij}[0] \le b_j,\ j = 1, \ldots, n. \qquad (7.31)$$

Если эти условия являются равенствами, то матрица X_0 – решение транспортной задачи. Если среди условий имеются неравенства, то осуществляется переход к первой итерации. На k-й итерации строится матрица $X_k = (x_{ij}[0])_{m,n}$. Близость этой матрицы к решению задачи характеризует число Δ_k — суммарная невязка матрицы X_k:

$$\Delta_k = \sum_{i=1}^{m} a_i + \sum_{j=1}^{n} b_j - 2\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}[k]. \qquad (7.32)$$

В результате первой итерации строится матрица X_1, состоящая из неотрицательных элементов. При этом $\Delta_1 < \Delta_0$. Если $\Delta_1 = 0$, то X_1 – оптимальное решение задачи. Если $\Delta_1 > 0$, то переходят к следующей итерации. Они проводятся до тех пор, пока Δ_k при некотором k не станет равным нулю. Соответствующая матрица X_k является решением транспортной задачи.

Венгерский метод наиболее эффективен при решении транспортных задач с целочисленными объемами производства и потребления. В этом случае число итераций не превышает величины $\Delta_0/2$ (Δ_0 – суммарная невязка подготовительного этапа).

Достоинством венгерского метода является возможность оценивать близость результата каждой из итераций к оптимальному плану перевозок. Это позволяет контролировать процесс вычислений и прекратить его при достижении определенных точностных показателей. Данное свойство существенно для задач большой размерности.

7.2.1.3. Метод потенциалов

Метод потенциалов является модификацией симплекс-метода решения задачи линейного программирования применительно к транспортной задаче. Он позволяет, отправляясь от некоторого допустимого решения, получить оптимальное решение за конечное

число итераций. Общая схема отдельной итерации такова. По допустимому решению каждому пункту задачи сопоставляется число, называемое его *предварительным потенциалом.* Пунктам A_i соответствуют числа u_i, пунктам B_j – числа v_j. Они выбираются таким образом, чтобы их разность на k-й итерации была равна C_{ij} – стоимости перевозки единицы продукции между пунктами A_i и B_j:

$$v_j[k] - u_i[k] = C_{ij},\ i = 1, \ldots, m;\ j = 1, \ldots, n. \qquad (7.33)$$

Если разность предварительных потенциалов для каждой пары пунктов A_i, B_j не превосходит C_{ij}, то полученный план перевозок является решением задачи. В противном случае указывается способ получения нового допустимого плана, связанного с меньшими транспортными издержками. За конечное число итераций находится оптимальный план задачи.

7.3. Прямые методы условной оптимизации

7.3.1. Основные определения

Задача условной оптимизации заключается в поиске минимального или максимального значения скалярной функции $f(x)$ n-мерного векторного аргументах (в дальнейшем без ограничения общности будут рассматриваться задачи поиска минимального значения функции):

$$f(x) \to \min, \qquad (7.34)$$

при ограничениях:

$$\begin{gathered} g_i(x) = 0,\ i = 1, \ldots, k; \\ h_j(x) \le 0,\ j = 1, \ldots, m; \\ a \le x \le b. \end{gathered} \qquad (7.35)$$

Здесь x, a, b – векторы-столбцы:

$$x = \begin{vmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{vmatrix}, \quad a = \begin{vmatrix} a_1 \\ a_2 \\ \ldots \\ a_n \end{vmatrix}, \quad b = \begin{vmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{vmatrix}. \qquad (7.36)$$

Оптимизируемую функцию $f(x)$ называют *целевой функцией.* Каждая точка x в n-мерном пространстве переменных $x_1, ..., x,$ в которой выполняются ограничения задачи, называется *допустимой точкой задачи.* Множество всех допустимых точек называется *допустимой областью* G . Будем считать, что это множество не пусто. *Решением задачи* считается допустимая точка x^*, в которой целевая функция $f(x)$ достигает своего минимального значения. Вектор x^* называют *оптимальным.* Если целевая функция $f(x)$ и ограничения задачи представляют собой линейные функции независимых переменных $x_1, ..., x_n,$ то соответствующая задача является *задачей линейного программировании,* в противном случае – *задачей нелинейного программирования.* В дальнейшем будем полагать, что функции $f(x), g(x), i = 1, ..., k , h_j(x), j = 1, ..., m,$ – непрерывные и дифференцируемые.

В общем случае численные методы решения задач нелинейного программирования можно разделить на прямые и непрямые. *Прямые методы* оперируют непосредственно с исходными задачами оптимизации и генерируют последовательности точек $\{x[k]\}$, таких что $f(x[k + 1])<(x[k])$. В силу этого такие методы часто называют *методами спуска.* Математически переход на некотором k-ом шаге $(k = 0, 1, 2, ...)$ от точки $x[k]$ к точке $x[k + 1]$ можно записать в следующем виде:

$$x[k + 1] = x[k] + a_k p[k] , \qquad (7.37)$$

где $p[k]$ – вектор, определяющий направление спуска; a_k – длина шага вдоль данного направления. При этом в одних алгоритмах прямых методов точки $x[k]$ выбираются так, чтобы для них выполнялись все ограничения задачи, в других эти ограничения могут нарушаться на некоторых или всех итерациях. Таким образом, в прямых методах при выборе направления спуска ограничения, определяющий допустимую область $G,$ учитываются в явном виде.

Непрямые методы сводят исходную задачу нелинейного программирования к последовательности задач безусловной оптимизации некоторых вспомогательных функций. При этих методах ограничения исходной задачи учитываются в неявном виде.

Рассмотрим некоторые алгоритмы прямых методов.

7.3.2. Метод проекции градиента

Рассмотрим данный метод применительно к задаче оптимизации с ограничениями-неравенствами. В качестве начальной выбирается некоторая точка допустимой области $G.$ Если $x[0]$ – внутренняя точка множества G (рис. 7.11), то рассматриваемый метод является обычным градиентным методом:

$$x[k + 1] = x[k] – a_k f'(x[k]), k = 0, 1, 2, ..., \qquad (7.38)$$

где $f'(x[k]) = \left(\frac{\partial f(x[k])}{\partial x_1}, \ldots, \frac{\partial f(x[k])}{\partial x_n} \right)^T$ (7.39) – градиент целевой функции $f(x)$ в точке $x[k]$.

После выхода на границу области G в некоторой граничной точке $x[k] , k = 0, 1, 2,...,$ движение в направлении антиградиента – $f'(x[k])$ может вывести за пределы допустимого множества (см. рис. 7.11). Поэтому антиградиент проецируется на линейное многообразие $M,$ аппроксимирующее участок границы в окрестности точки $x[k]$. Двигаясь в направлении проекции вектора $–f'(x[k])$ на многообразие $M,$ отыскивают новую точку $x[k + 1]$, в которой $f(x[k + 1]) < f(x[k])$, принимают $x[k + 1]$ за исходное приближение и продолжают процесс. Проведем более подробный анализ данной процедуры.

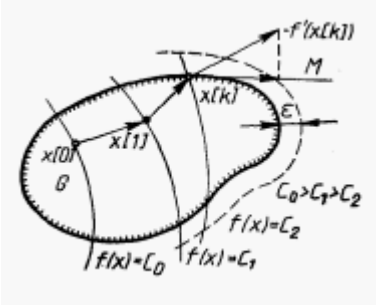


Рис. 7.12. Геометрическая интерпретация метода проекции градиента

В точке $x[k]$ часть ограничений-неравенств удовлетворяется как равенство:

$$h_i(x) = 0, j = 1, ..., l;\ l<m.$$

Такие ограничения называют *активными.*

Обозначим через J набор индексов $j(1 \le j \le l)$ этих ограничений. Их уравнения соответствуют гиперповерхностям, образующим границу области G в окрестности точки $x[k]$. В общем случае эта граница является нелинейной (см. рис. 3.1). Ограничения $h_j(x)$, $j \in J$, аппроксимируются гиперплоскостями, касательными к ним в точке $x[k]$:

$$\sum_{i=1}^{N} \frac{\partial h_i\left(x\left[k\right]\right)}{\partial x}\left(x_i - x_i\left[k\right]\right) = 0, \quad j \in J. \tag{7.39}$$

Полученные гиперплоскости ограничивают некоторый многогранник M, аппроксимирующий допустимую область G в окрестности точки $x[k]$ (см. рис. 7.11).

Проекция $p[k]$ антиградиента $-f'(x[k])$ на многогранник вычисляется по формуле:

$$p[k] = \mathrm{P}[-f'(x[k])]. \tag{7.40}$$

Здесь P – оператор ортогонального проектирования, определяемый выражением:

$$P = E - A^T(AA^T)^{-1}A, \tag{7.41}$$

где E – единичная матрица размеров n; A – матрица размеров lxn . Она образуется транспонированными векторами-градиентами a_j, $j = 1, ..., l$, активных ограничений. Далее осуществляется спуск в выбранном направлении:

$$x[k+1] = x[k] + a_k p[k]. \tag{7.42}$$

Можно показать, что точка $x[k+1]$ является решением задачи минимизации функции $f(x)$ в области G тогда и только тогда, когда:

$$\mathrm{P}[-f'(x[k])] = 0, \tag{7.43}$$

т. е.
$$-f'\left(x\left[k\right]\right) = \sum_{j=1}^{l} u_j a_j \tag{7.44}$$

и
$$u = (u_1, ..., u_l) = (A^TA)^{-1}A^T(-f'(x[k])) > 0. \tag{7.45}$$

Эти условия означают, что антиградиент ($-f'(x[k])$) целевой функции является линейной комбинацией с неотрицательными коэффициентами градиентов ограничений $h_j(x) = 0$.

В соответствии с изложенным, алгоритм метода проекции градиента состоит из следующих операций.

1. В точке $x[k]$ определяется направление спуска $p[k]$.
2. Находится величина шага a_k.
3. Определяется новое приближение $x[k+1]$.

Рассмотрим детально каждую из этих операций.

1. Определение направления спуска состоит в следующем. Пусть найдена некоторая точка $x[k] \in G$ и известен набор активных ограничений $h_i(x[k]) = 0$, $j \in J$. На основании данной информации вычисляют ($-f'(x[k])$) и определяют проекцию $P[-f'(x[k])]$. При этом возможны два случая:

– $P[-f'(x[k])]$ не равна 0. В качестве направления спуска $p[k]$ принимают полученную проекцию;

– $P[-f'(x[k])] = 0$, т. е.

$$-f'\left(x\left[k\right]\right) = \sum_{j=1}^{l} u_j a_j . \tag{7.46}$$

Данное выражение представляет собой систему из n уравнений для определения коэффициентов u_j. Если все $u_j \ge 0, j \in$ J, то в соответствии с вышеизложенным точка $x[k]$ является решением задачи. Если же некоторый компонент $u_q < 0$, то соответствующий ему градиент выводится из матрицы A и порождается новая проецирующая матрица P. Она определит новое направление спуска.

Для определения величины шага a_k целевая функция минимизируется по направлению $p[k]$ при условии соблюдения ограничений задачи с установленной точностью. Последняя задается введением некоторого положительного числа ε. Считают, что точка x удовлетворяет условиям задачи с заданной точностью, если $h_i(x) \le \varepsilon$, $j = 1, ..., m$. Величина шага a_k определяется решением задачи вида:

$$f(x[k] + ap[k]) > \min; \tag{7.47}$$

$$h_j(x[k] + ap[k]) \le \varepsilon, j = 1, ..., m. \tag{7.48}$$

Определение нового приближения состоит в следующем. Очередная точка вычисляется по формуле:

$$x[k+1] = x[k] + a_k p[k]. \qquad (7.49)$$

Признаком сходимости является стремление к нулю векторов $p[k]$. Рассмотренный метод является в некотором смысле аналогом градиентных методов для решения задач на безусловный экстремум, и ему свойствен их недостаток – медленная сходимость.

7.3.3. Комплексный метод Бокса

Этот метод представляет модификацию метода деформируемого многогранника и предназначен для решения задачи нелинейного программирования с ограничениями-неравенствами. Для минимизации функции n переменных $f(x)$ в n-мерном пространстве строят многогранники, содержащие $q>n+1$ вершин. Эти многогранники называют *комплексами,* что и определило наименование метода.

Введем следующие обозначения:

$$x[j, k] = (x_1[j, k], \ldots, x_i[j, k], \ldots, x_n[j, k])^T, \qquad (7.50)$$

где $j = 1, ..., q$; $k = 0, 1, 2, ...$ – j-ая вершина комплекса на k-ом этапе поиска;

$x[h, k]$ – вершина, в которой значение целевой функции максимально, т. е. $f(x[h, k]) = \max\{f(x[1, k]), ..., f(x[q, k])\}$;

$x[h, k]$ – центр тяжести всех вершин, за исключением $x[h, k]$.

Координаты центра тяжести вычисляются по формуле:

$$x_i[l,k] = \frac{1}{q}\left(\sum_{j=1}^{q} x_i[j,k] - x_i[h,k]\right), \; i = 1, ..., n. \qquad (7.51)$$

Алгоритм комплексного поиска состоит в следующем. В качестве первой вершины начального комплекса выбирается некоторая допустимая точка $x[1, 0]$. Координаты остальных $q - 1$ вершин комплекса определяются соотношением

$$x_j[j, 0] = a_i + r_i(b_i - a_i), \; i = 1, ..., n; \; j = 2, ..., q. \qquad (7.52)$$

Здесь a_i, b_i – соответственно нижнее и верхнее ограничения на переменную x_i', r_i – псевдослучайные числа, равномерно распределенные на интервале [0, 1]. Полученные таким образом точки удовлетворяют ограничениям $a \le x \le b$, однако ограничения $h_j(x) \le 0$ могут быть нарушены. В этом случае недопустимая точка заменяется новой, лежащей в середине отрезка, соединяющего недопустимую точку с центром тяжести выбранных допустимых вершин. Данная операция повторяется до тех пор, пока не будут выполнены все ограничения задачи. Далее, как и в методе деформируемого многогранника, на каждой итерации заменяется вершина $x[h, k]$, в которой значение целевой функции имеет наибольшую величину. Для этого $x[h, k]$ отражается относительно центра тяжести $x[l, k]$ остальных вершин комплекса. Точка $x[p, k]$, заменяющая вершину $x[h, k]$, определяется по формуле:

$$x[p, k] = (a + 1)x[l, k] + ax[h, k], \qquad (7.53)$$

где $a>0$ – некоторая константа, называемая *коэффициентом отражения.*

Наиболее удовлетворительные результаты дает значение $a = 1{,}3$. При этом новые вершины комплекса отыскиваются за небольшое количество шагов, а значения целевой функции уменьшаются достаточно быстро.

Если $f(x[p, k])>f(x[h, k])$, то новая вершина оказывается худшей вершиной комплекса, В этом случае коэффициент a уменьшается в два раза. Если в результате отражения нарушается какое-либо из ограничений, то соответствующая переменная просто возвращается внутрь нарушенного ограничения. Если при отражении нарушаются ограничения $h_j(x)<0$. то коэффициент a каждый раз уменьшается вдвое до тех пор, пока точка $x[p, k]$ не станет допустимой. Вычисления заканчиваются, если значения целевой функции мало меняются в течение пяти последовательных итераций: $|f(x[l, k+1]) - f(x[l, k])| \le \varepsilon$, $k = 1, ..., 5$, где ε – заданная константа. В этом случае центр тяжести комплекса считают решением задачи нелинейного программирования.

Достоинствами комплексного метода Бокса являются его простота, удобство для программирования, надежность в работе. Метод на каждом шаге использует информацию только о значениях целевой функции и функций ограничений задачи. Все это обусловливает успешное применение его для решения различных задач нелинейного программирования.

Выбор начальной точки допустимой области

Для применения прямых методов решения задач нелинейного программирования требуется знание некоторой допустимой началь-

ной точки области G. Если структура этой области сложная, отыскание такой точки представляет серьезные трудности. Произвольно выбранная начальная точка в общем случае может удовлетворять только части ограничений. Следовательно, необходим алгоритм, приводящий из произвольной точки в допустимую область. На практике для получения начального вектора применяют тот же метод, которым решают исходную задачу нелинейного программирования. Рассмотрим один из способов отыскания такого вектора.

Пусть $\bar{x}$ – произвольная точка, в которой часть ограничений не удовлетворяется. Обозначим через J_1 множество индексов ограничений, выполняющихся в точке $\bar{x}$, и через J_2 – множество индексов ограничений, не выполняющихся в ней, т.е.

$$J_1=\left\{j|hj\left(\bar{x}\right)\le 0 \quad j=1,...,m\right\}; \tag{7.54}$$

$$J_2=\left\{j|hj\left(\bar{x}\right)>0 \quad j=1,...,m\right\}. \tag{7.55}$$

Введем дополнительную переменную ζ и сформулируем задачу поиска допустимой точки следующим образом: найти минимум функции $z = \zeta$ при ограничениях:

$$\begin{aligned} hj\left(\bar{x}\right)&\le 0 \quad j\in j_1;\\ hj\left(\bar{x}\right)-\xi&\le 0 \quad j\in j_1. \end{aligned} \tag{7.57}$$

Допустимый вектор этой задачи находится довольно просто. Действительно, если положить:

$$\bar{\xi}=\max_{j\in J_2}\left(-h_2\left(\bar{x}\right)\right), \tag{7.58}$$

то точка $\left(\bar{\xi},\bar{x}\right)$ является допустимым решением сформулированной задачи. Так как область G исходной задачи не пуста, то существует такая точка $\bar{x}$, что:

$$h(x)<0, j = 1, \ldots, m. \tag{7.59}$$

Следовательно, минимальное значение ζ меньше нуля. В силу этого после конечного числа шагов некоторого прямого алгоритма будут получены $x[0]$, ζ, такие, что $\zeta<0$, и условия задачи удовлетворяются. Точка $x[0]$ и принимается в качестве начальной для исходной задачи нелинейного программирования.

7.4. Методы штрафных функций

7.4.1. Основные определения

Методы штрафных функций относятся к группе непрямых методов решения задач нелинейного программирования:

$$f(x) \to \min; \tag{7.60}$$

$$\begin{aligned} g_i(x) &= 0,\ i = 1, \ldots, k;\\ h_j(x) &< 0,\ j = 1, \ldots, m;\\ a \le x &\le b. \end{aligned} \tag{7.61}$$

Они преобразуют задачу с ограничениями в последовательность задач безусловной оптимизации некоторых вспомогательных функций. Последние получаются путем модификации целевой функции с помощью функций-ограничений таким образом, чтобы ограничения в явном виде в задаче оптимизации не фигурировали. Это обеспечивает возможность применения методов безусловной оптимизации. В общем случае вспомогательная функция имеет вид

$$F(x,a) = f(x) + \Phi(x, a). \tag{7.62}$$

Здесь $f(x)$ – целевая функция задачи оптимизации; $\Phi(x, a)$ – штрафная функция; параметр $a > 0$. Точку безусловного минимума функции $F(x, a)$ будем обозначать через $x(a)$.

7.4.2. Методы внутренних штрафных функций

В зависимости от вида $\Phi(x, a)$ различают методы внутренних штрафных, или барьерных, функций и методы внешних штрафных функций.

Эти методы применяются для решения задач нелинейного программирования с ограничениями-неравенствами. В рассматриваемых методах функции $\Phi(x, a)$ подбирают такими, чтобы их значения неограниченно возрастали при приближении к границе допустимой

области G (рис. 7.12). Иными словами, приближение к границе «штрафуется» резким увеличением значения функции $F(x, a)$. На границе G построен «барьер», препятствующий нарушению ограничения в процессе безусловной минимизации $F(x, a)$. Поиск минимума вспомогательной функции $F(x, a)$ необходимо начинать с внутренней точки области G . При этом в процессе оптимизации траектория спуска никогда не выйдет за пределы допустимой области. Все перечисленные особенности функции $\Phi(x, a)$ определили наименование рассматриваемой группы методов.

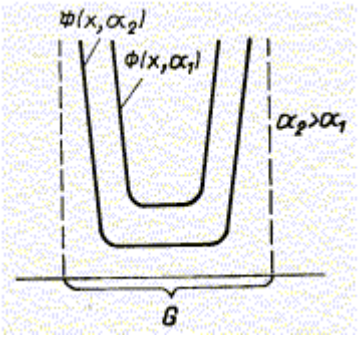


Рис. 7.13. Внутренняя штрафная функция

Таким образом, внутренняя штрафная функция $\Phi(x, a)$ может быть определена следующим образом:

$$\Phi(x,a)=\begin{cases}\infty, & \textit{если} \quad x\notin G;\\ 0, & \textit{если} \quad x\in G, \quad x\notin dG; \quad a\to 0,\\ \infty, & \textit{если} \quad x\in G, \quad x\to dG.\end{cases} \qquad (7.63)$$

Здесь dG –граница области G.

Общий вид внутренней штрафной функции:

$$\Phi(x,a)=a\left(\sum_{j=1}^{m}\varphi_j\left(h_j(x)\right)\right), \qquad (7.64)$$

где φ_j – непрерывные дифференцируемые функции, определяемые ограничениями-неравенствами исходной задачи нелинейного программирования. Вспомогательная функция $F(x, a)$ при этом имеет форму:

$$F(x,a)=f(x)+a\left(\sum_{j=1}^{m}\varphi_j\left(h_j(x)\right)\right). \qquad (7.65)$$

Она определена в области G и неограниченно возрастает, если $h_j(x) > 0$ для некоторого j. В качестве внутренних штрафных функций используют, например, такие:

$$\Phi(x,a)=a\sum_{j=1}^{m}\frac{1}{h_j(x)};\ \Phi(x,a)=-a\sum_{j=1}^{m}\ln h_j(x). \qquad .(7.66)$$

Алгоритм метода внутренних штрафных функций состоит в следующем. В качестве начальной точки $x[0]$ выбирается произвольная внутренняя точка области G. Задается некоторая монотонно убывающая сходящаяся к нулю последовательность положительных чисел $\{a_k\}$, $k = 1, 2, ...$ Для первого элемента a_1 этой последовательности решается задача безусловной минимизации функции $F(x, a)$, в результате чего определяется точка $x(a_1)$. Эта точка используется в качестве начальной для решения задачи поиска минимума функции $F(x, a_2)$, где $a_2<a_1$, и т. д. Таким образом, решается последовательность задач безусловной минимизации функций $F(x, a_k)$, $k = 1, 2, ...$, причем решение предыдущей задачи $x(a_k)$ используется в качестве начальной точки для поиска последующего вектора $x(a_{k+1})$. Последовательность полученных таким способом точек $x(a_k)$ сходится к оптимальному решению исходной задачи – локальному минимуму x^*. Вычисления прекращают при выполнении условий:

$$|f(x[k]) - f(x[k-1])|\le \varepsilon; \qquad (7.67)$$
$$\|x[k] - x[k-1]\| \le \beta, \qquad (7.68)$$

где ε, β – заданные числа, определяющие точность вычислений.

Можно показать, что рассмотренный метод внутренних штрафных функций обладает следующими свойствами:

1) $\lim\limits_{k\to\infty} a_k\sum\limits_{j=1}^{m}\varphi_j\left(h_i\left(x[k]\right)\right)=0;$

2) $\lim\limits_{k\to\infty} f\left(x[k]\right)=f\left(x^*\right)$ и $f\left(x[k]\right)$ монотонно убывает;

3) $\lim\limits_{k\to\infty} F\left(x[k]a_k\right) = f\left(x^*\right)$.

Эти свойства справедливы для задач, содержащих непрерывные функции и имеющих локальные минимумы внутри области *G*.

7.4.3. Методы внешних штрафных функций

Данные методы применяются для решения задачи оптимизации в общей постановке, т. е. при наличии как ограничений-неравенств, так и ограничений-равенств. В рассматриваемых методах функции Ф(*x, a*) выбирают такими, что их значения равны нулю внутри и на границе допустимой области *G,* а вне ее – положительны и возрастают тем больше, чем сильнее нарушаются ограничения (рис. 7.13). Таким образом, здесь «штрафуется» удаление от допустимой области *G*.

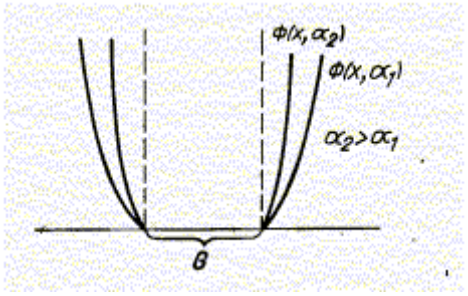


Рис. 7.14. Внешняя штрафная функция

Внешняя штрафная функция Ф(*x, a*) в общем случае может быть определена следующим образом:

$$\Phi\left(x,a\right) = \begin{cases} 0, & \text{если} \quad x \in G; \\ \infty, & \text{если} \quad x \in G, \quad a \to \infty. \end{cases} \tag{7.69}$$

Поиск минимума вспомогательной функции *F*(*x, a*) можно начинать из произвольной точки. В большинстве случаев она является недопустимой, поэтому траектория спуска располагается частично вне допустимой области. Если минимум целевой функции расположен на границе допустимой области, то эта траектория полностью находится снаружи области *G*. Перечисленные

особенности функции Ф(*x, a*) определили название данной группы методов. Общий вид внешней штрафной функции:

$$\Phi\left(x,a\right) = a\left(\sum_{i=1}^{k}\varphi_i\left(g_i\left(x\right)\right) + \sum_{j=1}^{m}\varphi_j\left(h_j\left(x\right)\right)\right), \tag{7.70}$$

где φ_j, ϕ_j – функции, определяемые соответственно ограничениями-равенствами и неравенствами исходной задачи нелинейного программирования. Вспомогательная функция *F*(*x, a*) при этом имеет форму:

$$F\left(x,a\right) = f\left(a\right) + a\left(\sum_{i=1}^{k}\varphi_i\left(g_i\left(x\right)\right) + \sum_{j=1}^{m}\varphi_j\left(h_j\left(x\right)\right)\right). \tag{7.71}$$

Одна из применяемых внешних штрафных функций имеет вид:

$$\Phi\left(x,a\right) = a\left(\sum_{i=1}^{k}\left(\max\left(0,g_i\left(x\right)\right)\right)^2 + \sum_{i=1}^{k}\varphi_j\left(h_j\left(x\right)\right)^2\right), \tag{7.72}$$

здесь $$\max\left(0,g_i\left(x\right)\right) = \begin{cases} 0, & \text{если} \quad g_i\left(x\right) \le 0; \\ g_j\left(x\right), & \text{если} \quad g_i\left(x\right) > 0. \end{cases} \tag{7.73}$$

Алгоритм метода внешних штрафных функций формулируется так же, как и алгоритм метода внутренних штрафных функций, и обладает аналогичными свойствами. Однако в этом случае не требуется, чтобы начальная точка $x[0] \in G$, а последовательность $\{a_k\}$, $k = 1, 2, \ldots$, положительных чисел должна быть монотонно возрастающей.

Анализ методов штрафных функций позволяет сделать следующие выводы об их вычислительных свойствах. В соответствии с методами внутренних штрафных функций ведут поиск решения, не выходя за пределы допустимой области. Это весьма важно в тех случаях, когда целевая функция или ограничения не определены за пределами допустимого множества. Кроме того, прервав вычисления в любой момент времени, мы всегда получим допустимое решение. Однако для задания в качестве начальной некоторой допустимой точки иногда требуется решать задачу, по сложности

сравнимую с исходной задачей нелинейного программирования. В этом смысле метод внешних штрафных функций предпочтительнее, так как он обеспечивает решение из любой начальной точки. В результате программирование для ЭВМ алгоритмов внешних штрафных функций существенно упрощается. Общим недостатком методов штрафных функций является сложность вспомогательной функции $F(x, a)$, которая часто имеет овражную структуру. Степень овражности увеличивается с увеличением a. Кроме того, при больших значениях a точность вычислений минимума $F(x, a)$ сильно уменьшается из-за ошибок округления.

7.4.4. Комбинированные алгоритмы штрафных функций

Для задач нелинейного программирования с ограничениями-равенствами методы внутренних штрафных функций неприменимы. Однако при практических расчетах в ряде случаев необходимо выполнение некоторых ограничений-неравенств в течение всего процесса оптимизации. В подобных обстоятельствах используют *комбинированные алгоритмы,* учитывающие особенности внутренних и внешних штрафных функций. Вспомогательная функция $F(x, a)$ включает в себя слагаемые в виде внутренних штрафных функций $\varphi(x, a)$ для ограничений-неравенств и внешних штрафных функций $\phi(x, a)$ для ограничений-равенств:

$$F(x, a) = f(x) + \Phi(x, a) + \phi(x, a). \qquad (7.74)$$

В таком случае неравенства будут выполнены на протяжении всего вычислительного процесса, а равенства – при приближении к минимуму. Например, в качестве внутренней можно использовать логарифмическую штрафную функцию, а в качестве внешней – квадратичную функцию, т. е.

$$F(x, a_k) = f(x) + a_k \sum_{j=1}^{m} \ln\left(-h(x)\right) + \frac{1}{a_k} \sum_{i=1}^{k} g^2(x). \qquad (7.75)$$

В качестве начальной точки выбирается вектор $x[0]$, удовлетворяющий условиям $h_j(x) > 0$, $j = 1, ..., m$. Последовательность $\{a_k\}$, $k = 1, 2, ...$, положительных чисел является монотонно убывающей и сходящейся к нулю.

Выбор начальной точки допустимой области

В данном случае задача состоит в поиске точки, удовлетворяющей строгим неравенствам $h_i(x) > 0$, $j = 1, ..., m$. Рассмотрим два множества индексов:

$$J_1 = \left\{ j \mid hj\left(\bar{x}\right) < 0 \quad j = 1, ..., m \right\} \qquad (7.76)$$

$$J_1 = \left\{ j \mid hj\left(\bar{x}\right) \le 0 \quad j = 1, ..., m \right\} \qquad (7.77)$$

Поиск внутренней точки должен состоять в том, чтобы перейти от точки $\left(\bar{x}\right)$ к новой точке $\left(\tilde{x}\right)$, в которой удовлетворяются некоторые из ограничений множества J_2. При этом ни одно из условий множества J_1 не должно нарушаться. Затем необходимо перейти от точки $\left(\tilde{x}\right)$ к другой точке, и так далее до тех пор, пока не будут удовлетворены все ограничения. Эта процедура реализуется минимизацией функции

$$W\left(x, a_k\right) = \sum_{j \in J_2} h_j\left(x\right) + a_k V\left(x\right), \qquad (7.78)$$

где $V(x)$ – внутренняя штрафная функция одного из видов, представленных выше, относительно ограничений из множества J_1.

Последовательность $\{ak\}$, $k = 1, 2, ...$, сходится к нулю. В процессе минимизации производится проверка знаков функций hj, $j \in J2$. Как только какое-либо из этих ограничений удовлетворяется, оно переводится во второе слагаемое, т. е. формируется соответствующая штрафная функция $V(x)$. Минимизация полученной в результате новой функции $W(x, a)$ осуществляется из последней найденной точки $x[k]$.

Для решения задач оптимизации многостадийных процессов, а также для процессов, которые могут быть математически описаны как многостадийные, применяется *метод динамического программирования.*

Динамическое программирование применяется для оптимизации *математически описанных процессов.* Поэтому в дальнейшем для многостадийного процесса предполагается известным математическое описание его каждой стадии, которое представляется в общем виде системой уравнений:

$$x_k^{(i)} = \varphi_k^{(i)}(x_1^{(i-1)}, \ldots, x_m^{(i-1)}, u_1^{(i)}, \ldots, u_r^{(i)}) ; \qquad (7.79)$$

$$k = 1, \ldots, m;\ i = 1, \ldots, N,$$

связывающей выходные параметры i-й стадии $x_k^{(i)}$ с выходными параметрами предыдущей стадии $x_k^{(i-1)}$ и управлением $u_l^{(i)}$ ($l = 1, \ldots, r$), используемым на i-й стадии.

Систему уравнений удобно также представить в векторной форме:

$$x^{(i)} = \varphi^{(i)}(x^{(i-1)}, u^{(i)}); \qquad (7.80)$$

Причем $x^{(i)}$ – вектор совокупности переменных состояния (или выход) i-й стадии;

$$x^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_m^{(i)}), \qquad (7.81)$$

где А $u^{(i)}$ – вектор совокупности управляющих воздействий (управление) на i-й стадии:

$$u^{(i)} = (u_1^{(i)}, u_2^{(i)}, \ldots, u_r^{(i)}). \qquad (7.82)$$

Размерности векторов состояния $x^{(i)}$ и управления $u^{(i)}$ в общем случае могут быть различными для разных стадий процесса. Однако далее, не нарушая общности, можно принять, что размерности m и r векторов состояния и управления для всех стадий процесса одинаковы.

В реальных процессах на значения переменных состояния $x^{(i)}$ и управляющих воздействий $u^{(i)}$ могут быть наложены ограничения, определяющие диапазон изменения или взаимосвязь указанных переменных. Математически это находит выражение в появлении дополнительных условий в виде равенств или неравенств

$$F_j(x^{(1)}, \ldots, x^{(N)}, u^{(1)}, \ldots, u^{(N)}), \qquad (7.83)$$

которые должны учитываться при решении задачи оптимизации.

В дальнейшем при необходимости выразить, что значения переменных состояния или управляющих воздействий удовлетворяют ограничениям, будем использоваться запись:

$$x^{(i)} \in X, \quad u^{(i)} \in U, \qquad (7.84)$$

Смысл записи заключается в том, что значения переменных $x^{(i)}$ и $u^{(i)}$ принадлежат к допустимым областям их изменения X и U, ограниченным соответствующими соотношениями.

Предполагается, что эффективность каждой стадии процесса оценивается некоторой скалярной величиной

$$r_i = r_i^*(x^{(i)}, u^{(i)}), \qquad (7.85)$$

заданной в виде функции от переменных состояния стадии $x^{(i)}$ и принятого на ней управления $u^{(i)}$.

С учетом математического описания стадии функциональная зависимость эффективности может быть представлена также как

$$r_i = r_i(x^{(i-1)}, u^{(i)}), \qquad (7.86)$$

т. е. как функция состояния входа $x^{(i-1)}$ на i-й стадии и используемого на ней управления $u^{(i)}$

Результирующая оценка эффективности многостадийного процесса в целом определяется как аддитивная функция результатов, получаемых на каждой стадии:

$$R_N = \sum_{i=1}^{N} r_i\left(x^{(i-1)}, u^{(i)}\right). \qquad (7.87)$$

Естественно, что значение критерия оптимальности R_N зависит от совокупности $u_N^{(i)}$ управляющих воздействий на всех стадиях процесса, которые представляет собой набор значений векторов $u^{(i)}$ для всех стадий:

$$u_N = (u^{(1)}, u^{(2)}, \ldots, u^{(N)}). \qquad (7.88)$$

Совокупность управлений для всех стадий процесса u_N будем называть в дальнейшем *стратегией управления многостадийным процессом* или просто *стратегией управления.*

Таким образом, задачу оптимизации многостадийного процесса можно сформулировать как задачу отыскания оптимальной стратегии

$$\tilde{u}_{Nопт} = (u_{\text{ОПТ}}{}^{(1)}, u_{\text{ОПТ}}{}^{(2)}, \ldots, u_{\text{ОПТ}}{}^{(N)}), \qquad (7.89)$$

для которой критерий оптимальности r_n принимает в зависимости от постановки оптимальной задачи максимальное или минимальное значение.

Принцип оптимальности

В основу метода динамического программирования положен принцип оптимальности, который в переложении для многостадийного процесса может быть сформулирован следующим образом. *Оптимальная стратегия обладает тем свойством, что каковы бы ни были начальное состояние $x^{(0)}$ многостадийного процесса и управление на первой стадии $u^{(1)}$, последующие управления на всех стадиях $u^{(i)}$ ($i = 2, \ldots, N$) должны составлять оптимальную стратегию u_{N-1} относительно состояния $x^{(1)}$ первой стадии, определяемого начальным состоянием процесса $x^{(0)}$ и управлением на первой стадии $u^{(1)}$.*

В приведенной формулировке принципа оптимальности под оптимальной стратегией u_{N-1} понимается стратегия управления многостадийным процессом, включающим N–1 последних стадий исходного процесса, придающая критерию оптимальное значение.

$$R_{N-1} = \sum_{i=1}^{N} r_i\left(x^{(i-1)}, u^{(i)}\right). \qquad (7.90)$$

Другими словами, оптимальная стратегия u_{N-1} находится для $(N-1)$-стадийного процесса, для которого величина является начальным состоянием.

Таким образом, если известна оптимальная стратегия управления u_{N-1} для любого возможного состояния $x^{(1)}$ первой стадии $N-1$ стадийного процесса, то уже не составляет труда выбрать оптимальное управление и на первой стадии $u_{\text{опт}}{}^{(1)}$, поскольку на последующих стадиях оно определяется только состоянием выхода первой стадии:

$$u_{N-1} = u_{N-1}(x^{(1)}). \qquad (7.91)$$

Процедура применения принципа оптимальности для оптимизации N-стадийного процесса, очевидно, должна начинаться с последней стадии процесса, для которой не существует последующих стадий, которые могут повлиять согласно принципу оптимальности на выбор управления $u_{\text{опт}}{}^{(N)}$ на этой стадии. После того как оптимальное управление $u_{\text{опт}}{}^{(N)}$ найдено для всех возможных состояний входа последней стадии $x^{(N-1)} \in X$, можно приступить к определению оптимального управления для предыдущей $(N-1)$-стадии, для которой оптимальная стратегия управления на последующих стадиях (т. е. на последней N-й стадии) известна, и т. д.

В результате может быть найдена оптимальная стратегия управления для всего многостадийного процесса, являющаяся функцией начального состояния процесса $u_{\text{N}}{}^{(x(0))}$. Если начальное состояние $x^{(0)}$ известно (задано или выбрано из условия оптимума критерия R), то его значение определяет оптимальные управления для всех стадий процесса.

7.5. Информационные технологии поддержки принятия решений

Главной особенностью информационной технологии поддержки принятия решений является качественно новый метод организации взаимодействия человека и компьютера. Выработка решения, что является основной целью этой технологии, происходит в результате итерационного процесса, в котором участвуют:

– система поддержки принятия решений в роли вычислительного звена и объекта управления;

– человек как управляющее звено, задающее входные данные и оценивающее полученный результат вычислений на компьютере.

Окончание итерационного процесса происходит по воле человека. В этом случае можно говорить о способности информационной системы совместно с пользователем создавать новую информацию для принятия решений.

Дополнительно к этой особенности информационной технологии поддержки принятия решений можно указать еще ряд ее отличительных характеристик:

– ориентация на решение плохо структурированных задач;

– сочетание традиционных методов доступа и обработки компьютерных данных с возможностями математических моделей и методами решения задач на их основе;

– направленность на непрофессионального пользователя компьютера;

– высокая адаптивность, обеспечивающая возможность приспосабливаться к особенностям имеющегося технического и программного обеспечения, а также требованиям пользователя.

Информационная технология поддержки принятия решений может использоваться на любом уровне управления. Кроме того, решения, принимаемые на различных уровнях управления, часто должны координироваться. Поэтому важной функцией и систем, и технологий является координация лиц, принимающих решения, как на разных уровнях управления, так и на одном уровне.

Основные компоненты. Рассмотрим структуру системы поддержки принятия решений, а также функции составляющих ее блоков, которые определяют основные технологические операции.

В состав системы поддержки принятия решений входят три главных компонента: база данных, база моделей и программная подсистема, которая состоит из СУБД, системы управления базой моделей (СУБМ) и системы управления интерфейсом между пользователем и компьютером.

База данных играет в информационной технологии поддержки принятия решений (СППР) важную роль. Данные могут использоваться непосредственно пользователем для расчетов при помощи математических моделей. Рассмотрим источники данных и их особенности:

1. Часть данных поступает от информационной системы операционного уровня. Чтобы использовать их эффективно, эти данные должны быть предварительно обработаны.

Для этого существуют две возможности:

– использование для обработки данных об операциях фирмы систему управления базой данных, входящую в состав системы поддержки принятия решений;

– обработка за пределами системы поддержки принятия решений, создание для этого специальной базы данных. Этот вариант более предпочтителен для фирм, производящих большое количество коммерческих операций. Обработанные данные об операциях фирмы образуют файлы, которые для повышения надежности и

быстроты доступа хранятся за пределами системы поддержки принятия решений.

2. Помимо данных об операциях фирмы для функционирования системы поддержки принятия решений требуются и другие внутренние данные, например данные о движении персонала, инженерные данные и т.п., которые должны быть своевременно собраны, введены и поддержаны.

3. Важное значение, особенно для поддержки принятия решений на верхних уровнях управления, имеют данные из внешних источников. В числе необходимых внешних данных следует указать данные о конкурентах, национальной и мировой экономике. В отличие от внутренних, внешние данные обычно приобретаются у специализирующихся на их сборе организаций.

4. В настоящее время широко исследуется вопрос о включении в базу данных еще одного источника данных – документов, содержащих записи, письма, контракты, приказы и т.п. Если содержание этих документов будет записано в памяти и затем обработано по некоторым ключевым характеристикам (поставщики, потребители, даты, виды услуг и др.), то система получит новый мощный источник информации.

СУБД должна обладать следующими возможностями:

– составление комбинаций данных, получаемых из различных источников посредством использования процедур агрегирования и фильтрации;

– быстрое прибавление или исключение того или иного источника данных;

– построение логической структуры данных в терминах пользователя;

– использование и манипулирование неофициальными данными для экспериментальной проверки рабочих альтернатив пользователя; БД от других операционных БД, функционирующих в рамках фирмы.

База моделей. Целью создания моделей являются описание и оптимизация некоторого объекта или процесса. Использование моделей обеспечивает проведение анализа в системах поддержки принятия решений. Модели, базируясь на математической интерпретации проблемы, при помощи определенных алгоритмов способствуют нахождению информации, полезной для принятия правильных решений.

Например, модель линейного программирования дает возможность определить наиболее выгодную производственную программу выпуска нескольких видов продукции при заданных ограничениях на ресурсы.

Использование моделей в составе информационных систем началось с применения статистических методов и методов финансового анализа, которые реализовывались командами обычных алгоритмических языков. Позже были созданы специальные языки, позволяющие моделировать ситуации типа «что будет, если?» или «как сделать, чтобы?» Такие языки, созданные специально для построения моделей, дают возможность построить модели определенного типа, обеспечивающие нахождение решения при гибком изменении переменных.

Существует множество типов моделей и способов их классификации, например по цели использования, области возможных приложений, способу оценки переменных и т.п.

По цели использования модели подразделяются на оптимизационные, связанные с нахождением точек минимума или максимума некоторых показателей (например, управляющие часто хотят знать, какие их действия ведут к максимизации прибыли или минимизации затрат) и описательные, которые описывают поведение некоторой системы и не предназначены для целей управления (оптимизации).

По способу оценки модели классифицируются на детерминированные, использующие оценку переменных одним числом при конкретных значениях исходных данных, и стохастические, оценивающие переменные несколькими параметрами, так как исходные данные заданы вероятностными характеристиками.

Детерминированные модели более популярны, чем стохастические, потому что они менее дорогие, их легче строить и использовать. К тому же часто с их помощью получается вполне достаточная информация для принятия решения.

По области возможных приложений модели разбиваются на специализированные, предназначенные для использования только одной системой, и универсальные – для использования несколькими системами.

Специализированные модели более дорогие, они обычно применяются для описания уникальных систем и обладают большей точностью.

В системах поддержки принятия решения база моделей состоит из стратегических, тактических и оперативных моделей, а также ма-

тематических моделей в виде совокупности модельных блоков, модулей и процедур, используемых как элементы для их построения.

Стратегические модели используются на высших уровнях управления для установления целей организации, объемов ресурсов, необходимых для их достижения, а также политики приобретения и использования этих ресурсов. Они могут быть также полезны при выборе вариантов размещения предприятий, прогнозировании политики конкурентов и т.п. Для стратегических моделей характерны значительная широта охвата, множество переменных, представление данных в сжатой агрегированной форме. Часто эти данные базируются на внешних источниках и могут иметь субъективный характер. Горизонт планирования в стратегических моделях, как правило, измеряется в годах. Эти модели обычно детерминистские, описательные, специализированные для использования на одной определенной фирме.

Тактические модели применяются управляющими среднего уровня для распределения и контроля использования имеющихся ресурсов. Среди возможных сфер их использования следует указать финансовое планирование, планирование требований к работникам, планирование увеличения продаж, построение схем компоновки предприятий. Эти модели применимы обычно лишь к отдельным частям фирмы (например, к системе производства и сбыта) и могут также включать в себя агрегированные показатели. Временной горизонт, охватываемый тактическими моделями – от одного месяца до двух лет. Здесь также могут потребоваться данные из внешних источников, но основное внимание при реализации данных моделей должно быть уделено внутренним данным фирмы. Обычно тактические модели реализуются как детерминистские, оптимизационные и универсальные.

Оперативные модели используются на низших уровнях управления для поддержки принятия оперативных решений с горизонтом, измеряемым днями и неделями. Возможные применения этих моделей включают в себя ведение дебиторских счетов и кредитных расчетов, календарное производственное планирование, управление запасами и т. д. Оперативные модели обычно используют для расчетов внутрифирменные данные. Они, как правило, детерминистские, оптимизационные и универсальные (т. е. могут быть использованы в различных организациях).

Математические модели состоят из совокупности модельных блоков, модулей и процедур, реализующих математические методы. Сюда могут входить процедуры линейного программирования, статистического анализа временных рядов, регрессионного анализа и т. п. – от простейших процедур до сложных пакетов прикладных программ. Модельные блоки, модули и процедуры могут использоваться как по отдельности, так и комплексно для построения и поддержания моделей.

Система управления базой моделей (СУБМ) должна обладать следующими возможностями: создавать новые модели или изменять существующие, поддерживать и обновлять параметры моделей, манипулировать моделями.

Система управления интерфейсом. Эффективность и гибкость информационной технологии во многом зависят от характеристик интерфейса системы поддержки принятия решений. Интерфейс определяет: язык пользователя; язык сообщений компьютера, организующий диалог на экране дисплея; знания пользователя.

Язык пользователя – это те действия, которые пользователь производит в отношении системы путем использования возможностей клавиатуры; электронных карандашей, пишущих на экране; джойстика; мыши; команд, подаваемых голосом, и т.п. Наиболее простой формой языка пользователя является создание форм входных и выходных документов. Получив входную форму (документ), пользователь заполняет его необходимыми данными и вводит в компьютер. Система поддержки принятия решений производит необходимый анализ и выдает результаты в виде выходного документа установленной формы.

Значительно возросла за последнее время популярность визуального интерфейса. С помощью манипулятора «мышь» пользователь выбирает представленные ему на экране в форме картинок объекты и команды, реализуя таким образом свои действия.

Управление компьютером при помощи человеческого голоса – самая простая и поэтому самая желанная форма языка пользователя. Она еще недостаточно разработана и поэтому малопопулярна. Существующие разработки требуют от пользователя серьезных ограничений: определенного набора слов и выражений; специальной надстройки, учитывающей особенности голоса пользователя; управления в виде дискретных команд, а не в виде обычной гладкой речи. Технология этого подхода интенсивно совершенствуется, и в ближайшем будущем можно ожидать появления систем поддержки принятия решений, использующих речевой ввод информации.

Язык сообщений – это то, что пользователь видит на экране дисплея (символы, графика, цвет), данные, полученные на принтере, звуковые выходные сигналы и т.п. Важным измерителем эффективности используемого интерфейса является выбранная форма диалога между пользователем и системой. В настоящее время наиболее распространены следующие формы диалога: запросно-ответный режим, командный режим, режим меню, режим заполнения пропусков в выражениях, предлагаемых компьютером.

Каждая форма в зависимости от типа задачи, особенностей пользователя и принимаемого решения может иметь свои достоинства и недостатки.

Долгое время единственной реализацией языка сообщений был отпечатанный или выведенный на экран дисплея отчет или сообщение. Теперь появилась новая возможность представления выходных данных– машинная графика. Она дает возможность создавать на экране и бумаге цветные графические изображения в трехмерном виде. Использование машинной графики значительно повышает наглядность и интерпретируемость выходных данных и становится все более популярным в информационной технологии поддержки принятия решений.

За последние несколько лет наметилось новое направление, развивающее машинную графику, – мультипликация. Мультипликация оказывается особенно эффективной для интерпретации выходных данных систем поддержки принятия решений, связанных с моделированием физических систем и объектов.

Например, система поддержки принятия решений, предназначенная для обслуживания клиентов в банке, с помощью мультипликационных моделей может реально просмотреть различные варианты организации обслуживания в зависимости от потока посетителей, допустимой длины очереди, количества пунктов обслуживания и т.п.

В ближайшие годы следует ожидать использования в качестве языка сообщений человеческого голоса. Сейчас эта форма применяется в системе поддержки принятия решений сферы финансов, где в процессе генерации чрезвычайных отчетов голосом поясняются причины исключительности той или иной позиции.

Знания пользователя – это то, что пользователь должен знать, работая с системой. К ним относится не только план действий, находящийся в голове у пользователя, но и учебники, инструкции, справочные данные, выдаваемые компьютером.

Совершенствование интерфейса системы поддержки принятия решений определяется успехами в развитии каждого из трех указанных компонентов. Интерфейс должен обладать следующими возможностями:

– манипулировать различными формами диалога, изменяя их в процессе принятия решения по выбору пользователя;

– передавать данные системе различными способами;

– получать данные от различных устройств системы в различном формате;

– гибко поддерживать знания пользователя (оказывать помощь по запросу, подсказывать).

7.6. Информационные технологии экспертных систем. Характеристика и назначение

Наибольший прогресс среди компьютерных информационных систем отмечен в области разработки экспертных систем, основанных на использовании искусственного интеллекта. Экспертные системы дают возможность менеджеру или специалисту получать консультации экспертов по любым проблемам, о которых этими системами накоплены знания.

Под искусственным интеллектом обычно понимают способности компьютерных систем к таким действиям, которые назывались бы интеллектуальными, если бы исходили от человека.

Решение специальных задач требует специальных знаний. Однако не каждая компания может себе позволить держать в своем штате экспертов по всем связанным с ее работой проблемам или даже приглашать их каждый раз, когда проблема возникла. 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 (эвристик). Эвристики не гарантируют получения оптимального результата с такой же уверенностью, как обычные алгоритмы, используемые для решения задач в рамках технологии поддержки принятия решений. Однако часто они дают в достаточной степени приемлемые решения для практического использования. Все это дает возможность использовать технологию экспертных систем в качестве советующих.

Сходство информационных технологий, используемых в экспертных системах и системах поддержки принятия решений, состоит в том, что обе они обеспечивают высокий уровень поддержки принятия решений. Однако имеются три существенных различия. Первое связано с тем, что решение проблемы в рамках систем поддержки принятия решений отражает уровень ее понимания пользователем и его возможности получить и осмыслить решение. Технология экспертных систем, наоборот, предлагает пользователю принять решение, превосходящее его возможности. Второе отличие указанных технологий выражается в способности экспертных систем пояснять свои рассуждения в процессе получения решения. Очень часто эти пояснения оказываются более важными для пользователя, чем само решение. Третье отличие связано с использованием нового компонента информационной технологии – знаний.

Основными компонентами информационной технологии, используемой в экспертной системе, являются: интерфейс пользователя, база знаний, интерпретатор, модуль создания системы.

Интерфейс пользователя. Менеджер (специалист) использует интерфейс для ввода информации и команд в экспертную систему и получения выходной информации из нее. Команды включают в себя параметры, направляющие процесс обработки знаний. Информация обычно выдается в форме значений, присваиваемых определенным переменным.

Менеджер может использовать четыре метода ввода информации: меню, команды, естественный язык и собственный интерфейс.

Технология экспертных систем предусматривает возможность получать в качестве выходной информации не только решение, но и необходимые объяснения.

Различают два вида объяснений:

– объяснения по запросам. Пользователь в любой момент может потребовать от экспертной системы объяснения своих действий;

– объяснения полученного решения проблемы. После получения решения пользователь может потребовать объяснений того, как оно было получено. Система должна пояснить каждый шаг своих рассуждений, ведущих к решению задачи. Хотя технология работы с экспертной системой не является простой, пользовательский интерфейс этих систем является «дружественным» и обычно не вызывает трудностей при ведении диалога.

База знаний содержит факты, описывающие проблемную область, а также логическую взаимосвязь этих фактов. Центральное место в базе знаний принадлежит правилам. Правило определяет, что следует делать в данной конкретной ситуации, и состоит из двух частей: условие, которое может выполняться или нет, и действие, которое следует произвести, если выполняется условие.

Все используемые в экспертной системе правила образуют систему правил, которая даже для сравнительно простой системы может содержать несколько тысяч правил.

Все виды знаний в зависимости от специфики предметной области и квалификации проектировщика (инженера по знаниям) с той или иной степенью адекватности могут быть представлены с помощью одной либо нескольких семантических моделей. К наиболее распространенным моделям относятся логические, продукционные, фреймовые и семантические сети.

Интерпретатор. Это часть экспертной системы, производящая в определенном порядке обработку знаний, находящихся в базе знаний. Технология работы интерпретатора сводится к последовательному рассмотрению совокупности правил (правило за правилом). Если условие, содержащееся в правиле, соблюдается, то выполняется определенное действие, и пользователю предоставляется вариант решения его проблемы.

Кроме того, во многих экспертных системах вводятся дополнительные блоки: база данных, блок расчета, блок ввода и корректировки данных. Блок расчета необходим в ситуациях, связанных с принятием управленческих решений. При этом важную роль играет база данных, где содержатся плановые, физические, расчетные, отчетные и другие постоянные или оперативные показатели. Блок ввода и корректировки данных используется для оперативного и своевременного отражения текущих изменений в базе данных.

Модуль создания системы служит для создания набора (иерархии) правил. Существует два подхода, которые могут быть положены в основу модуля создания системы: использование алгоритмических языков программирования и использование оболочек экспертных систем.

Для представления базы знаний специально разработаны языки Лисп и Пролог, хотя можно использовать и любой известный алгоритмический язык.

Оболочка экспертных систем представляет собой готовую программную среду, которая может быть приспособлена к решению определенной проблемы путем создания соответствующей базы знаний. В большинстве случаев использование оболочек позволяет создавать экспертные системы быстрее и легче в сравнении с программированием.

7.7. Эволюция систем поддержки принятия решений

В процессе своего развития системы поддержки принятия решений прошли следующий путь.

Первые системы – системы обработки транзакций (TSP) – это компьютерные системы, предназначенные для выполнения рутинных операций регистрации, накопления, хранения и выдачи информации в заранее заданной форме. Как видим, в рамках таких систем принятие решений обеспечивалось только информацией.

Следующим этапом развития информационных систем было появление концепции автоматизированной системы управления (АСУ). На Западе эта концепция получила название MIS. Это компьютерная система, предназначенная для обеспечения своевременной информацией, необходимой для принятия управленческих решений.

Уровень поддержки решений при использовании данной концепции – информационный, применяются отдельные модели и методы для принятия оптимальных решений.

Отметим, что в существенной мере характер всех поколений систем и их концепций определялся техническими возможностями обработки информации, имеющимися на тот период. Системы автоматизации конторской деятельности (OAS) реализовывали распределенные базы данных. Устранялась излишняя цент поддержки решений – информационный, здесь применяются отдельные модели и методы для

принятия оптимальных решений. OAS – это компьютерная система для выполнения комплекса операций функционирования системы управления как таковой.

Следующий этап – системы поддержки принятия решений (DDS). DDS – диалоговая компьютерная система, использующая формализованные правила и модели объекта управления совместно с базой данных и личным опытом менеджера для выработки и проверки вариантов управленческих решений. Как видим, система этого рода не обеспечивает информационно процесс принятия решений, а участвует в нем. Вершиной развития информационных систем являются экспертные системы (ES). Экспертная система – это компьютерная система, использующая знания одного или нескольких экспертов, представленные в некотором формальном виде, для решения задач принятия решений (ESS – это вариант решений DDS для высшего руководства).

Примеры задач, решаемых с привлечением СППР: выбор методов завоевания рынка бытовой техники; оценка перспективности видов альтернативного горючего для автомобилей.

Итак, система поддержки принятия решений – диалоговая автоматизированная информационная система, использующая правила решений и соответствующие модели с базами данных, а также интерактивный компьютерный процесс моделирования, поддерживающий принятие самостоятельных и неструктурированных решений отдельными менеджерами и личным опытом лица, принимающего решения, для получения конкретных, реализуемых решений проблем, не поддающихся решению обычными методами.

В последнее время системы поддержки и принятия решений начинают применяться и в интересах малого и среднего бизнеса (например, выбор варианта размещения торговых точек, выбор кандидатуры на замещение вакантной должности, выбор варианта информатизации и т.д.). В общем, они способны поддержать индивидуальный стиль и соответствовать персональным потребностям менеджера.

Существуют системы, созданные для решения сложных проблем в больших коммерческих и государственных организациях:

Система авиалиний. В отрасли авиаперевозок используется система поддержки принятия решений – Аналитическая Информационная Система Управления. Она была создана *American Airlines*, но используется и остальными компаниями,

производителями самолетов, аналитиками авиаперевозок, консультантами и ассоциациями. Эта система поддерживает множество решений в этой отрасли путем анализа данных, собранных во время утилизации транспорта, оценки грузопотока, статистического анализа графика. Например, она позволяет делать прогнозы для авиарынка по долям компаний, выручке и рентабельности. Эта система позволяет руководству авиакомпании принимать решения относительно цены билетов, запросов в транспорте и т.д.

Географическая система. Географическая информационная система – это специальная категория систем поддержки, которая позволяет интегрировать компьютерную графику с географическими БД и с другими функциями систем поддержки принятия решений. Например, *IBMs GeoManager* – это система, которая позволяет конструировать и показывать карты и другие визуальные объекты для помощи при принятии решений относительно географического распределения людей и ресурсов. Например, она позволяет создать географическую карту преступности и помогает верно перераспределить силы полиции. Также ее используют для изучения степени урбанизации, в лесной промышленности, железнодорожном бизнесе и т.д.

Контрольные вопросы

1. Перечислите виды информационных технологий (ИТ) по степени охвата задач управления.
2. Дайте характеристику и опишите назначение ИТ обработки данных.
3. Охарактеризуйте основные компоненты ИТ обработки данных.
4. Дайте характеристику и опишите назначение ИТ управления.
5. Охарактеризуйте основные компоненты ИТ управления.
6. Какие задачи относятся к офисным?
7. Что называют электронным офисом?
8. Дайте характеристику и опишите назначение ИТ автоматизации офиса.
9. Охарактеризуйте основные компоненты ИТ автоматизации офиса.

10.Дайте характеристику и опишите назначение ИТ поддержки принятия решений.

11.Охарактеризуйте основные компоненты ИТ поддержки принятия решений.

12.Что является главной особенностью информационной технологии поддержки принятия решений?

13.Какими возможностями должна обладать система управления базой моделей?

14.Из каких моделей состоит база моделей в системах поддержки принятия решения?

15.Дайте характеристику и опишите назначение ИТ экспертных систем.

16.Перечислите основные компоненты ИТ экспертных систем.

17.Что понимают под искусственным интеллектом?

18.В чем состоит сходство информационных технологий, используемых в экспертных системах и системах поддержки принятия решений?

19.Что содержится в базе знаний?

20.Какова эволюция систем поддержки принятия решений?

СПИСОК ЛИТЕРАТУРЫ

1. Батин, Н. В. Основы информационных технологий : учеб.-метод. пособие / Н. В. Батин [и др.] под общ. ред. В. В. Шкурко. – Минск : Институт подготовки научных кадров НАН Беларуси, 2008. – 253 с.

2. Быков, В. Л. Информатика: пособие / В. Л. Быков, Н. Г. Серебрякова. – Минск : БГАТУ, – 2013. – 656 с.

3. Морозевич, А. Н. Информатика / А. Н. Морозевич, А. М. Зеневич. – Минск : Высшая школа, 2006. – 285 с.

4. Сапун, О. Л. Компьютерные информационные технологии: учеб.-метод. комплекс / О. Л. Сапун [и др.]. – Минск : БГАТУ, 2012. – 160 с.

5. Фурунжиев, Р. И. Основы программирования (в визуальной среде Delphi). В 2-х ч. / Р. И. Фурунжиев, Т. В. Ероховец, Е. М. Исаченко. – Минск : БГАТУ, 2010.

6. Методические указания к лабораторно-практическим занятиям по дисциплине «Вычислительная техника и информатика» «Команды системы MATLAB» / сост. Б. М. Киселев, М. А. Прищепов. – Минск, 2004. – 74 с.

7. Бахвалов, Н. С. Численные методы / Н. С. Бахвалов, Н. П. Жидков, Г. Н. Кобельков. М. : БИНОМ. Лабаратория знаний, 2003. – 632 с.

8. Бахвалов, Н. С. Численные методы в задачах и упражнениях / Н. С. Бахвалов, А. В. Лапин, Е. В. Чижонков. – М.: Высшая школа, 2000. 192 с.

9. Вержбицкий, В. М. Численные методы. Линейная алгебра и нелинейные уравнения / В. М. Вержбицкий. – М. : Высшая школа, 2000. – 268 с.

10.Вержбицкий, В. М. Численные методы. Математический анализ и обыкновенные дифференциальные уравнения / В. М. Вержбицкий. – М. : Высшая школа, 2001. – 383 с.

11.Волков, Е. А. Численные методы / Е. А. Волков. – СПб. : Лань, 2004. – 248 с.

12.Шуп, Т. Е. Прикладные численные методы в физике и технике / Т. Е. Шуп. М. : Высшая школа, 1990. – 255 с.

ДЛЯ ЗАМЕТОК

Учебное издание

Серебрякова Наталья Григорьевна
Сапун Оксана Леонидовна
Фурунжиев Риза Ибраимович

ОСНОВЫ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Пособие

Ответственный за выпуск *Н. Г. Серябрякова*
Редактор *Д. О. Бабакова*
Корректор *В. А. Лукьянчук*
Компьютерная верстка *В. А. Лукьянчука*
Дизайн обложки *Д. О. Бабаковой*

Подписано в печать 18.06.2015. Формат $60\times84^{1}/_{16}$.
Бумага офсетная. Ризография.
Усл. печ. л. 23,25. Уч.-изд. л. 18,18. Тираж 98 экз. Заказ 482.

Издатель и полиграфическое исполнение:
Учреждение образования
«Белорусский государственный аграрный технический университет».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий
№ 1/359 от 09.06.2014.
№ 2/151 от 11.06.2014.
Пр. Независимости, 99–2, 220023, Минск.