В.А. Острейковский, И.В. Полякова

ИНФОРМАТИКА
Теория и практика

Учебное пособие

Допущено
Министерством образования и науки
Российской Федерации в качестве учебного пособия
для студентов учреждений среднего профессионального
образования

Москва
ОНИКС

УДК 007
ББК 32.81
О-76

Издано при финансовой поддержке Федерального агентства по печати и массовым коммуникациям в рамках Федеральной целевой программы «Культура России»

Рецензенты:
зав. кафедрой «Прикладная математика» Обнинского государственного технического университета атомной энергетики
доктор физико-математических наук, профессор *В.А. Галкин;*
директор Центра новых информационных технологий Сургутской высшей гимназии-лаборатории *С.П. Митрофанов*

Острейковский В.А.

О-76 Информатика. Теория и практика: Учеб. пособие / В.А. Острейковский, И.В. Полякова. — М.: Издательство Оникс, 2008. — 608 с.: ил.

ISBN 978-5-488-02110-5

В учебном пособии в соответствии с требованиями Государственного образовательного стандарта рассмотрены процессы получения, преобразования, хранения и использования информации. Выделены семь содержательных линий дисциплины: информация и информационные процессы, системы счисления и основы логики, компьютер, информационные технологии, моделирование и формализация, алгоритмизация, программирование.

Выгодно отличает данное издание от существующих учебных книг по информатике наличие в нем методически выверенного цикла практических работ с подробным алгоритмом их выполнения, что обеспечивает прочное усвоение теоретических знаний и приобретение необходимых умений и навыков. Изложение наиболее сложных тем курса предваряют методические указания для преподавателей.

Для студентов учреждений среднего профессионального образования и всех тех, кто стремится самостоятельно овладеть информационными технологиями и эффективно использовать персональные компьютеры.

УДК 007
ББК 32.81

ISBN 978-5-488-02110-5

ОГЛАВЛЕНИЕ

Список основных сокращений

АИС — автоматизированная информационная система
АЛУ — арифметико-логическое устройство
АПД — аппаратура передачи данных
АРМ — автоматизированное рабочее место
АСНИ — автоматизированная система научных исследований
АСУ — автоматизированная система управления
АСУП — автоматизированная система управления предприятием
АСУТП — автоматизированная система управления технологическими процессами
АЦПУ — алфавитно-цифровое печатающее устройство
БЗ — база знаний
БД — база данных
БИС — большая интегральная схема
БнД — банк данных
Бит — [от англ. bi(nary digi)t — двоичная цифра] — двоичная единица измерения энтропии и количества информации
ВЗУ — внешнее запоминающее устройство
ВС — вычислительная система
ВСт — вычислительная сеть
ВТ — вычислительная техника
ВУ — внешнее устройство
ГМД — гибкий магнитный диск
ЕС ЭВМ — единая система ЭВМ
ЗУ — запоминающее устройство
ИВЦ — информационно-вычислительный центр
ИИ — искусственный интеллект
ИР — информационный ресурс
ИС — информационная система
ИТ — информационная технология
К — контроллер
КБ — конструкторское бюро
КМ — канал мультиплексный
КС — канал селекторный
ЛВС — локальная вычислительная сеть
МД — магнитный диск
МЛ — магнитная лента

МП — микропроцессор
НГМД — накопитель на гибком магнитном диске
НЖМД — накопитель на жестком магнитном диске
НМД — накопитель на магнитном диске
НМЛ — накопитель на магнитной ленте
ОГАС — общегосударственная автоматизированная система
ОЗУ — оперативное запоминающее устройство
ОП — оперативная память
ОС — операционная система
ПК — персональный компьютер
ПО — программное обеспечение
ПП — постоянная память
ПЗУ — постоянное запоминающее устройство
ППП — пакет прикладных программ
ПУ — печатающее устройство
ПЭВМ — персональная ЭВМ
САПР — система автоматизированного проектирования
СБИС — сверхбольшая интегральная схема
СВМ — система виртуальных машин
СЗ — система знаний
СОП — сверхоперативная память
СМ ЭВМ — серия мини-ЭВМ
СП — сопроцессор
СПО — система поддержки оперативного персонала
СУБД — система управления базами данных
ТОУ — технологический объект управления
УВВ — устройство ввода-вывода
УВв — устройство ввода
УВыв — устройство вывода
УУ — устройство управления
УПД — устройство подготовки данных
УСО — устройство связи с объектом
ЦВМ — цифровая вычислительная машина
ЦП — центральный процессор
ЭВМ — электронная вычислительная машина
ЭЛТ — электронно-лучевая трубка
ЭС — экспертная система
CR-ROM — Compact Disk — Read Only Memory
IBM — International Business Machines Corporation
MS — Microsoft
WWW(W3) — World Wide Web

Предисловие

Во второй половине XX в. человечество вступило в новый этап своего развития: начался переход от индустриального общества к информационному. Процесс, обеспечивающий этот переход, получил название **информатизации**. **Информатизация** — это процесс создания, развития и всеобщего применения информационных средств и технологий, обеспечивающих достижение и поддержание уровня информированности всех членов общества, необходимого и достаточного для кардинального улучшения качества труда и условий жизни в обществе. При этом информация становится важнейшим стратегическим ресурсом общества и занимает ключевое место в экономике, образовании и культуре.

Неизбежность информатизации обусловлена резким возрастанием роли и значения информации. Информационное общество характеризуется высокоразвитой информационной сферой, которая включает деятельность человека по созданию, переработке, хранению, передаче и накоплению информации.

Научным фундаментом процесса информатизации общества служит новая научная дисциплина — информатика.

Целями данного учебного пособия являются:

— четкое изложение основных понятий и современных подходов к информатике как естественнонаучной дисциплине;

— изучение математических основ информатики как инструмента для решения прикладных задач;

— освоение первоначальных знаний в области структуры и функций блоков ЭВМ, алгоритмизации и программирования;

— демонстрация возможностей информатики в современных информационных технологиях;

— приобретение студентами необходимых умений и навыков на практических занятиях.

В соответствии с поставленными целями учебный материал изложен в семи разделах и 24 темах. В первых трех разделах подробно раскрываются объект и предмет информатики, системы счисления и основы логики; дается количественная оценка информации с описанием форм ее представления и преобразования; излагаются сведения о технических средствах информационных процессов, о структурах и функциях основных блоков электронных вычислительных машин (ЭВМ) и истории развития электронных вычислительных средств, о составе и назначении узлов и устройств персональной ЭВМ (ПЭВМ) и персонального компьютера (ПК), об организации и представлении данных, об интерфейсе пользователя.

В четвертом разделе — одном из центральных в книге — рассмотрены примеры информационных технологий, которые позволяют студенту овладеть современными приложениями информатики как науки не только сегодня, но и в ближайшей перспективе; содержится материал по технологии обработки на ЭВМ текстовой, числовой и графической информации, по технологии хранения, поиска и сортировки информации; большое внимание уделено мультимедийным технологиям и компьютерным телекоммуникациям, локальным и глобальным компьютерным сетям, базам, банкам и хранилищам данных; использованию информационных ресурсов сети Internet; классификации, содержанию и применению пакетов прикладных программ (ППП) как одного из эффективных способов внедрения информатики в практику.

В пятом разделе излагаются основные понятия моделирования как метода познания объективного мира; рассмотрены примеры материальных и информационных моделей, причем основной уклон сделан в сторону построения информационных моделей.

Шестой и седьмой разделы посвящены алгоритмизации и программированию. Здесь детально изложены понятия и свойства алгоритмов, способы их записи и основные типы алгоритмов; описан один из языков программирования — Паскаль; приведены различные технологии программирования: алгоритмическое, объектно-ориентированное, программирование методом последовательной детализации («сверху вниз») и сборочным методом («снизу вверх»); кратко охарактеризованы современные языки и средства программирования.

Для закрепления теоретических знаний и приобретения необходимых умений и навыков в третьем, четвертом и седьмом разделах предложен цикл практических работ, в ходе выполнения которых студенты должны овладеть умениями и навыками создания электронного документа с помощью технологии мультимедиа; поиска нужной информации в сети Internet; разработки и программирования задач с циклической структурой, массивами, матрицами с использованием записей, с обработкой текстовой информации, с использованием внешних файлов и подпрограмм, стандартных алгоритмов и нестандартных функций; создания простейших графических изображений; построения графиков функций и диаграмм.

В Заключении рассмотрены перспективы развития информатики.

Основными отличиями предлагаемого пособия от существующих учебных книг по информатике являются:

— систематическое изложение теоретической и прикладной информатики в соответствии с требованиями Государственного стандарта среднего профессионального образования;

— ориентация на конкретные учебные программы по образовательным областям «Математика», «Естествознание» и дисциплинам гуманитарного профиля;

— изложение материала с двух позиций: «информационные технологии — информационный ресурс» и «модель — алгоритм — программа»;

— наличие в одном издании теоретических основ информатики и методически выверенного цикла практических работ.

В результате изучения курса информатики студент должен:

знать:

— функции языка как способа представления информации;

— способы хранения и основные виды хранилищ информации;

— основные единицы измерения количества информации;

— правила выполнения арифметических операций в двоичной системе счисления;

— основные логические операции, их свойства и обозначения;

— общую функциональную схему компьютера;

— назначение и основные характеристики устройств компьютера;

— назначение и основные функции операционной системы;

— назначение и возможности электронных таблиц;

— назначение и основные возможности баз данных;

— основные объекты баз данных и допустимые операции над ними;

— этапы информационной технологии решения задач с использованием компьютера;

уметь:

— приводить примеры получения, передачи и обработки информации в деятельности человека, живой природе, обществе и технике;

— перечислять основные характерные черты информационного общества;

— переводить числа из одной системы счисления в другую;

— строить логические схемы из основных логических элементов по формулам логических выражений;

— применять текстовый редактор для редактирования и форматирования текстов;

— применять графический редактор для создания и редактирования изображений, строить диаграммы;

— применять электронные таблицы для решения задач;

— создавать простейшие базы данных, осуществлять сортировку и поиск информации в базе данных, перечислять и описывать различные типы баз данных;

— работать с файлами (создавать, копировать, переименовывать, осуществлять поиск файлов), вводить и выводить данные;

— работать с носителями информации, пользоваться антивирусными программами;

— записывать на языке программирования алгоритмы решения учебных задач и отлаживать их;

иметь представление:

— об информационных основах процессов управления;

— о методах поиска информации;

— о принципах кодирования информации, о системах счисления;

— о возможности соединения разнотипной информации в одном электронном документе с помощью технологии мультимедиа;

— о работе электронной почты, об информационных ресурсах и технологии поиска информации в сети Internet.

Некоторые темы изложены авторами несколько шире примерной программы дисциплины «Информатика», заложенной в Государственный образовательный стандарт. Это даст возможность учебному заведению использовать материал данной книги для более рационального построения рабочей учебной программы по информатике.

Авторы считают своим долгом выразить искреннюю признательность доктору физико-математических наук, профессору В.А. Галкину и директору Центра новых информационных технологий Сургутской высшей гимназии-лаборатории С.П. Митрофанову — за ценные замечания, сделанные при рецензировании рукописи; коллегам и сотрудникам кафедры информатики и вычислительной техники Сургутского государственного университета — за большую помощь в отборе материала учебного пособия и участие в дискуссии по его содержанию, а также С.П. Саакяну и Н.В. Темирбаевой — за помощь в подготовке рукописи к печати.

Особую благодарность авторы выражают ректору Сургутского государственного университета профессору Г.И. Назину за внимание к работе над рукописью.

Раздел 1

ИНФОРМАЦИЯ И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

В результате изучения материала первого раздела студент должен:

знать:

— основные понятия информации и информационных процессов;

— роль каналов связи как главного элемента между источником информации и ее получателем;

— виды, особенности и формы информационного ресурса;

уметь:

— приводить примеры получения, передачи и обработки информации в деятельности человека, живой природе, обществе и технике;

— перечислять основные характерные черты информационного общества;

— перечислять основные компоненты информационной культуры человека;

иметь представление:

— об информационных основах процессов управления;

— о методах поиска информации.

Тема 1.1

ВВЕДЕНИЕ В ДИСЦИПЛИНУ. ЧЕЛОВЕК И ИНФОРМАЦИЯ

Методические указания

В результате изучения данной темы студенты должны: знать определения объекта и предмета информатики; четко представлять смысл понятий «информация», «данные», «знания»;

разбираться в содержании терминов «информационная технология» и «информационный ресурс»; запомнить определение и содержание естественнонаучной дисциплины «Информатика», а также понимать значение и необходимость информатики и ЭВМ в жизни общества.

Уже в начале изучения курса студенты должны усвоить понятие канала связи как необходимого элемента между источником информации и ее получателем и роль в этом процессе ЭВМ. Краткое изложение истории развития информатики способствует становлению естественнонаучного мышления. Освоение содержания темы 1.1 дает возможность ответить на ряд вопросов: что такое информатика? Для чего она нужна? Каково содержание информатики? Какие задачи она решает?

Преподаватель обязан донести до студента главную мысль: появление науки информатики – это объективный процесс развития общества, без внедрения информатизации оно не может развиваться дальше.

1.1.1. Терминология информатики

Термин «информация» имеет множество определений. Вот как он трактуется в «Энциклопедии кибернетики»: «Информация (от лат. informatio — разъяснение, изложение, осведомленность) — одно из наиболее общих понятий науки, обозначающее некоторые сведения, совокупность каких-либо данных, знаний и т. п.». В широком смысле информация — это отражение реального мира; в узком смысле — это любые сведения, являющиеся объектом хранения, передачи и преобразования.

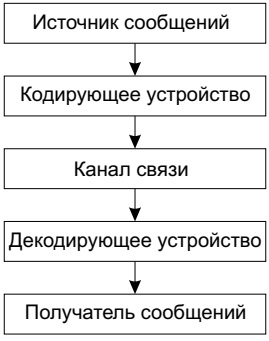


Рис. 1.1. Общая схема передачи информации

Понятие «информация» вместе с понятиями «вещество» и «энергия» относится к основным понятиям науки. С практической точки зрения информация всегда представляется в виде сообщения. Информационное сообщение связано с источником сообщения, получателем сообщения и каналом связи (рис. 1.1).

Сообщение от источника к приемнику передается в материально-энергетической форме (электрический, световой, звуковой сигналы и т. д.). Человек воспринимает сообщения посредством органов чувств. Приемники информации в технике воспринимают сообщения с помощью различной измерительной и регистрирующей аппаратуры. В обоих случаях с приемом информации связано изменение во времени какой-либо величины, характеризующей состояние приемника. В этом смысле информационное сообщение можно представить функцией $x(t)$, характеризующей изменение во времени материально-энергетических параметров физической среды, в которой осуществляются информационные процессы.

Функция $x(t)$ принимает любые вещественные значения в диапазоне изменения времени t. Если функция $x(t)$ непрерывна, то имеет место непрерывная (или аналоговая) информация, источником которой обычно являются различные природные объекты (например, температура, давление, влажность воздуха), объекты технологических производственных процессов (например, нейтронный поток в активной зоне, давление и температура теплоносителя в контурах ядерного реактора). Если функция $x(t)$ дискретна, то информационные сообщения, используемые человеком, имеют характер дискретных сообщений (например, сигналы тревоги, передаваемые посредством световых и звуковых сообщений; языковые сообщения, передаваемые в письменном виде или с помощью звуковых сигналов; сообщения, передаваемые с помощью жестов, и др.).

В современном мире информация, как правило, обрабатывается на вычислительных машинах. Поэтому информатика тесно связана с инструментарием — вычислительной машиной.

Компьютер — устройство преобразования информации посредством выполнения управляемой программой последовательности операций. Синоним слова «компьютер» — вычислительная машина (чаще — электронная вычислительная машина (ЭВМ)).

В зависимости от вида перерабатываемой информации вычислительные машины (ВМ) подразделяют на два основных

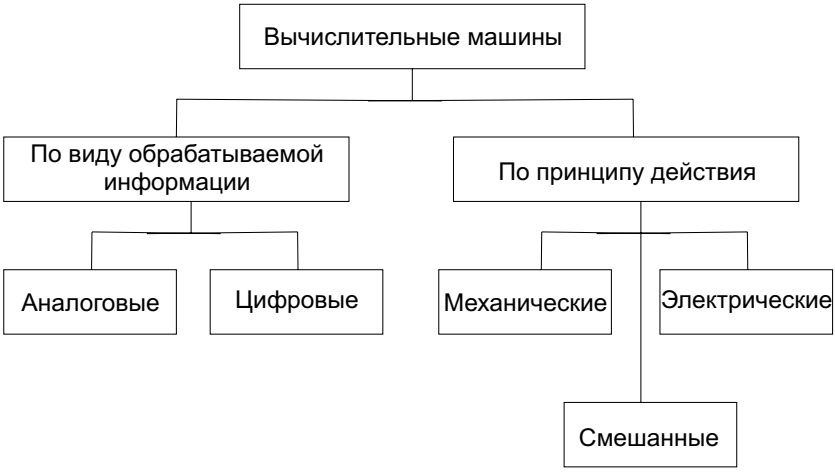


Рис. 1.2. Классы вычислительных машин

класса: аналоговые и цифровые (рис. 1.2). **Аналоговая вычислительная машина (АВМ)** — это машина, оперирующая информацией, представленной в виде непрерывных изменений некоторых физических величин. При этом в качестве физических переменных выступают сила тока электрической цепи, угол поворота вала, скорость и ускорение движения тела и т. п. Используя тот факт, что многие явления в природе математически описываются одними и теми же уравнениями, АВМ позволяют с помощью одного физического процесса моделировать различные процессы.

Цифровая вычислительная машина (ЦВМ) — это машина, оперирующая информацией, представленной в дискретном виде. В настоящее время разработаны методы численного решения многих видов уравнений, что дало возможность решать на ЦВМ различные уравнения и задачи с помощью набора простых арифметических и логических операций. Поэтому если АВМ обычно предназначены для решения определенного класса задач, т. е. являются специализированными, то ЦВМ, как правило, универсальное вычислительное средство. Наибольшее распространение получили электронные вычислительные машины, выполненные с использованием новейших достижений электроники.

Очень широко употребляется еще один термин — «данные» (от лат. data). Его принято применять в отношении информации, представленной в виде, позволяющем хранить, передавать или обрабатывать ее с помощью технических средств. Поэтому наряду с терминами «ввод информации», «обработка информации», «хранение информации», «поиск информации» используются термины «ввод данных», «обработка данных», «хранение данных» и т. п.

Информация существует не только в виде данных, но и в виде знаний. Здесь **знания** — это совокупность объективных фактов, способов и технологий, систематизированных и дающих реальное представление о предметах, процессах и явлениях, т. е. специальным образом структурированная информация.

Данные и знания в виде информационных потоков циркулируют в **информационных системах (ИС)**. Сбор, накопление, обработка, хранение и использование информации в ИС осуществляются с помощью информационных технологий. **Информационные технологии (ИТ)** — это машинизированные (инженерные) способы обработки информации, которые реализуются посредством **автоматизированных информационных систем (АИС)**.

Объектом информатики выступают автоматизированные (человеко-машинные), основанные на ЭВМ и телекоммуникационной технике информационные системы различных классов и назначения. Поэтому рассмотрим более подробно объект и предмет информатики.

1.1.2. Объект и предмет информатики

АИС как объект информатики получили широчайшее распространение. Классификацию АИС осуществляют по ряду признаков, и в зависимости от решаемой задачи можно выбрать разные признаки классификации. При этом одна и та же АИС может характеризоваться одним или несколькими признаками. В качестве признаков классификации АИС выступают область применения, охватываемая территория, организация информационных процессов, направление деятельности, назначение, структура и др.

Классификация АИС по направлению деятельности показана на рис. 1.3.

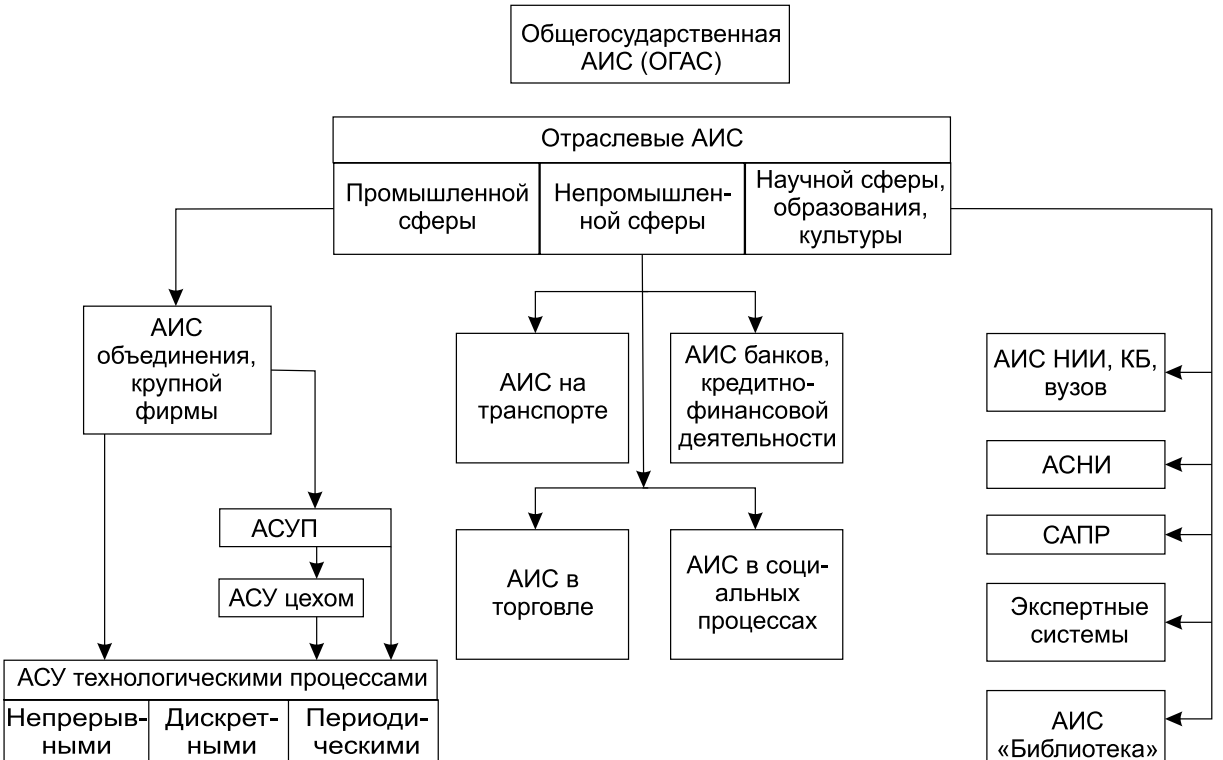


Рис. 1.3. Классификация АИС по направлению деятельности

В промышленной сфере превалирует отраслевой характер иерархии АИС. Классификация АИС по территориальному признаку приведена на рис. 1.4.

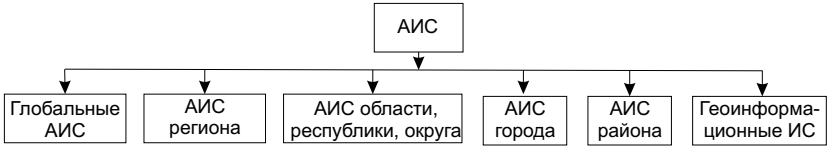


Рис. 1.4. Классификация АИС по территории

В зависимости от организации информационных процессов АИС подразделяют на управляющие и информационные. В информационных системах управление отсутствует (автоматизированные системы научных исследований — АСНИ, «Библиотека», системы автоматизированного проектирования — САПР, экспертные системы — ЭС и др.).

По области применения АИС классифицируют на административные, производственные, учебные, медицинские, военные, метеорологические, экологические, криминалистические и др. В данном случае назначение и структура построения АИС характеризуются наличием соответствующих подсистем (рис. 1.5). Этот класс АИС исторически является одним из первых на производстве.

АИС различают также по уровню развития — в зависимости от поколений ЭВМ, на которых они базируются. Разнообразие ИТ и АИС постоянно растет.

Следует выделить ИТ и АИС, с помощью которых регулируются микросоциальные процессы. Это ИТ и АИС денежно-кассовых операций, распределения мест на транспорте и в гостиницах, обработки метеорологической информации и т. п. Они также относятся к объекту информатики.

Здесь необходимо особо отметить тот факт, что объектом информатики выступают не сами по себе ЭВМ как программно-технические комплексы, а ИТ, воплощением которых служат компьютеризованные, или человеко-машинные, точнее социотехнические, системы. В отличие от производственных, энергопреобразующих технологий, ИТ (объект информатики) относятся к социальным, знаниепреобразующим технологиям.

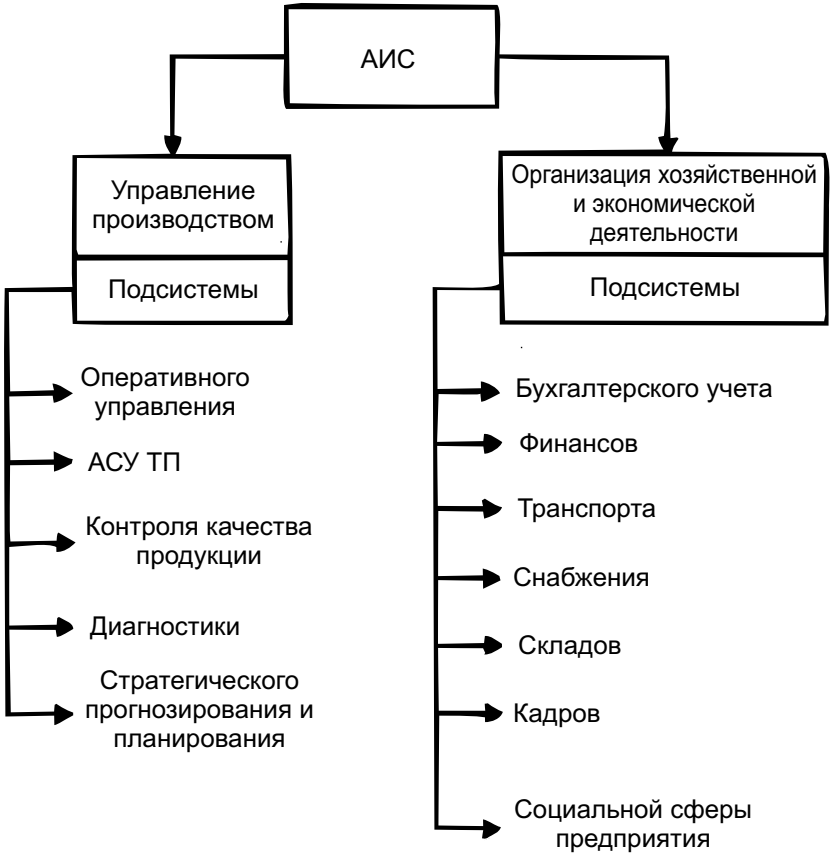


Рис. 1.5. Классификация АИС на предприятии

Переход к ИТ как технической базе автоматизированных информационно-управляющих систем обнажил и до крайности обострил проблемы, относящиеся ко всему технологическому циклу сбора, переработки и применения информации в планово-управленческих, познавательных и других процессах, выдвинул на первый план общие проблемы содержания информационных процессов и значения ИТ. Говорить о воздействии науки на что-либо вне информационного процесса бессмысленно. Научная идея должна превратиться в информацию, т. е. быть закодированной, переданной по каналу связи, принятой адресатом, чтобы ее можно было применить на практике. Чем больше потенциал знаний, тем важнее задача

развития информационно-коммуникативных сфер экономики. Знания должны определенным образом фиксироваться, трансформироваться, распределяться, приниматься и перерабатываться.

Информационные технологии и выступили новым средством превращения знаний в информационный ресурс (ИР) общества, его новым движущим фактором, средством его эффективного использования. Информационный ресурс наряду с другими ресурсами общества (материальными, финансовыми, природными, трудовыми и др.) стал основным ресурсом человечества, главной ценностью современной цивилизации. Но возникли и сложные проблемы, относящиеся к роли, механизму функционирования, социальным последствиям использования ИР. Для их решения и появилась новая наука — информатика.

Предметом информатики как новой фундаментальной науки выступает информационный ресурс — его сущность, законы функционирования, механизмы взаимодействия с другими ресурсами общества и воздействия на социальный прогресс. Переход на уровень ИР в его содержательной трактовке означает переход к изучению внутренних связей и закономерностей, основанных на использовании ИТ.

1.1.3. Виды, особенности и формы информационного ресурса

Понятие информационного ресурса сводится к следующему: ИР — это знание, обладающее всеми атрибутами понятия «информация»; и наоборот: ИР — это информация, которая имеет все свойства понятия «знание». Можно сказать и так: ИР — это информация, т. е. информация в виде понятийного знания.

Принципиальное значение в понимании ИР имеет форма существования и представления знаний. Знание — это отражение той или иной стороны объективной действительности в виде идей, понятий, представлений о каком-то предмете или явлении. Здесь речь идет о понятийном знании — знании в смысловом выражении. Имеется, как известно, и другой вид знаний —

знания интуитивные, рожденные в подсознательной сфере человека, которые, прежде чем стать сообщениями, должны быть выражены в виде понятий.

Знания, будучи живой, диалектической системой, передаются другим людям, материализуются и существуют в трех видах:

— «живые» знания (квалификация);

— овеществленные знания;

— информация (сообщения).

Особо следует отметить, что не все знания, в частности книги, газеты, радио- и телепередачи, патенты и т. п., выступают в качестве ИР. Появляется новая проблема — информативность сообщений. Книги, патентные описания и другие сообщения еще должны найти своих потребителей. Из них еще должна быть извлечена информация. *Перед человечеством стоит громадная по своей важности и сложности задача: извлечь максимум информации из сообщений, накопленных за всю историю человечества, и превратить ее в активно функционирующий ресурс.* Речь идет о превращении книжных описаний и других «рассеянных» знаний в алгоритмы и программы — это часть работ по формированию ИР.

Главная трудность в понимании природы и функций ИР как интеллектуального ресурса, фактора коллективного творчества состоит в раскрытии механизма перехода знаний в силу, способов его воздействия на материальные факторы прогресса.

Но не сам по себе ИР «движет» системы. Движителями систем выступают материальные силы: энергетические и трудовые факторы. ИР же, словно фермент, переводит эти материальные факторы из латентного в активное состояние и вводит их в заданное русло, обеспечивая тем самым прирост свободной энергии в целенаправленных системах социальной природы за счет снижения их энтропии.

Каковы же основные особенности информационного ресурса?

Первая особенность ИР: в отличие от других видов ресурсов (в частности, материальных), он практически неисчерпаем; по мере развития общества и роста потребления знаний их запасы не убывают, а растут.

Вторая особенность ИР: по мере использования он не исчезает, а сохраняется и даже увеличивается (за счет трансфор-

мации полученных сообщений с учетом опыта и местных условий).

Третья особенность ИР: он несамостоятелен и сам по себе имеет лишь потенциальное значение. Только соединяясь с другими ресурсами — опытом и квалификацией персонала, техникой, энергией, сырьем, — ИР проявляется как движущая сила.

Четвертая особенность ИР: эффективность его применения связана с эффектом повторного производства знаний. «Продукт умственного труда — наука — всегда ценится далеко ниже ее стоимости, потому что рабочее время, необходимое для ее восприятия, не идет ни в какое сравнение с тем рабочим временем, которое требуется для того, чтобы первоначально ее произвести. Так, например, теорему о биноме школьник может выучить в течение одного часа» (К. Маркс). Информационное взаимодействие позволяет получить новое знание ценой значительно меньших затрат по сравнению с затратами труда, энергии, времени, затраченными на его прямое генерирование.

Пятая особенность ИР: он является формой непосредственного включения науки в состав производительных сил. В индустриальном обществе наука выступает опосредованной производительной силой. Посредниками при этом служат машины, новые материалы и препараты. В информационном обществе наука представляет собой непосредственную производительную силу, выступая в форме ИР. Радикально изменяется характер действия ИР: производительность труда повышается не на проценты, а на порядки — в 10—100 раз. Осуществляется переход к производственным системам, основой функционирования которых служат ИР и информационные связи.

Шестая особенность ИР: он возникает в результате не просто умственного труда, а его творческой части. Рутинная часть умственной работы сама по себе не информативна: она не увеличивает потенциала нужных знаний, не меняет представления о путях достижения цели. Любой умственный труд, будь то наука или управление, включает две части: рутинную и творческую. Увеличение умственной работы за счет рутинной ее части не ведет к росту ИР. Вместе с тем трудовые и материально-энергетические затраты необходимы при выработке, передаче и использовании ИР.

Существует две формы ИР как отчуждаемых знаний, становящихся сообщениями: пассивная и активная.

К *пассивным формам ИР* относятся книги, журнальные статьи, патенты и банки данных, а также знания, привязанные к конкретным предметным областям (например, выборки, извлечения данных и т. п.), если они не комплексные, т. е. недостаточны для целенаправленного применения.

Активные формы ИР: модель, алгоритм, программа, проект и база знаний (БЗ). Их можно трактовать в целом как стадии созревания ИР, степени доведения его до готовности и превращения в силу. Естественно, что каждая из этих форм ИР имеет разный научно-технический уровень и различную завершенность.

Модель — это описание системы, отображающее определенную группу ее свойств. Создание модели системы позволяет предсказывать ее поведение в определенном диапазоне условий.

Алгоритмы подразделяют в зависимости от степени общности. Важно стремиться к созданию решающих алгоритмов.

Программа и **проект** — конечные, синтетические формы существования ИР в его жизненном цикле. В виде программы и проекта, а часто в виде решающего алгоритма ИР непосредственно противостоит энтропии рассматриваемого объекта. В этом плане вводится понятие *информационной емкости программы,* которое означает величину потенциального уменьшения неопределенности. В ходе реализации программы идет как бы заполнение пространства объекта информацией, которая в нем сконцентрирована.

Модель, алгоритм, программа, проект и особенно БЗ как активные формы ИР — это антиэнтропийные инструменты. Однако программа и проект выделяются среди них своей закономерностью и готовностью к прямому информационному воздействию на объект с целью снятия его неопределенности.

Теперь можно дать определение понятия «информатика». **Информатика** — это научная дисциплина, изучающая закономерности, методы и способы накопления, хранения, передачи и использования информации с помощью ЭВМ и других технических средств.

Информатика как наука о законах получения, передачи и использования ИР в общественной практике подводит теоре-

тический фундамент под использование ЭВМ в автоматизированных системах, которые и предназначены для усиления информационных процессов в обществе, для использования ИР. Речь идет прежде всего о специальных ИР, основанных на компьютерной технике и реализующих ИТ, т. е. инженерную обработку знаний (Knowledge Engineering).

1.1.4. Краткая история развития информатики

Информатика как наука стала развиваться с середины прошлого столетия, что связано с появлением ЭВМ и начавшейся компьютерной революцией. Появление вычислительных машин в 1950-е гг. создало для информатики необходимую аппаратную поддержку, т. е. благоприятную среду для ее развития как науки. Всю историю информатики принято подразделять на два больших этапа: предысторию и историю.

Предыстория информатики такая же древняя, как и история развития человеческого общества. В предыстории также выделяют (весьма приближенно) ряд этапов. Каждый из них характеризуется резким возрастанием, по сравнению с предыдущим этапом, возможностей хранения, передачи и обработки информации.

Начальный этап предыстории информатики — освоение человеком развитой устной речи. Членораздельная речь, язык стали специфическим социальным средством хранения и передачи информации.

Второй этап — возникновение письменности. На этом этапе резко возросли возможности хранения информации. Человек получил искусственную внешнюю память. Организация почтовых служб позволила использовать письменность и как средство для передачи информации. Кроме того, возникновение письменности было необходимым условием для начала развития наук (вспомним, например, Древнюю Грецию). С этим же этапом, по всей видимости, связано и возникновение понятия «натуральное число». Все народы, обладавшие письменностью, владели понятием числа и пользовались той или иной системой счисления.

Третий этап — книгопечатание. Его можно смело назвать первой информационной технологией. Воспроизведение ин-

формации было поставлено на поток, на промышленную основу. По сравнению с предыдущим на этом этапе не столько увеличились возможности хранения информации (хотя и здесь был выигрыш: письменный источник — это часто один-единственный экземпляр, печатная книга — это целый тираж экземпляров, а следовательно, и малая вероятность потери информации при хранении, как это случилось, например, с рукописью «Слова о полку Игореве»), сколько повысились доступность информации и точность ее воспроизведения.

Четвертый (последний) *этап предыстории информатики связан с успехами точных наук* (прежде всего математики и физики) *и начинающейся научно-технической революцией*. Этот этап характеризуется возникновением таких мощных средств связи, как радио, телефон и телеграф, а позднее и телевидение. Появились новые возможности получения и хранения информации — фотография и кино. К ним очень важно добавить разработку методов записи информации на магнитные носители (магнитные ленты, диски).

С разработкой первых ЭВМ принято связывать возникновение информатики как науки, начало ее ***истории***. Для такой привязки имеется несколько причин. Во-первых, сам термин «информатика» появился благодаря развитию вычислительной техники, и поначалу под ним понималась наука о вычислениях (первые ЭВМ большей частью использовались для проведения числовых расчетов). Во-вторых, выделению информатики в отдельную науку способствовало такое важное свойство современной вычислительной техники, как единая форма представления обрабатываемой и хранимой информации. Вся информация, вне зависимости от ее вида, хранится и обрабатывается на ЭВМ в двоичной форме. Так получилось, что компьютер в одной системе объединил хранение и обработку числовой, текстовой (символьной) и аудиовизуальной (звук, изображение) информации. В этом состояла инициирующая роль вычислительной техники при возникновении и оформлении новой науки.

На сегодняшний день *информатика представляет собой комплексную научно-техническую дисциплину*. Под этим названием объединен довольно обширный комплекс наук, таких, как кибернетика, системотехника, программирование, моделирование

и др. Каждая из них занимается изучением одного из аспектов понятия информатики. Учеными прилагаются интенсивные усилия по сближению наук, составляющих информатику. Однако процесс их сближения идет довольно медленно, и создание единой и всеохватывающей науки об информации представляется делом будущего.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ТЕМЕ 1.1

1. Дайте определение понятия «информация».
2. В чем отличие знаний от данных?
3. Опишите общую схему системы передачи информации.
4. В чем разница между АВМ и ЦВМ?
5. Как вы понимаете смысл дисциплины «информатика»?
6. Что такое АИС?
7. Раскройте понятие «информационная технология».
8. Дайте определение понятия «информационный ресурс».
9. Какие вы знаете виды ИР?
10. Каковы основные особенности ИР?
11. Перечислите и раскройте формы существования ИР.
12. Как вы понимаете «объект» и «предмет» информатики?
13. Сформулируйте определение информатики как науки.

Тема 1.2
ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

Методические указания

Цель изучения данной темы — усвоение основополагающих знаний об информационных процессах. Студенты должны четко представлять основные этапы технологического процесса в информационных системах, суть информационных процессов во всех сферах живой природы, в обществе и технике. Очень важно акцентировать внимание на анализе сути информационных процессов и их значении в повседневной деятельности и жизни человека. Преподаватель должен донести до каждого студента суть понятия «управление», особо подчеркнуть, что управление — чисто информационный процесс. Это чрезвычайно важно, ибо сегодняшние студенты — это будущие управленцы.

Заключительная часть темы 1.2 посвящена процессам информатизации общества. Студенты должны уяснить: будущее человечества — это жизнь в информационном обществе с его свойствами и особенностями. Все члены общества должны владеть компьютером и информационными технологиями, так как альтернативы информатизации не существует!

1.2.1. Этапы технологического процесса в информационных системах

Если информация категория всеобщая, присуща всем историческим периодам, то информатика категория конкретно-историческая, присуща лишь современному и последующим историческим периодам. Ранее, когда не было ИТ, т. е. не было объекта информатики, информация перерабатывалась немашинными, рутинными способами. На базе ЭВМ рождаются ИС, реализующие специальные технологии сбора, переработки, передачи и применения информации.

Понятие технологии вообще включает комплекс научных и инженерных знаний, воплощенных в приемах труда, наборах материальных, технических, энергетических, трудовых факторов производства, способов их соединения для создания продукта или услуги, отвечающих определенным требованиям, стандартам. В таком понимании термин «технология» неразрывно связан с машинизацией производственного или непроизводственного (социального) процесса. Последнее важно подчеркнуть для обозначения точных исторических границ информатики: ее не могло быть в домашинный период обработки и представления знаний, когда не было ИТ. Переработка информации с помощью ЭВМ, выработка новых знаний, соотнесенных с целями пользователей, — функциональное назначение ИТ. Как всякая технология, индустриальная ИТ состоит из этапов.

Информационная технология содержит:

— определенные наборы материальных средств (носители информации, технические средства измерения ее состояний, обработки, передачи и т. п.);

— способы их взаимодействия;

— совокупность определенных методов организации работы специалистов.

Информационная технология решения большого числа разных задач включает следующие важнейшие процедуры, которые могут быть сгруппированы по функционально-временным стадиям:

— сбор и регистрация информации;

— передача ее к месту обработки;

— машинное кодирование данных;

— хранение;

— поиск;

— вычислительная обработка;

— тиражирование информации;

— использование информации, т. е. принятие решений.

Как правило, информация подвергается всем процедурам преобразования, но в ряде случаев некоторые процедуры могут отсутствовать. Последовательность выполнения процедур также бывает различной, а некоторые из них могут повторяться. Состав процедур преобразования и особенности их выполнения во многом зависят от объема и системы автоматизированной обработки информации.

Рассмотрим особенности выполнения основных этапов преобразования информации.

1. Процедура **сбора и регистрация информации** осуществляется по-разному на различных объектах. Наиболее сложен этот этап в автоматизированных управленческих процессах таких промышленных предприятий, фирм и т. п., на которых происходят сбор и регистрация первичной учетной информации, отражающей производственно-хозяйственную деятельность объекта. Особое значение при этом придается достоверности, полноте и своевременности первичной информации.

На предприятии сбор и регистрация информации осуществляются при выполнении различных хозяйственных операций (прием готовой продукции, получение и отпуск материалов и т. п.). Сначала информацию собирают, затем ее фиксируют. Учетные данные могут возникать на рабочих местах (например, в результате подсчета количества обработанных деталей,

прошедших сборку узлов, изделий и т. п.). Для сбора фактической информации производятся измерения, подсчет, взвешивание материальных объектов, получение временных и количественных характеристик работы отдельных исполнителей. Сбор информации, как правило, сопровождается регистрацией, т. е. фиксацией информации в бумажном документе или на машинном носителе.

Запись в первичные документы в основном осуществляется вручную, поэтому процедура сбора и регистрации остается пока наиболее трудоемкой. В условиях автоматизации управления предприятием особое внимание придается использованию технических средств сбора и регистрации информации, ее накопления и передачи по каналам связи в ЭВМ с целью формирования первичного документа.

Необходимость передачи информации для различных объектов обосновывается по-разному. Так, в автоматизированной системе управления (АСУ) предприятием она вызвана тем, что процедура сбора и регистрации информации нередко территориально отделена от процедуры ее обработки. Первая, как правило, осуществляется на рабочих местах, а вторая — в вычислительном центре или на персональной ЭВМ (ПЭВМ).

2. **Передача информации** может производиться различными способами: с помощью курьера, пересылкой по почте, доставкой транспортными средствами, дистанционно по каналам связи. Последний способ сокращает время передачи данных. Для дистанционной передачи информации по каналам связи необходимы специальные технические средства. Некоторые технические средства сбора и регистрации автоматически собирают информацию с датчиков, установленных на рабочих местах, и передают ее в ЭВМ.

Дистанционно может передаваться как первичная информация с мест ее возникновения, так и результатная — в обратном направлении. В этом случае результатная информация отражается на различных устройствах: дисплеях, табло, печатных устройствах. Поступление информации по каналам связи в центр обработки в основном осуществляется двумя способами: на машинном носителе или непосредственно в ЭВМ с помощью специальных программных и аппаратных средств.

Дистанционная передача информации постоянно развивается и совершенствуется. Применение этого способа в многоуровневых межотраслевых системах значительно ускоряет прохождение информации с одного уровня управления на другой и сокращает общее время обработки данных.

3. **Машинное кодирование информации** — это процедура машинного представления (записи) информации на машинных носителях в кодах, принятых в ЭВМ. Такое кодирование производится путем переноса данных первичных документов на магнитные диски, информация с которых затем вводится в ЭВМ для обработки. В процессе записи информации на машинные носители возникает наибольшее количество ошибок, что объясняется трудоемкостью данной операции. Поэтому обязательно выполняются операции контроля записи разными методами на специальных устройствах либо на ЭВМ.

4. **Обработка информации на ЭВМ** производится, как правило, централизованно, а на мини- и макроЭВМ — в местах возникновения первичной информации, где организуются автоматизированные рабочие места специалистов той или иной управленческой службы (отдела материально-технического снабжения и сбыта, отдела главного технолога, конструкторского отдела, бухгалтерии, планового отдела и т. п.).

Автоматизированное рабочее место (АРМ) специалиста включает персональную ЭВМ (ПЭВМ), работающую автономно или в вычислительной сети, и набор программных средств и информационных массивов для решения функциональных задач.

Обработка информации на ПЭВМ, или ПК, начинается при полной готовности всех устройств машины. Оператор или пользователь ПЭВМ руководствуется специальной инструкцией по эксплуатации технических и программных средств.

В начале работы в машины загружаются программа и различные информационные массивы (условно-постоянные, переменные, справочные), каждый из которых сначала, как правило, обрабатывается для получения каких-либо результатных показателей, а затем массивы объединяются для получения сводных показателей.

При обработке информации на ЭВМ выполняются арифметические и логические операции. Арифметические операции обработки данных в ЭВМ включают все виды математиче-

ских действий, обусловленных программой. Логические операции обеспечивают соответствующее упорядочение данных в массивах (первичных, промежуточных, постоянных, переменных), подлежащих дальнейшей арифметической обработке. Значительное место в логических операциях занимают такие виды сортировальных работ, как упорядочение, распределение, подбор, выборка, объединение.

В ходе решения задач на ЭВМ в соответствии с машинной программой формируются результатные сводки, которые печатаются машиной.

5. **Хранение и накопление** информации вызвано необходимостью ее многократного использования, применения постоянной информации и комплектации первичных данных до их обработки. Хранение осуществляется на машинных носителях в виде информационных массивов, где данные располагаются по установленному в процессе проектирования группировочному признаку.

6. **Поиск данных** — это выборка нужных данных из хранимой информации, включая поиск информации, подлежащей корректировке или замене. Процедура поиска выполняется на основе составленного запроса на нужную информацию.

7. **Печать сводок** может сопровождаться процедурой тиражирования, если документ с результатной информацией необходимо представить нескольким пользователям.

8. **Принятие решения** в автоматизированной системе организационного управления, как правило, осуществляется специалистом без применения технических средств, но на основе тщательного анализа результатной информации, полученной на ПЭВМ. Процедура принятия решения осложняется тем, что специалисту приходится искать из множества допустимых решений наиболее приемлемое, сводящее к минимуму потери ресурсов (временных, трудовых, материальных и т. д.). Благодаря применению ПЭВМ и терминальных устройств повышается степень аналитичности обрабатываемых сведений, а также обеспечивается постепенный переход к автоматизации выработки оптимальных решений в процессе диалога пользователя с вычислительной системой. Математическая теория принятия решений (ТПР) строится на основе теории игр и исследования операций.

1.2.2. Информационные процессы в живой природе, обществе и технике

Наследственность. Человека всегда интересовало, почему он похож на своих родителей и других родственников — братьев, сестер, дедушек, бабушек, причем не только внешне, но и характером, привычками. Все это объясняется наследственностью. Людям хотелось знать, как наследственная информация передается из поколения в поколение, но эта тайна природы была раскрыта только в середине XX в.

Информация проявляется и в растительном, и в животном мире в виде передачи потомству наследственных признаков в ряде поколений. Наследственность передается самовоспроизведением генов, находящихся в хромосомах ядра клетки, и вместе с изменчивостью обеспечивает постоянство и многообразие форм жизни. Однако генетический аппарат нужен не только для воспроизведения потомства. В течение всей жизни организма клеточный состав непрерывно меняется, и именно генетический аппарат следит за тем, чтобы организм оставался самим собой.

Животные и люди с точки зрения кибернетики — науки об управлении, связи и переработке информации — самоорганизующиеся и самообучающиеся системы. Общим в работе нервной системы живых существ и компьютера является получение, запоминание и переработка информации. Но в том, как они это делают, есть существенные различия.

Компьютер отличается от мозга животного и человека прежде всего бóльшим быстродействием. При этом все вычисления, в соответствии с архитектурой Дж. фон Неймана, компьютер производит только последовательно, шаг за шагом.

Мозг человека обладает гораздо меньшим быстродействием, однако огромное количество (несколько миллиардов) нервных клеток — нейронов — способно производить параллельные «вычисления». Необходимым условием работы мозга является наличие памяти, как оперативной, так и долговременной.

Рассмотрим более подробно, как происходит передача наследственных признаков потомству. Эти вопросы изучает *генетика* — наука о законах наследственности и изменчивости организмов.

Клетка состоит из ядра, окруженного двухслойной мембраной, и остального содержимого — протоплазмы, позднее названной цитоплазмой. В каждой клетке есть участки — органеллы, отвечающие за различные процессы. Для реализации процессов органеллам нужны белки. Синтезом всех белков занимаются мельчайшие структуры в цитоплазме — рибосомы.

В XIX в. биологи обратили внимание на хромосомы, которые помещаются в ядре клетки. Оказалось, что каждому виду растений или животных свойственно определенное число хромосом. В процессе деления клеток хромосомы удваиваются, и каждая дочерняя клетка снова имеет полное их число. В результате был сделан вывод, что передача потомству наследственных признаков связана именно с хромосомами.

Основы генетики заложил Грегор Мендель — основоположник учения о наследственности, названного впоследствии менделизмом. В 1854—1863 гг. он провел эксперименты, доказавшие существование индивидуальных наследственных факторов — генов.

Хромосомную теорию наследственности в 1910-х гг. обосновал американский биолог Томас Морган (1866—1945) — один из основоположников генетики, лауреат Нобелевской премии 1933 г. Ученые школы Моргана доказали, что наследственные факторы — это гены, размещающиеся в хромосомах; они расположены линейно и сцеплены между собой, а во время созревания половых клеток могут разъединяться. Школе Моргана удалось установить порядок расположения генов в хромосомах некоторых животных и растений — мухи дрозофилы, кур, кукурузы, ряда бактерий. Муха дрозофила стала излюбленным подопытным объектом генетиков из-за ее способности быстро давать потомство, что удобно при изучении наследственности.

В 1920—1930 гг. важный вклад в развитие генетики внесли выдающиеся отечественные биологи Н.К. Кольцов (1872—1940) и Н.И. Вавилов (1887—1943). Именно Н.К. Кольцов в 1928 г. высказал предположение, что хромосомы — это гигантские молекулы.

Материальную природу генов удалось раскрыть Джеймсу Уотсону (р. 1928) и Френсису Крику (1916—2004). В 1953 г. они предложили структурную модель так называемой двойной спирали дезоксирибонуклеиновой кислоты (ДНК) — высокополи-

мерного природного соединения, которое содержится в ядрах клеток живых организмов. ДНК вместе с белками образует вещество хромосом. Структурная модель объясняла, каким образом генетическая информация записывается в молекулах ДНК, и позволила высказать предположение о химических механизмах самовоспроизведения этих молекул. Именно ДНК является носителем генетической информации. Отдельные ее участки соответствуют определенным генам. Молекула ДНК состоит из двух цепей, закрученных одна вокруг другой в спираль, поэтому она называется **двойной спиралью**. Эти цепи построены из большого числа мономеров четырех типов — нуклеотидов. Сочетание нуклеотидов, стоящих рядом в цепи ДНК, составляет генетический код. Нарушение их последовательности приводит к мутациям — наследственным изменениям. ДНК точно воспроизводится при делении клеток. Это обеспечивает передачу наследственных признаков в ряду поколений отдельных клеток и целых организмов.

Основой жизни растений и животных являются белки. Они представляют собой сложные органические соединения, состоящие из различного числа аминокислот. В организме каждого растения или животного есть множество белков, выполняющих различные функции. Мышечная ткань содержит миозин, эритроциты крови — гемоглобин — переносчик кислорода, поджелудочная железа — инсулин. Огромную роль играют белки-ферменты. Синтез всех белков происходит в клетках, а их специфические особенности определяются генетической информацией, хранящейся в ДНК хромосом.

Синтез белков в клетках осуществляется под контролем ДНК и с участием молекул трех различных видов рибонуклеиновых кислот (РНК): транспортной, информационной и рибосомной. Молекулы РНК меньше молекул ДНК и состоят из меньшего количества нуклеотидов.

Учеными-генетиками был изучен механизм поступления информации, закодированной в генах, из ядра клетки в цитоплазму, где происходит синтез белков, необходимых для жизни организма. Для выполнения своих функций органеллам нужны различные белки. Американский молекулярный биолог Понтер Блобел открыл основные принципы управления подбором белков, необходимых для каждой органеллы. Он обнаружил

адресную последовательность в молекуле белка: своеобразную «этикетку с адресом доставки», которая обычно расположена на кончике молекулы. «Прочитав» эту «этикетку», части клетки принимают нужный им белок.

Белковые молекулы — главное вещество и главная действующая сила организма животного и человека. Они выполняют различные функции. Одни белки работают в качестве транспорта. К примеру, белок гемоглобин в легких загружается кислородом и разносит его по всему организму с потоком крови. Из других белков собраны ткани и органы. Третьи разбирают на детали химическое сырье, получаемое с пищей, четвертые помогают собирать из этих деталей новые белки или более простые.

Итак, гены представляют собой участки длинной молекулы ДНК. В каждой такой молекуле могут заключаться многие тысячи генов, каждый из которых несет определенную наследственную информацию, например о росте, или цвете глаз, или форме носа.

Д. Уотсон и Ф. Крик стали основателями современной молекулярной генетики. Их открытие структуры ДНК (двойной спирали) наравне с теорией относительности, открытием строения атома и появлением компьютеров можно считать одним из выдающихся достижений науки XX в. За свое открытие они совместно с М. Уилкинсом в 1962 г. были удостоены Нобелевской премии.

Успехи молекулярной генетики привели к созданию генетической, или генной, инженерии. Это раздел молекулярной биологии, задача которого заключается в целенаправленном конструировании новых, еще не существовавших в природе сочетаний генов с помощью методов биохимии и генетики. Методы генной инженерии основаны на извлечении из клеток какого-нибудь организма одного гена или целой их группы. Гены соединяют с определенными молекулами нуклеиновых кислот, затем полученные гибридные молекулы внедряют в клетки другого организма.

Методы генной инженерии в наши дни обеспечили поразительные успехи в сельском хозяйстве, биотехнологии, медицине, производстве новых лекарств. По существу, биотехнология является одним из видов информационных технологий. Специфика ее состоит в том, что информация «зашита» в генах.

Развитие генной инженерии привело к так называемой зеленой революции, т. е. к резкому увеличению урожайности сельскохозяйственных культур и выведению новых продуктивных пород животных. Появились инсектициды — химические препараты для борьбы с насекомыми-вредителями.

Совокупность всех генов данного биологического вида составляет его гено́м. Успехи современной генетики дают возможность полностью расшифровать геномы простейших организмов. Но это невероятно сложная задача, для решения которой нужны не только методы генетики, но и мощные компьютеры. Уже удалось расшифровать геном дождевого червя. Важнейшей задачей всей мировой науки начала XXI в. является полная расшифровка генома человека. Эту задачу, уже решаемую в наши дни, ученые надеются полностью решить через несколько лет. Расшифровка генома человека будет иметь основополагающее значение для медицины и позволит лечить самые страшные и сегодня неизлечимые болезни методами генетики.

Первый этап этого международного проекта (стоимостью несколько миллиардов долларов) был закончен к началу 2001 г.: прочитана последовательность нуклеотидов ДНК генома человека. Такой процесс назван секвенированием (похоже на запись телеграммы на неизвестном языке). Теперь начинается следующий, не менее сложный этап: предстоит расшифровать смысл этой «телеграммы», расчленить ее на «слова» — отдельные гены. Этим занимается новая наука — *биоинформатика*. Ее задача — компьютерный анализ всех нуклеотидных последовательностей ДНК генома человека и других живых организмов (а их насчитывается несколько миллиардов). Цель биоинформатики — понять, где начинается и где заканчивается каждый ген, отвечающий за синтез одного конкретного белка. Длина генома человека (он состоит из 3 млрд нуклеотидных пар) достигает 2 м. Предполагается, что геном содержит 30—40 тыс. генов.

Биоинформатика дает возможность математическими методами предсказать ген с вероятностью до 85 %. Но это только исходная информация, по которой экспериментально проверяют наличие гена. Для ее получения предполагаемый ген «вырезают» из ДНК и проверяют, действительно ли этот фрагмент отвечает за синтез определенного белка.

Цель исследований — найти и опознать «больные» гены, а это прямой путь к диагностике и лечению, которым занимается *медицинская геномика*. С ее помощью можно быстрее создавать новые лекарства. За весь XX в. создано около 500 лекарств. После расшифровки генома человека такое же число новых лекарственных средств можно будет создать менее чем за 10 лет!

Клонирование. Телесные, соматические, клетки, как и половые, снабжены полной генетической информацией о развитии организма. Но каждая из этих клеток снабжена еще и информацией о своей будущей «специализации»: из одних соматических клеток «строится» печень, из других — кости, из третьих — нервная система и т. д. Если от соматической клетки получить потомство, то новый организм будет полной генетической копией своего родителя. Такой процесс называется **клонированием**. Однако клонировать можно все свойства, кроме интеллекта, ума, неповторимых черт личности данного человека.

Для клонирования из яйцеклетки удаляют ядро, содержащее половинный набор хромосом. Вместо него помещают ядро соматической клетки с двойным набором хромосом.

В 1997 г. появились сообщения о том, что осуществлено клонирование животных — овец и коров, т. е. из соматических клеток получены их генетические копии. Это вызвало большой шум и в научном мире, и в средствах массовой информации. Началось обсуждение возможности клонирования человека. Одни ученые предложили запретить подобные опыты, дабы не создать тирана вроде Гитлера; другие к этой возможности отнеслись спокойно, ссылаясь на то, что совсем недавно раздавались призывы запретить искусственное оплодотворение, а теперь это стало повседневной практикой и никого уже не удивляет.

Конечно, клонирование человека связано с этическими, социальными и даже философскими проблемами, ведь речь идет о биологическом бессмертии человека. Кроме того, при клонировании существует большая вероятность появления уродов. Пока на один удачный опыт приходится около 300 неудачных. Если при клонировании животных это не так опасно, то при клонировании человека неприемлемо.

Следует сказать, что, хотя способности и характер человека определяются главным образом его индивидуальным генетиче-

ским кодом, многое в его поведении и судьбе зависит от среды, в которой он растет, воспитывается и живет. А ее воссоздать невозможно. Так что полностью повторить Эйнштейна или Чехова нереально даже теоретически, но создать людей со способностями этих личностей представляется возможным. В большинстве развитых стран клонирование человека или запрещено, или на него введен временный мораторий. В 2001 г. такой мораторий введен на пять лет и в России.

Нервная система и память. Память — это способность сохранять и воспроизводить следы полученных впечатлений (зрительных, слуховых, обонятельных, осязательных, вкусовых), т. е. сохранять и воспроизводить некую информацию. Именно благодаря памяти человек живет не только в настоящем, но и в прошлом, и в будущем; она дает возможность планировать поведение в соответствии с заложенными инстинктами, опытом, знаниями и окружающей обстановкой, т. е. принимать решения. Память нужна человеку в течение всей его жизни.

Необходимо отметить коренное отличие генетической информации от индивидуальной памяти. Оно состоит в том, что генетическая информация передается по наследству от организма к организму, тогда как индивидуальная память не наследуется, а приобретается человеком в процессе обучения и воспитания.

Источником возбуждения для нас является окружающий мир, его предметы и события, а каждое возбуждение оставляет свой след в нервной системе. Нервная система состоит из особых клеток — нейронов. Каждый нейрон имеет тело и отростки — несколько коротких (дендриты) и один длинный (аксон). Нейроны в зависимости от формы бывают различных типов — пирамидальные, звездчатые, веретенные. С помощью дендритов и аксонов нейроны образуют общую нейронную сеть, способную воспринимать и передавать информацию, например сигналы из внешнего мира. Передаваемые сообщения — сигналы — представляют собой последовательные импульсы, проходящие по аксонам и дендритам центральной нервной системы от одного нейрона к другому.

Какова же скорость передачи импульсов? Она различна для разных нервов: в двигательных нервах у человека она составляет 20—60 м/с, а болевое ощущение передается со скоростью

всего 1—30 м/с. В седалищном нерве лягушки эта скорость составляет 27 м/с. Максимальная частота передачи импульсов по нерву составляет 500 импульсов в секунду. Если сравнивать эту скорость со скоростью передачи сигналов в современном компьютере, близкой к скорости света (300 000 км/с), то она кажется ничтожно малой.

Область контакта между аксоном и нейроном, которому адресуются импульсы, называется **синапсом.** Число синапсов у различных нейронов различно. Синапс обладает свойством одностороннего проведения импульсов, что обеспечивает определенный порядок распространения возбуждения в нервной системе. Еще одно важнейшее свойство синапса — его пороговый характер. Синапс не реагирует на одиночные возбуждения. Только после накопления возбуждений выше определенного порога он передает их дальше по цепи нейронов, а через некоторый промежуток времени осуществляет синаптическую задержку импульса. Сами импульсы имеют биоэлектрическую природу, а пороговый характер синапсов и синаптическая задержка — биохимическую природу. В нервной системе происходят два противоположных активных процесса — возбуждение и торможение, являющиеся основными законами ее действия на всех уровнях. При этом возбуждение создает основной тон, а торможение его корректирует. Между возбуждением и торможением происходит борьба, исход которой определяет судьбу каждого сообщения: будет оно передано дальше или нет.

Общее число нейронов в мозге человека составляет гигантскую цифру: десятки миллиардов, а связей между ними — синапсов — на два порядка больше. Все вместе они составляют нейронную сеть.

Нейроны получают раздражение от рецепторов, воспринимающих внешние раздражения — зрительные, слуховые, обонятельные и осязательные. Рецепторы (от лат. receptor — принимающий) — это окончания чувствительных нервных волокон или специализированные клетки (сетчатки глаза, внутреннего уха и др.). Внешние раздражения преобразуются рецепторами в нервное возбуждение, передаваемое в центральную нервную систему.

Контроль же за собственно телом осуществляют проприорецепторы. Они находятся во всех связках, суставах, мышцах и

сухожилиях. Благодаря сигналам, передаваемым центральной нервной системе, мы можем в любой момент ощущать свое тело и контролировать действие двигательных органов. Мозг, получая информацию от рецепторов, посылает через нервную систему команды мышцам. Они при этом сокращаются, вызывая движение рук, ног, шеи или глаз. Это прямые связи. Мышцы с помощью рецепторов сообщают мозгу о своем положении. Это обратная связь, т. е. сообщение о результатах действия.

Реакция организма, осуществляемая по командам центральной нервной системы в ответ на сигналы от рецепторов, — это рефлекс. Вся деятельность организма является комбинацией различных рефлексов. Они делятся на безусловные (врожденные, заложенные генетически) и условные (вырабатываемые в течение жизни). Пример безусловного рефлекса — реакция на боль, например от прикосновения к горячему. При этом организм принимает решение автоматически, без участия сознания, с помощью спинного мозга. Сигнал от рецептора поступает, минуя головной мозг, непосредственно в спинной мозг, который немедленно посылает команду мышцам руки или ноги отдернуть руку или ногу от горячего предмета. Так как сигнал от рецептора к мышцам проходит по короткой цепи, то и выполнение команды происходит с максимально возможной скоростью. Спинной мозг управляет и такими сложными безусловными рефлексами, как дыхание, биение сердца, координация движений при ходьбе, деятельность органов пищеварения. Обо всех этих процессах мы не думаем, в течение жизни они совершаются автоматически, помимо нашего сознания. Так, например, при изменении нагрузки меняется пульс — при ходьбе, беге, поднятии тяжестей. Все эти безусловные рефлексы являются формой памяти, заложенной в нас генетически, при рождении.

У животных и человека существуют еще инстинкты (от лат. instinctus — побуждение) — сложные безусловные рефлексы (пищевой, оборонительный, половой и др.). Они представляют собой совокупность сложных врожденных реакций (актов поведения) организма, возникающих в ответ на внешние или внутренние раздражения. Инстинкты человека, в отличие от животных, контролируются его сознанием.

В течение жизни мы приобретаем множество условных рефлексов — еще один вид памяти, представляющий собой

реакцию на определенные повторяющиеся события или время. Так, например, чувство голода может возникать у нас во время работы при приближении обеденного перерыва.

Безусловный рефлекс — реакция на боль, возникшая у ребенка при первом прикосновении, например, к раскаленной печке, — превращается затем в условный рефлекс. Больше ребенок никогда к раскаленной печке не прикоснется. Условный рефлекс — это своеобразная память, которую живой организм приобретает и накапливает в течение своей жизни.

Научившийся ходить, плавать, ездить на велосипеде, играть на музыкальных инструментах человек сохраняет это умение на всю жизнь. А как мышечный аппарат выполняет эти навыки? Исследованием этих механизмов занимается наука *биомеханика*, или *физиология активности*. Создателем биомеханики является российский ученый, нейро- и психофизиолог Н.А. Бернштейн (1896—1966).

Биомеханика изучает механические свойства живых тканей, органов и организма в целом, а также происхождение в них механического явления (при движениях, дыхании и т. д.). Ее еще называют наукой о рычагах человеческого тела. Н.А. Бернштейну удалось установить, что в сложнейшем «концерте» — управлении множеством мышц при выполнении самых различных действий (например, при игре на рояле) — участвуют все уровни головного мозга.

Биомеханика имеет большое значение при изучении трудовой, музыкальной, спортивной деятельности. Она помогает строить роботов, копирующих движения человека и животных.

Профессиональная, **оперативная и долговременная память**. У взрослых людей появляется еще и ***профессиональная память***. Особенно хорошо это видно на примере врачей, помнящих невероятный объем информации: сотни и тысячи симптомов болезней, названия лекарств, имена пациентов; на примере музыкантов, помнящих огромное число музыкальных произведений, и шахматистов, помнящих множество шахматных дебютов, целых партий и способных играть сеансы одновременной игры «вслепую» с большим числом партнеров. Правда, у каждого профессионала вырабатываются свои приемы запоминания, связанные с хорошими знаниями и интересом к своей деятельности.

Память связана с высшей нервной деятельностью человека. Она существует двух родов: кратковременная (или оперативная) и долговременная.

Оперативная память играет вспомогательную роль и используется, например, для арифметических расчетов. При умножении чисел она дает возможность запоминать промежуточные результаты («два пишем, четыре в уме»). Но после выполнения всей операции промежуточные результаты становятся ненужными и только «засоряют» память, поэтому они быстро забываются. Глубину кратковременной памяти психологи измеряют количеством цифр или слогов, которые человек может запомнить с первого раза. При этом ему предлагают запомнить их в полном беспорядке, без какого-либо смысла. Оказалось, что емкость оперативной памяти составляет всего семь-восемь слогов и у каждого человека эта емкость постоянна. При изучении оперативной (механической) памяти было замечено, что лучше всего запоминаются первые и последние цифры или слоги.

Долговременная память работает совсем не так, как оперативная, поскольку многие вещи нам нужно помнить практически всю жизнь. Например, таблицу умножения мы заучиваем в первом классе школы и помним ее в течение всей жизни, в отличие от результатов каких-то текущих арифметических вычислений. Таким же образом мы запоминаем алфавит родного или иностранного языка, грамматические правила и великое множество самой различной информации.

Способность к обучению и запоминанию новой информации на многие десятки лет особенно сильно проявляется у детей в возрасте до трех лет. Именно в этот период ребенок получает не менее половины информации, которую он запоминает за всю дальнейшую жизнь. Достаточно посмотреть, как быстро (за первые полтора-два года) ребенок начинает говорить на родном языке. Да и изучение иностранных языков (как и любое другое обучение) дается детям гораздо легче, чем взрослым.

Ослабление памяти в старости начинается с ослабления оперативной памяти — именно так проявляется склероз. Сохранению долговременной памяти в пожилом возрасте способствует интеллектуальная работа, в особенности при занятии любимым делом. Долговременная память у пожилых людей сохраняется гораздо дольше, чем оперативная.

В чем же состоит коренное отличие долговременной памяти от оперативной? Оперативная память в основном механическая. Правда, запоминание можно сделать смысловым. Простейшим приемом является так называемый «узелок на память», т. е. установление некоторых смысловых связей.

Определенная часть запомненной информации отбирается и передается на хранение в долговременную память.

Предметы и понятия хранятся в нашей памяти в виде образов, имеющих расплывчатый, обобщенный характер. Например, образ стола, стула или шкафа не имеет каких-то конкретных деталей, а носит схематический, условный характер, при этом обладая всеми основными признаками предмета. То же касается и природных явлений, например дождя, снега, шторма на море. В памяти человека хранится огромное количество таких образов и понятий.

Какое же количество информации может запомнить человек за всю жизнь? Он способен обработать около 20 бит информации в секунду, т. е. оценить около миллиона различных возможностей за ту же секунду. В день за 14 часов можно обработать 18 млрд бит. Для хранения такой информации достаточно одной тысячной части всех нервных клеток мозга. Человек способен вспоминать нужную информацию за десятые доли секунды, для чего требуется скорость поиска около 50 млрд бит в секунду. Обработка такого гигантского количества информации обеспечивается параллельной работой нервных структур, в отличие от компьютера, где все операции происходят только последовательно.

Общеизвестна догма: нервные клетки не восстанавливаются. Недавние исследования ученых показали, что это неверно. В течение жизни человека в мозговой ткани происходит образование новых нейронов. Предполагают, что этот процесс связан с действием механизма долговременной памяти. И это дает врачам надежду на возможность лечения болезни Альцгеймера — старческого слабоумия, угрожающего каждому из нас в возрасте 85 лет и старше.

Искусственный интеллект. Одно из самых загадочных явлений — распознавание человеком образов. Ведь мы можем почти мгновенно узнать в толпе знакомого человека или его голос по телефону.

Человек распознает образы в течение всей жизни: он сравнивает увиденные и услышанные образы с хранящимися в его памяти и опознает знакомые. В соответствии с этим он принимает решения о своих действиях. Этот процесс представляет собой одну из самых сложных загадок человеческого мозга, на исследование которой уже потрачены многие годы и значительные научные силы. Знать механизм распознавания образов очень важно для работ по созданию искусственного интеллекта и автономных роботов.

В начале 1950-х гг. Алан Тьюринг (1912—1954) сформулировал такой тест: компьютер можно считать разумным, если он способен заставить нас при общении с ним поверить, что мы имеем дело не с машиной, а с человеком.

По мере продвижения работ в области искусственного интеллекта появляются новые его определения. Одно из самых полных принадлежит крупнейшему ученому в этой области Марвину Мински (США): искусственный интеллект — «это наука по созданию машин, которые могут делать то, что им позволяет делать уровень человеческого интеллекта».

Начало работ по созданию машин, обладающих искусственным интеллектом, стимулировал Норберт Винер (1894—1964) своей знаменитой книгой «Кибернетика, или Управление и связь в животном и машине», появившейся в 1948 г. Он выдвинул принцип обратной связи, заключающийся в использовании информации, поступающей из окружающей среды, для изменения поведения машины. В своей книге Винер доказывал, что благодаря обратной связи все живое приспосабливается к окружающей среде и добивается своей цели.

В ходе исследований Н. Винер увидел глубокую аналогию между поведением машин и живых организмов в их приспособлении к изменениям в окружающей среде с помощью универсального механизма обратной связи — общего для техники и живой природы.

В технике примеров действия обратной связи множество, например действия водителя при управлении автомобилем. Водитель держит руки на руле и следит за дорогой. Как только зрение подсказывает ему, что машина отклоняется от прямого пути, он корректирует ее движение, поворачивая руль на больший или меньший угол (в зависимости от величины отклоне-

ния от прямого пути) до тех пор, пока не восстановится прямолинейное движение. И такой процесс продолжается в течение всего времени движения. По такому же принципу в технике работает множество автоматических регуляторов температуры, давления, влажности и других параметров.

Н. Винер обратил внимание на важнейшую роль обратной связи для поддержания у живых организмов гомеостаза — механизма поддержания устойчивости основных физиологических функций организма.

Между нервной системой и вычислительной машиной Н. Винер установил следующую аналогию: важнейшей функцией обеих является память, «то есть способность сохранять результаты прежних действий для использования в будущем». Он отметил, что существует память, необходимая для выполнения текущих процессов, например умножения. При этом промежуточные результаты не имеют ценности после завершения процесса и должны уничтожаться. Такая память должна позволять быструю запись, считывание и стирание. Но существует и память, предназначенная служить частью архива (или постоянной записи) машины или мозга и составлять основу будущего поведения машины.

В то же время Н. Винер увидел различия между поведением машины и мозга. Машина предназначена для выполнения многих последовательных программ, ее память может быть очищена при переходе от одной программы к другой, а мозг в нормальных условиях никогда не очищается от своих прошлых записей. Поэтому мозг не является полным подобием вычислительной машины.

Говоря о памяти, Н. Винер отмечает, что «хороший способ построить кратковременную память — это заставить последовательность импульсов циркулировать по замкнутой цепи до тех пор, пока эта цепь не будет очищена внешним воздействием». Весьма правдоподобно, что это и происходит в нашем мозгу при хранении импульсов, относящихся к так называемому мнимому настоящему. Этот способ был воспроизведен в вычислительных машинах.

Таким образом, Винер указал способ построения схем оперативной (или — до введенного им термина — кратковременной) памяти с помощью обратной связи.

Приведем одно важное для понимания информационных процессов высказывание Н. Винера из его книги «Кибернетика и общество»: «Информация — это обозначение содержания, полученного из внешнего мира в процессе нашего приспособления к нему и приспосабливания к нему наших чувств. Процесс получения и использования информации является процессом нашего приспособления к случайностям внешней среды и нашей жизнедеятельности в этой среде... сообщение и управление точно так же связаны с самой сущностью человеческого существования, как и с жизнью человека в обществе».

Здесь уместно напомнить определение понятия «жизнь», которое обычно дается в современных энциклопедиях: «Жизнь — одна из форм существования материи. Живые организмы отличаются от неживых объектов обменом веществ, раздражимостью, способностью к росту, развитию, активной регуляции своего состава и функций к различным формам движения, приспособляемостью к среде».

Современные роботы уже могут приспосабливаться к изменениям среды обитания и даже способны производить себе подобных. Пока от живых организмов их отделяет отсутствие обмена веществ со средой обитания.

В 1943 г. нейрофизиолог Уоррен Маккаллох и математик Уолтер Питтс разработали теорию деятельности головного мозга. Основываясь на результатах изучения нейронов, полученных Маккаллохом, они предложили гипотезу: нейроны можно упрощенно рассматривать как устройства, работающие в двоичном коде. На основе этой гипотезы они построили схему сети электронных «нейронов», способную выполнять любые числовые и логические операции. Конечную цель своих исследований Маккаллох и Питтс видели в создании «адаптивной сети», «самоорганизующейся системы» или «обучающейся машины». Эти устройства должны уметь следить за окружающей средой и с помощью обратной связи изменять свое поведение.

Исследования в области искусственного интеллекта идут уже более полувека. За эти годы пройден огромный путь, достигнуты значительные результаты в отдельных областях, например в области стратегических игр.

Как уже говорилось, человек во время всей своей жизни должен принимать решения в соответствии с изменениями

в окружающей среде. То же самое относится и к играм. При этом происходит распознавание образов, характеризующих окружающую среду. Затем производится их анализ и классификация. Для этого мозг сравнивает новые образы с теми, которые известны человеку из прошлого и хранятся в его долговременной памяти. Так и происходит распознавание. Но механизм распознавания является одной из самых сложных задач в области искусственного интеллекта. При решении подобных задач важно изучение механизма извлечения знаний из долговременной памяти. У человека этот процесс происходит с помощью ассоциаций. Исследователи искусственного интеллекта пытаются создать нечто похожее. При решении задач искусственного интеллекта приходится перебирать огромное число вариантов. При этом возможны три типа действий: случайный поиск, полный перебор и так называемый эвристический поиск.

Любое действие по поиску знаний заставляет перебирать *дерево решений*. Его называют так потому, что с каждым шагом поиск «ветвится» на новые варианты. При *случайном поиске* никакого метода поиска нет, все делается по принципу «если повезет». Но вероятность такого везения ничтожно мала, поэтому эффективность этого пути близка к нулю.

Полный перебор по заранее намеченному плану, безусловно, ведет к цели, но число вариантов может быть столь огромно, что время поиска может приближаться к бесконечности. Ярким примером такого перебора является игра в шахматы. Число возможных вариантов позиций в ней так велико, что для их полного перебора не хватит не только мощности самых современных суперкомпьютеров, но и целой жизни. Поэтому шахматисты ограничиваются перебором вариантов только на несколько ходов вперед. Чем глубже перебор, тем лучше играет шахматист. Ведь при этом он перебирает в памяти не только возможные варианты, но и множество уже встречавшихся комбинаций в партиях других шахматистов, известных данному игроку.

Но не только в этом заключается сила игрока. Каждый из шахматистов вырабатывает свои приемы, ведущие к правильной оценке каждой позиции и партии в целом. Творческий подход моделируется при эвристическом поиске.

При *эвристическом поиске* в каждой его точке применяются эвристики — правила, облегчающие результативность каж-

дого варианта с точки зрения скорейшего достижения цели. Слово «эвристика» взято по аналогии со словом «эврика» (греч. héurēka — я нашел). Этот метод оказался плодотворным и применяется в современных программах для игры в шахматы, все более усиливая класс игры компьютера. Эвристики являются своеобразной заменой ассоциаций для компьютера.

В 1997 г. компьютерная шахматная программа фирмы IBM выиграла матч у чемпиона мира Гарри Каспарова. Газеты поместили фотографию склонившегося над доской великого шахматиста с трагическим выражением лица. Это событие подавалось чуть ли не как победа робота над человеческим интеллектом. Однако на самом деле это как раз победа интеллекта человека. Ведь это человек создал замечательную программу, играющую наравне с гроссмейстером. Осуществилась мечта Михаила Ботвинника, который одним из первых начал работу по созданию компьютерных шахматных программ. В 2003 г. Каспаров сыграл вничью с усовершенствованным компьютером. Подобные программы являются важной частью работ по созданию искусственного интеллекта, т. е. способности робота воспринимать и оценивать обстановку, а затем принимать самостоятельные решения в меняющихся условиях.

Эта способность нужна, например, роботу-тележке, который сейчас обследует поверхность Марса. При возникновении препятствий робот сможет, не запрашивая команды с Земли (а ждать их приходится долго), самостоятельно принять решение о дальнейшем движении.

Корпорация Sony создала робот-игрушку: «щенка» по имени AIBO ERS-210 (сокращение слов artifical intelligence — искусственный интеллект и robot — по-японски «товарищ»). Первым серийным домашним роботом со способностью распознавания голоса управляет микропроцессор, не уступающий по мощности процессору персонального компьютера. Оперативная память робота составляет 16 Мбайт. Голова, ноги, уши и хвост у «щенка» подвижные.

Благодаря своему программному обеспечению — операционной системе Sony Aperios, робот-«щенок» обладает эмоциями (он может «радоваться», «сердиться»), инстинктами, способностью обучаться и «взрослеть», а благодаря датчику давления, закрепленному на его голове, AIBO реагирует на дру-

жеское поглаживание или шлепки в наказание за «проступки». Дальномер не позволяет ему натыкаться на стены. Светочувствительные глаза робота загораются зеленым цветом, если он «радуется», и красным, если он «сердится». Встроенной в «лоб» камерой CCD робот может делать фотоснимки, которые позволяют вам увидеть то, что «видел» он. При включении AIBO поднимается на ноги, как щенок. Он может выполнять различные трюки, если предоставлен сам себе, или следовать командам с пульта дистанционного управления. AIBO реагирует на розовый цвет и меняет направление движения, если двигать перед ним мячик розового цвета.

По прогнозам ученых, в этом веке появятся роботы, которые смогут соперничать по своим способностям с человеком. Пока интеллектуальные возможности современных персональных компьютеров находятся на уровне паука, но через 20 лет они сравняются с млекопитающими, а через 40 лет — с человеком. Компьютеры будут способны производить себе подобных и самостоятельно развиваться. А самые смелые футурологи надеются, что с использованием биологических технологий интеллект человека сольется с искусственным.

1.2.3. Информационная деятельность человека

Информация в медицине. Долгие века при определении причины болезни врач мог доверять только своим рукам, глазам и ушам. Первыми приборами, которые стали помогать врачу при осмотре пациентов, были стеклянный ртутный термометр для определения температуры тела, секундомер для подсчета пульса и деревянная слуховая трубка — стетоскоп — для прослушивания сердца, изобретенная французским врачом Рене Ланно в 1819 г. Позднее стетоскоп сменил фонендоскоп с чувствительной мембраной, камера под которой соединена с двумя гибкими трубками. Затем к этому прибавились химические анализы состава крови и мочи.

В 1860 г. итальянский врач Риваччи придумал простой и удобный метод измерения артериального давления. Он основан на измерении внешнего давления, которое нужно для полного пережатия артерии. Для этого на руку выше локтя накла-

дывают полую резиновую манжету и соединяют ее с резиновой грушей и манометром (ртутным или стрелочным). С помощью груши в манжету закачивают воздух и одновременно следят за пульсом на артерии предплечья (у локтевого сгиба) и за показаниями манометра. Давление воздуха увеличивают до тех пор, пока не исчезнет пульс, т. е. пока артерия не будет полностью пережата. Измеренное в этот момент давление воздуха в манжете соответствует систолическому давлению. В 1905 г. русский врач Н.С. Коротков усовершенствовал метод Ривароччи. Он предложил прослушивать пульс фонендоскопом. Это позволило измерять не только систолическое, но и диастолическое давление крови (т. е. соответственно при сокращении и расслаблении сердечной мышцы). Современные автоматические цифровые тонометры оснащены миниатюрным воздушным насосом и датчиком давления в манжете. При измерении давления таким аппаратом резиновая груша и фонендоскоп не нужны. Надо только надеть манжету и нажать на кнопку аппарата. Он проделает весь цикл измерения и покажет цифрами на дисплее величины систолического (верхнего) и диастолического (нижнего) давления, частоту пульса. Выпускаются даже тонометры, манжета которых надевается на запястье или на палец, но они, хотя и удобнее, не дают такой же точности измерения.

Открытие Вильгельмом Рентгеном (1845—1923) Х-лучей, названных его именем, дало врачам возможность «заглянуть» внутрь тела человека, не повредив его. Рентгеновское обследование позволило увидеть теневое изображение костей и внутренних органов. Появление рентгеновского аппарата вызвало к жизни новую область медицины — рентгенологию, изучающую применение рентгеновского излучения для исследования строения и функций органов и систем организма человека.

С середины XX в. началось применение электрокардиографии — метода исследования сердечной мышцы, основанного на регистрации биоэлектрических потенциалов работающего сердца. Записанная на движущейся бумажной ленте или фотопленке электрокардиографа кривая линия электрокардиограммы (ЭКГ) используется для диагностики заболеваний сердца.

Для исследования биоэлектрической активности головного мозга служит электроэнцефалография — графическая регист-

рация потенциалов головного мозга специальным прибором — электроэнцефалографом. Записываемая при этом кривая — электроэнцефалограмма — используется в исследовательских и диагностических целях.

Все более широкое применение в медицине находит ультразвуковая диагностика — использование ультразвуковых колебаний для распознавания заболеваний мозга (эхоэнцефалография), сердца (эхокардиография) и т. д. Подобная диагностика основана на свойстве ультразвуковых волн отражаться от границ, разделяющих среды, что позволяет видеть контуры внутренних органов и различать образования с различной плотностью.

Ультразвуковое исследование (УЗИ) проводят для диагностики болезней органов брюшной полости, например желчнокаменной болезни. Определение пола будущего ребенка на ранней стадии беременности стало обыденной процедурой. Аппараты УЗИ есть даже на станциях московского метро.

«Заглянуть» в такие внутренние органы, как пищевод, желудок, мочевой пузырь, бронхи, позволяет эндоскоп — оптический прибор, который вводится внутрь исследуемого органа. Эндоскоп представляет собой световод — тонкий гибкий пучок стеклянных волокон из специального оптического стекла, — который освещает внутреннюю поверхность органа и передает его изображение на экран телевизора или в фотокамеру.

В конце 1960-х гг. началось использование томографии (от греч. tómos — ломоть, слой и grapho — пишу) — метода неразрушающего послойного исследования внутренней структуры объекта, например мозга. Томография осуществляется с помощью многократного просвечивания в различных пересекающихся направлениях (так называемое сканирующее просвечивание).

Теперь о молекулярной медицине, эпоха которой наступила в начале нынешнего века в результате поразительных успехов, достигнутых генетикой и генной инженерией. *Молекулярная медицина* — это диагностика, лечение и профилактика на генном уровне. Развитие этого направления позволит выявлять генетическую предрасположенность человека к различным болезням и проводить лечение. При этом в качестве лекарственного препарата выступают гены. Генная терапия не только

устраняет определенные симптомы болезни, но и корректирует функции клеток и всего организма. Ее терапевтический эффект может достигаться заменой больного гена здоровым, коррекцией его структуры и функции, частичным или полным его подавлением.

В будущем предполагается создать «генетический паспорт» человека. Он должен содержать информацию о наличии в его геноме генов наследственных болезней и генов предрасположенности к другим заболеваниям.

Кроме всех этих технических и биологических средств диагностики, у современного врача есть экспертные системы, которые на основе проведенных исследований помогают установить правильный диагноз и назначить соответствующее лечение. В эти экспертные системы заложен весь предыдущий врачебный опыт.

Благодаря развитию современных информационных технологий и прежде всего средств связи, начала развиваться *телемедицина* (т. е. медицина на расстоянии) — использование современных компьютерных средств обработки и передачи информации между «центром» и «периферией» системы здравоохранения.

Телемедицина дает возможность врачам даже небольших городов и населенных пунктов консультироваться у специалистов из медицинских центров Москвы, Санкт-Петербурга, Новосибирска; передавать истории болезни из одной клиники в другую; проводить всероссийские медицинские телеконференции, курсы повышения квалификации врачей, что называется, «без отрыва от производства». Участковый врач не может одинаково хорошо разбираться во всех болезнях. Его задача — быстро и точно определить характер заболевания, оказать необходимую помощь. Но в более сложных случаях появляется необходимость направить больного к специалисту. А квалифицированные специалисты чаще всего работают в больницах и институтах больших городов.

Смысл телемедицины — в создании федеральной информационной сети, объединяющей медицинские учреждения (специализированные академические институты, клиники, больницы, учебные медицинские институты и училища). Медицинские учреждения внутри каждого города будут соединены

системами локальной связи (оптоволоконными, телефонными, радиорелейными), а в разных городах — междугородными коммуникациями (в основном системами спутниковой связи). Для этого необходимо развивать средства передачи, приема и воспроизведения информации в самих медицинских учреждениях и объединять их внутрибольничными сетями. Подобные сети позволяют передать информацию о больном (рентгеновские снимки, результаты анализов, электрокардиограммы, данные компьютерной томографии, УЗИ) напрямую квалифицированному специалисту. И тогда во многих случаях необходимость поездки больного в крупный город на консультацию отпадет.

Приходит на помощь медицине и робототехника. Уже выполнено 35 уникальных операций на сердце с помощью хирургической роботизированной системы. Одна из них проведена в России, в Научном центре сердечно-сосудистой хирургии имени Бакулева.

В операционной хирург должен соответствующим образом положить больного, «подвести» к нему робота, правильно расположить его, а дальше операция проводится роботом по указаниям компьютера. Хирург голосом отдает команды компьютеру, в который заложены определенные программы, а компьютер, в свою очередь, «командует» роботом, имеющим набор инструментов: иглодержатель, пинцет, ножницы и др. Инструменты зафиксированы в руках-держателях робота, обладающих высокой подвижностью. Робот манипулирует ими лучше, чем бригада из двух-трех хирургов. Он может работать в самых неудобных положениях.

Роботизированная система способна продлить профессиональную деятельность выдающихся хирургов.

Информация в военном деле. Во все времена любой военачальник стремился упредить неприятеля, первым узнать о его силах и намерениях. Информация, поставляемая разведкой (в том числе электронной) и средствами связи, значит в военном деле больше, чем численный перевес в живой силе и технике.

В середине XIX в. наивысшая скорость передачи информации по телеграфу составляла 30 слов в минуту, а для обороны территории в 10 км^2 требовалось 40 тыс. солдат. Во Второй ми-

ровой войне по телетайпу передавалось 66 слов в минуту, а для обороны той же территории требовалось 360 солдат. Во время операции «Буря в пустыне» в 1991 г. по компьютерной сети передавалось 192 тыс. слов в минуту, а для обороны той же территории потребовалось 23 солдата. По прогнозам экспертов, в 2010 г. скорость передачи информации возрастет до 1,5 млрд слов в минуту, а на оборону территории в 10 км^2 понадобится всего 2 солдата. Сочетание информационных технологий и высокоточного оружия даст возможность небольшим мобильным группировкам быстрого реагирования решать любые стратегические задачи.

Что же входит в арсенал «информационной» войны? Спутники-шпионы и миниатюрные беспилотные самолеты, которые способны различать на земле предметы размером меньше одного метра; мощные суперкомпьютеры, способные быстро проанализировать, оценить ситуацию и помочь принять верное решение; компьютерные вирусы, способные проникнуть в телекоммуникационные сети и системы управления противника и парализовать их действие.

Компьютерная вирусная война ведется уже сегодня. Например, в ходе конфликта между Израилем и арабскими террористами хакеры-взломщики ведут взаимные атаки компьютерных сетей.

Наиболее ярким примером применения современных информационных технологий в военном деле стала операция «Буря в пустыне», проведенная войсками США и их союзниками против Ирака в 1991 г. Это была хорошо спланированная операция, блестяще осуществленная с использованием высокоточного оружия, самолетов-невидимок, беспилотных самолетов-разведчиков, приборов ночного видения, спутников и компьютеров.

За несколько недель до начала боевых действий агенты с помощью портативных компьютеров внедрили в телефонные станции и радиолокационные посты Ирака программные вирусы. В назначенный день и час эти вирусы отключили системы слежения и в первые минуты воздушного налета парализовали систему противовоздушной обороны Ирака. Таким же образом были выведены из строя бортовые радиолокационные системы истребителей иракских ВВС. Это дало возможность

авиации союзников в первые часы уничтожить основные объекты иракских ПВО и завоевать господство в воздухе.

В ответ на авиационные налеты Ирак обстреливал Израиль и Саудовскую Аравию баллистическими ракетами «Скад», но их сбивали американские ракеты «Пэтриот» с помощью сигналов от космической системы предупреждения.

Большая часть объектов системы управления войсками Ирака располагалась в Багдаде. Союзники с помощью авиационных бомб с лазерным наведением и крылатых ракет сумели разрушить их уже к началу наземных операций.

За прошедшие годы операция «Буря в пустыне» была тщательно проанализирована. В результате анализа было выявлено, что информационное превосходство безусловно обеспечило успех операции. Но одновременно оно является достаточно уязвимым местом, доступным для информационных атак, например, со стороны хакеров. Для защиты информационных систем в США принят национальный план.

В результате обобщения опыта «Бури в пустыне» появилось новое военное понятие — «информационная операция».

Пластиковые карточки. А теперь нужно рассказать о носителе информации, который произвел настоящую революцию в системе банковских и торговых платежей, — об электронных деньгах. Это пластиковая smart-карта со встроенным микропроцессором, позволяющая любому человеку в большинстве развитых стран обходиться без наличных денег. Smart-карта представляет собой пластиковый прямоугольник размером 85 × 54 мм. Информация хранится в энергонезависимой памяти (EEPROM). Встроенный микропроцессор обеспечивает защиту памяти от несанкционированного доступа.

В мире существует несколько наиболее известных электронных платежных систем вроде VISA или EuroCard/MasterCard. Пластиковую карточку можно получить в банке, открыв счет.

Микропроцессор карточки следит за целостностью содержащихся в ее памяти данных. Доступ к данным защищен специальным «ключом», который назначается банком, выдавшим данную карточку. Поэтому smart-карту можно безбоязненно повсюду возить с собой. Владелец smart-карты имеет два пароля: один — для зачисления средств, а другой — для их списания. Оплачивая покупку, нужно вставить свою пластиковую карточ-

ку в специальное считывающее устройство торгового терминала и ввести свой пароль. Происходит оплата покупки, и владелец карточки получает чек. Таким же образом можно получать зарплату, расплачиваться в ресторане, парикмахерской или на бензоколонке.

Получить наличные деньги по пластиковой карточке можно через банкомат — электронно-механическое устройство, работающее круглые сутки. В банкоматах можно получить справку об остатке денег на счете и выписку о последних операциях. При выезде за рубеж можно взять с собой карточку с любой суммой и не вносить ее в таможенную декларацию. Пластиковой карточкой можно оплачивать товары и услуги в любой стране мира. Наличные деньги при этом не нужны.

Smart-карты начали применять в аптеках для учета и безналичной оплаты льготных рецептов на лекарства, что обеспечивает абсолютную прозрачность всех механизмов льготного обслуживания и высококачественную статистику. Благодаря этой системе можно узнать: о каждом больном — какие лекарства и как часто он получает; о враче — насколько верно и кому он назначает эти лекарства; об аптеке — какие лекарства через нее проходят, в каких формах и упаковках; о поликлинике — кого и как она обслуживает.

Есть множество и более простых карточек, например с магнитной полосой (карточки для телефонных разговоров, проезда на транспорте и т. д.).

Штриховой код. Еще одна цифровая технология, хорошо знакомая каждому человеку, — это штриховой код. Он представляет собой последовательность черных и белых полос с цифрами и буквами; печатается на простой бумаге или прямо на упаковке предмета либо на самоклеящейся этикетке.

Информация в символе штрихового кода определяется соотношением ширины штрихов и пробелов между ними. Высота не имеет информационного смысла и просто должна обеспечивать надежное считывание, т. е. пересечение лучом специального сканера всех штрихов кода. Ручной сканер имеет вид пистолета. Он вручную наводится на штриховой код, нанесенный на покупаемый товар, например на упаковку лекарства. Считывание кода происходит при нажатии на курок этого «пистолета». Для стационарных сканеров, размещенных, например,

у кассы, нужно поднести предмет с нанесенным штриховым кодом к сканеру. Он обнаруживает и считывает штриховой код, расшифровывает его и передает в компьютер.

Штриховые коды, по сравнению с магнитными и радиоизотопными, обладают высокой надежностью их считывания. Для обеспечения еще большей надежности применяются специальные самоконтролирующиеся и самокорректирующиеся коды. Они обнаруживают ошибки и исправляют их, если число ошибочных знаков не превышает 65—70 %, обеспечивая вероятность одной ошибки на 30 млн считанных знаков.

С помощью штрихового кодирования ведут складской учет товаров в торговле, учет лекарств в аптеках и больницах, движения книг в библиотеках, прохождения посетителей, имеющих пропуска, промаркированные штриховым кодом, в помещения, оборудованные сканерами кодов, и т. д.

Информация в жилище XXI в. Современные дома и квартиры буквально набиты всевозможными электроприборами — светильниками, микроволновыми печами, электрическими плитами, миксерами, кофеварками, стиральными машинами, электроутюгами, электрическими обогревателями, кондиционерами, вентиляторами, часами и т. д. Не меньше в наших жилищах аудио- и видеоаппаратуры — радио, телевизоров, магнитол, видеомагнитофонов, музыкальных центров. Все эти приборы обеспечивают удобство и комфорт, или, как теперь говорят, качество жизни. Уже давно для управления многими из них существуют пульты дистанционного управления. Но каждый из пультов управляет только одним или максимум двумя приборами и действует только в пределах одной комнаты.

Приходя в дом или квартиру, мы вынуждены последовательно включать все эти приборы и управлять ими, а перед сном или уходя из жилища — последовательно выключать их. При этом всегда остается опасность оставить включенными люстру, утюг или любой другой электрический прибор. Как часто мы, уйдя из дома, вдруг начинаем мучительно вспоминать, всё ли мы выключили перед уходом. Иногда эти сомнения заставляют нас возвращаться.

Для комплексного контроля и управления всеми электроприборами в доме или квартире в США создана автоматизиро-

ванная система управления ACTIVE HOME — АКТИВНЫЙ («послушный») ДОМ. Для установки этой системы не нужно сверлить стены и делать дополнительную проводку. Нужно просто установить базовый комплект оборудования ACTIVE HOME и включить его в электросеть дома. В комплект входят два набора — для управления домом и для управления аудио- и видеоприборами. Эти наборы позволят управлять всем в доме — от светильников до домашнего кинотеатра — с помощью одного пульта и самим программировать работу всех устройств.

Система ACTIVE HOME — своеобразный конструктор. Добавляя к базовому комплекту дополнительные модули, можно постепенно расширять ее возможности. ACTIVE HOME управляет всеми электроприборами и даже шторами и жалюзи из любого помещения дома или квартиры с помощью домашнего персонального компьютера и программирует их работу по желанию хозяина. Она позволяет создавать абсолютный эффект присутствия, когда никого нет дома, а также управлять всеми приборами или запускать программы управления домом по телефону из любой точки мира (например, вскипятить чайник, подогреть любое блюдо или изменить температуру в доме).

ACTIVE HOME снабжена системами пожарной безопасности и управления отоплением; миниатюрная цветная видеокарта позволяет наблюдать за происходящим в доме и вокруг него с помощью системы беспроводного видеонаблюдения. Можно также наблюдать за домом, воспользовавшись набором охраны дома Safety comfort и имитатором лая собаки Robo-Dog с последующим включением яркого света, а также другими «отпугивающими» программами: записью голосов людей, включением телевизора. Набор для охраны оборудован беспроводным датчиком движения.

Таким образом, ACTIVE HOME превращает дом или квартиру в единый легкоуправляемый комплекс.

Компьютер в роли переводчика. Первые программы автоматического (или машинного) перевода текстов с одного языка на другой с помощью компьютера появились уже в 1950-е гг. Однако широкое распространение они получили только после появления персональных компьютеров, которые позволяют пол-

ностью использовать главное преимущество машинного перевода по сравнению с обычным — его оперативность. За прошедшие десятилетия эти программы были усовершенствованы, но многие проблемы машинного перевода, особенно литературного, не решены до сих пор.

Компьютер не понимает нюансов языка, игры слов, намеков. При машинном переводе предложение расчленяется на отдельные части речи, в нем выделяются стандартные конструкции. Слова и словосочетания переводятся по словарям, занесенным в память компьютера. Затем переведенные части речи собираются по правилам другого языка. При этом получается не полноценный литературный перевод, а «полуфабрикат», черновик перевода — неправильный в литературном отношении подстрочник. Но даже такой подстрочник представляет ценность для человека, плохо владеющего языком оригинала, но обладающего способностями литературного редактора и знаниями в конкретной области науки, техники или искусства, к которой относится переводимый текст.

Точности технического перевода способствует правильный выбор словаря по специальности. Для качественного перевода важно, чтобы в этом словаре можно было найти максимальное число слов переводимого текста.

В мире существует очень много программ автоматического машинного перевода. В России наиболее распространена система Stylus (фирма «ПроМТ»), предназначенная для профессионального перевода больших объемов информации с русского на английский, французский, немецкий языки и обратного перевода с этих языков на русский. Наилучший результат дает использование программы машинного перевода с внешним текстовым редактором Microsoft Word for Windows.

В наши дни наблюдается повышенный интерес к системам автоматического перевода в связи с бурным развитием сети Internet. В ней доминирует английский язык, но есть веб-страницы и на других языках. Чтобы облегчить их просмотр пользователями, не знающими этих языков, появились специальные дополнения к браузерам Microsoft Internet и Explorer Netscape Navigator, созданные на базе программы Stylus. Они обеспечивают немедленный перевод просматриваемых фрагментов текстов.

1.2.4. Информационные основы процессов управления

Понятие «управление» — одно из центральных понятий кибернетики, науки о наиболее общих закономерностях в технических, биологических и социальных системах. Управление пронизывает все сферы деятельности человека и общества.

Управление — это целенаправленная организация того или иного процесса, протекающего в системе. В общем случае процесс управления состоит из следующих четырех элементов:

— получения информации о задачах управления (цель управления);

— получения информации о результатах управления (т. е. о поведении объекта управления);

— анализа полученной информации и выработки решения;

— использования решения (т. е. осуществления управляющих воздействий).

Процесс управления — это информационный процесс, который заключается в сборе информации о ходе процесса; передаче ее в пункты накопления и переработки; анализе поступающей, накопленной и справочной информации; принятии решения на основе выполненного анализа; выработке соответствующего управляющего воздействия и доведения его до объекта управления. Каждая фаза процесса управления протекает во взаимодействии с окружающей средой при воздействии различного рода помех. Цели, принципы и границы управления зависят от сущности решаемой задачи.

Управление всегда «разыгрывается» между двумя объектами — объектом управления и управляющим органом. Объект управления служит для удовлетворения материальных потребностей человека, т. е. для производства материальных благ (предметов потребления — продуктов промышленности и сельского хозяйства, транспортных средств, электричества, тепла, средств коммуникации и т. д.), а также для удовлетворения интеллектуальных потребностей (кино, телевидение, радио, образование, театр и т. д.). Через объект управления проходят в основном материальные потоки и в значительно меньшей степени — информационные; в то же самое время через управляющий орган проходят только информационные потоки.

Именно поэтому процесс управления является информационным процессом, а все люди, занятые в сфере управления, имеют дело только с информацией. Отсюда вторым важным понятием является понятие «система управления».

Система управления (СУ) — это совокупность взаимодействующих между собой объекта управления и органа управления, деятельность которых направлена на достижение заданной цели управления. В СУ решаются четыре основных типа задач управления: стабилизация, выполнение программы, сложение и оптимизация.

Задача *стабилизации системы* — поддержание ее выходных величин вблизи некоторых неизменных знаний, несмотря на действие помех (стабилизация напряжения U и частоты F тока в сети вне зависимости от изменения потребления энергии).

Задача *выполнения программы* возникает в случаях, когда заданные значения управляемых величин изменяются во времени заранее известным образом (полет ракеты, выполнение работ по заранее полученному графику).

Задача *сложения* — это как можно более точное соответствие между текущим состоянием и состоянием другой системы (управление производством в условиях изменения спроса; слежение за целью — самолетом, кораблем).

Задача *оптимизации* — наилучшим образом выполнить поставленную перед системой задачу при заданных реальных условиях и ограничениях.

1.2.5. Информатизация общества

В истории человечества было несколько промышленных и информационных переворотов. Первый промышленный переворот основывался на паровом двигателе, затем на электромоторе и двигателе внутреннего сгорания, т. е. на энергопреобразующих машинах. С ним связаны индустриализация общества, переход к индустриальной цивилизации. Второй промышленный переворот связан с овладением человечеством атомной энергией и проникновением в космос — современная научно-техническая революция (НТР).

Перевороты в информационной сфере были уже трижды. При первом перевороте — появлении письменности — возник специальный механизм фиксирования и распространения информации во времени и пространстве. Появились документирование информации и информационно-накопительные центры в виде библиотек, архивов. Второй информационный переворот связан с книгопечатанием — тиражированием документированной информации с помощью специальной машины — печатного станка. Если появление письменности коррелирует с ростом науки, просвещения и искусства, то типографский станок породил книжно-журнальную и газетную индустрию — основу дальнейшего культурного прогресса человечества, а также индустрию документооборота. Прорыв состоял в производстве «бумажной» информации, возникновении механизма ее размножения и хранения. Но переработка и использование информации по-прежнему зависели от физиологических возможностей человека с его способностью за секунду воспринимать восемь-девять единиц информации и в лучшем случае произвести 15 логических операций.

Скачкообразное увеличение скорости распространения информации, вызванное появлением электрического телеграфа, телефона, радио, телевидения, а также ростом на два-три порядка транспортных скоростей в связи с появлением автотранспорта и авиации, конечно, привело к огромной интенсификации информационных потоков и коммуникативных процессов (третий информационный переворот). Но это существенно не отразилось на скорости выработки сигналов обратной связи, которые попрежнему зависели от способностей «невооруженной» головы человека.

Таким образом, человечество столкнулось с новым противоречием — между машинной вооруженностью рук (в середине XX в. 99 % всей физической работы выполнялось машинами) и «ремесленной вооруженностью» головы (интеллекта). Информационно-управляющие возможности людей, ограниченные физическими рамками, перестали соответствовать вещественно-энергетическим возможностям производства, ставшим, по существу, безграничными в силу использования машинной техники. Постепенно разрастаясь, это противоречие вело к информационно-организационному кризису. Оно и разрешается

путем создания информационных, компьютеризованных технологий в области переработки и использования знаний.

С первым промышленным переворотом связано появление промышленных технологий, со вторым — появление информационных технологий, насыщение которыми народного хозяйства и означает информатизацию общества. Переворот в информационной сфере, ее индустриализация на базе ЭВМ составляют главное содержание современной НТР; в то же время его можно считать третьим (после изобретения письменности и книгопечатания) в истории великим информационным переворотом.

До начала 1970-х гг., когда компьютеризация была поверхностной и очаговой, потребность в информатизации общества остро не ощущалась. Резкое расширение процессов компьютеризации, массовое развитие микропроцессорной техники и особенно наметившийся переход к пятому поколению ЭВМ сделали информатизацию насущно необходимой. **Информатизация** — это как бы надстроечный процесс, происходящий на базе компьютеризации, т. е. индустриализации непроизводственной сферы народного хозяйства, процесс формирования новой, автоматизированной среды зарождения знаний, их переработки, распространения и превращения в силу, в материальный фактор.

Техническая база информатизации — это компьютерные и телекоммуникационные системы и сети, которые составляют «ядро» экономики, точнее — производственного аппарата будущего общества. Такой аппарат будет включать роботы и роботизированные производства, обрабатывающие центры, гибкие производственные системы, безлюдные участки, цехи, предприятия и, конечно, новые организационно-управленческие комплексы и системы связи.

Альтернативы информатизации нет. Это объективный этап социального прогресса во всех областях, прежде всего в экономике, управлении, науке и технологии.

В настоящее время стало ясно: чтобы та или иная страна могла занять достойное место в мире и на равных участвовать в экономическом соревновании с другими государствами, она должна перестраивать и приспосабливать свои структуры, приоритеты, ценности, институты к требованиям индустриальной ИТ. Экономические позиции той или иной страны вплоть до середины нынешнего века будут определяться такими нова-

торскими технологиями, как термоядерный синтез, биотехнология, плазменные процессы, космическая связь и др. Но их развитие, в свою очередь, зависит от развития информатики. Нельзя представить себе эффективное и надежное функционирование, например, атомных станций или космической связи без высокотехнологичного, т. е. компьютерного, информационно-организованного, обеспечения.

Информационное общество характеризуется следующими основными признаками:

1. Большинство работающих в информационном обществе (около 80 %) заняты в информационной сфере, т. е. сфере производства информации и информационных услуг.

2. Обеспечены техническая, технологическая и правовая возможности доступа любому члену общества практически в любой точке территории и в приемлемое время к нужной ему информации (за исключением военных и государственных секретов, оговоренных в соответствующих законодательных актах).

3. Информация становится важнейшим стратегическим ресурсом общества и занимает ключевое место в экономике, образовании и культуре.

Информатизация — необходимое условие научно-технического, социального, экономического и политического прогресса в обществе. Неизбежность информатизации обусловлена:

— беспрецедентным усложнением социально-экономических процессов в результате увеличения масштабов и темпов общественного производства, углубления разделения труда и его специализации в условиях научно-технической революции;

— необходимостью адекватно реагировать на возникающие проблемы в динамично меняющейся обстановке, присущей постоянно развивающемуся обществу;

— повышением степени самоуправления предприятий, территорий, регионов.

Процесс перехода от постиндустриального общества к информационному происходит не одновременно в различных странах, он характеризуется также и разными темпами развития. Первыми на этот путь встали в конце 1950-х — начале 1960-х гг. США, Япония и страны Западной Европы. В этих государствах, начиная с 1960—1970-х гг., проводится политика повсеместной информатизации всех сфер деятельности че-

ловека. Были разработаны и приняты на государственном уровне программы информатизации с целью наиболее полного использования информационного ресурса для ускорения экономического, социального и культурного развития общества. Предполагается, что США завершат переход к информационному обществу к 2020 г., Япония и основные страны Западной Европы — к 2030—2040 гг.

В СССР в 1989 г. была разработана Концепция информатизации общества. По предварительным оценкам, информатизация в России завершится к 2050 г. при условии стабилизации экономической и политической обстановки в стране. По мнению специалистов, любая страна, насколько бы индустриально развитой она ни была, перейдет в разряд стран «третьего» мира, если опоздает с информатизацией.

В настоящее время объем расходов США на информатизацию (создание, производство, монтаж, использование ЭВМ, информационных сетей и систем разного уровня, БД и т. д.) достиг нескольких сот миллиардов долларов в год и превышает объем военных расходов. Американские системы компьютерной оборонной инициативы (КОН) и особенно стратегической оборонной инициативы (СОИ) базируются на развитых ИТ, которые, в свою очередь, предполагают информатизацию управления, науки, проектных разработок, всех систем выработки, принятия и поддержки решений.

Если предшествующие этапы развития человечества длились каждый около трех веков, то ученые прогнозируют, что информационный этап продлится значительно меньше. Срок его существования ограничится, вероятно, сотней лет. Это означает, что основные регионы мира войдут в развитое информационное общество в XXI в. и тогда же начнется переход к постинформационному обществу.

Для информационного общества характерно обеспечение требуемой степени информированности всех его членов, возрастание объема и уровня информационных услуг, предоставляемых пользователю. Информационное общество в теоретическом аспекте характеризуется высокоразвитой информационной сферой (инфосферой), которая включает деятельность человека по созданию, переработке, хранению, передаче и накоплению информации.

К первоочередным проблемам информатизации следует отнести психологическую проблему готовности населения к переходу в информационное общество. Этот переход в настоящее время затрудняется низким уровнем информационной культуры населения, недостаточной компьютерной грамотностью, а отсюда и низкими информационными потребностями и отсутствием желания их развивать. Наблюдается невосприимчивость экономики, управления на всех уровнях к результатам НТР, и в первую очередь в инфосфере. Психофизиологический аспект проблемы определяется совместимостью человека и новой информационной техники и технологии.

Необходимо еще раз подчеркнуть, что информатизация общества предполагает организацию компьютерного ликбеза населения; подготовку и переподготовку кадров (специалистов по ЭВМ, пользователей, профессионалов в области программирования и вычислительной техники); компьютеризацию всех звеньев образования — от начальной школы до вуза и системы послевузовского образования; создание огромной сети переподготовки работников; формирование, особенно у молодежи, новой информационной культуры; расширение математического образования; преодоление барьеров на пути к ПЭВМ, машинным языкам и т. д.

Информатизация прежде всего охватывает все элементы рыночной инфраструктуры экономики:

— сеть оптовой и розничной торговли;

— товарные, фондовые, валютные биржи, биржи труда, ярмарки;

— сети банков и кредитно-финансовых учреждений;

— сеть независимых посреднических фирм и контор, оказывающих различные услуги, в том числе и информационные, субъектам рынка;

— складское и транспортное хозяйство;

— страховые компании;

— налоговую службу;

— систему подготовки и переквалификации кадров;

— аудиторскую службу и т. д.

Для изучения спроса и предложения (товаров, идей, видов обслуживания) создаются региональные сети самых различных форм (лаборатории изучения спроса, информационно-коммер-

ческие и рекламно-маркетинговые центры, центры аналитических исследований и научного прогнозирования и т. п.), специальные подразделения при крупных производственных, сбытовых и торговых предприятиях.

Важную роль в экономике играет сфера услуг. Она доминирует как в производстве валового продукта, так и в обеспечении занятости населения в регионе. Информатизация сферы услуг стимулирует развитие всех видов сервиса, особенно в торговле, финансово-кредитной сфере, здравоохранении, организации досуга и отдыха населения, что положительно сказывается на производственной сфере, обеспечивает экономический рост, снижает напряженность на региональном рынке трудовых ресурсов. При этом возрастают потребности самой сферы услуг в информационных средствах, технологиях и услугах. Информатизация сферы услуг в экономике способна повысить производительность труда и тем самым обеспечить реальное безынфляционное увеличение доходов и рост благосостояния населения в течение длительного времени.

Информатизация бирж труда позволяет рационально использовать людские ресурсы региона, сбалансировать предложение и спрос на рабочую силу, предоставлять жителям региона работу в соответствии с их способностями и возможностями.

Приведенные примеры свидетельствуют, что информатизация проводится по различным направлениям экономики и социальной политики общества.

Таким образом, **информационное общество** — это общество, структуры, техническая база и человеческий потенциал которого приспособлены для оптимального превращения знаний в ИР и переработки последнего с целью перевода его пассивных форм (книги, статьи, патенты и т. п.) в активные (модели, алгоритмы, программы, проекты). Но особое значение для активизации информационного потенциала общества имеет создание современных баз знаний. Это достигается на путях качественного преобразования традиционных баз данных (БД), рожденных ранними поколениями ЭВМ до появления искусственного интеллекта, в базы знаний (БЗ).

Научным фундаментом процесса информатизации общества является новая научная дисциплина — информатика. В ши-

роком смысле информатика — это наука об информационной деятельности, информационных процессах и их организации в человеко-машинных системах. К основным разделам информатики относятся исследование и разработка информационных средств и технологий, программных средств, а также моделирование предметных областей.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ТЕМЕ 1.2

1. Назовите этапы технологического процесса в информационных системах.

2. Раскройте содержание информационных процессов в живой природе (наследственность, клонирование, нервная система, память, искусственный интеллект).

3. В чем состоит аналогия между поведением машин и живых организмов?

4. Какова разница между мозгом и вычислительной машиной?

5. Возможно ли воспроизведение компьютерами себе подобных?

6. Дайте характеристику информационных процессов в медицине.

7. Что такое телемедицина?

8. Расскажите об информационных процессах в военном деле.

9. Что вы знаете о пластиковых карточках?

10. Что такое штриховой код?

11. Перечислите известные вам информационные процессы в жизни современного человека.

12. Можно ли создать машинный профессиональный переводчик?

13. Дайте определение понятия «управление».

14. Почему процесс управления является информационным процессом?

15. Какие вы знаете информационные перевороты?

16. Каковы сущность и цели информатизации?

17. В чем состоит техническая база информатизации?

18. Раскройте основные признаки информационного общества.

19. Что является научным фундаментом процесса информатизации общества?

Раздел 2

СИСТЕМЫ СЧИСЛЕНИЯ И ОСНОВЫ ЛОГИКИ

В результате изучения материала второго раздела студент должен:

знать:

— функции языка как способа представления информации;

— способы хранения и основные виды хранилищ информации;

— основные единицы измерения количества информации;

— правила выполнения арифметических операций в двоичной системе счисления;

— основные логические операции, их свойства и обозначения;

уметь:

— переводить числа из одной системы счисления в другую;

— строить логические схемы из основных логических элементов по формулам логических выражений;

— перечислять особенности и преимущества двоичной формы представления информации;

— решать задачи на определение количества информации;

— представлять логические выражения в виде формул и таблиц истинности;

иметь представление:

— о принципах кодирования информации;

— о системах счисления.

Методические указания

Так как понятие «информация» занимает центральное место в информатике, то будет естественным начать изучение математических основ информатики с методов и моделей количе-

ственной оценки информации. Трудно сейчас представить материал этого раздела о математических основах информатики без темы «Системы счисления». Во втором разделе рассмотрены основные на сегодня системы счисления, используемые в вычислительной технике: позиционные и смешанные, перевод чисел из одной системы счисления в другую; формы представления и преобразования информации; числовая система ЭВМ; представление чисел; кодирование и декодирование символьной информации в ЭВМ; форматы данных. Изложение проиллюстрировано доходчивыми примерами.

Студенты должны усвоить «кибернетическое» понятие информации как произвольного текста, т. е. последовательности символов некоторого алфавита, а также понятие «системы счисления»; знать формы представления и преобразования информации, т. е. каким образом в ЭВМ организована числовая и символьная информация; уметь определять количество информации в конкретных сообщениях (при заданном способе кодирования) и объем памяти ЭВМ, необходимый для хранения данной информации.

Тема 2.1

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ. КОЛИЧЕСТВО И ЕДИНИЦЫ ИЗМЕРЕНИЯ ИНФОРМАЦИИ

2.1.1. Язык как способ представления информации

Язык — это система обозначений и правил для передачи *сообщений*. Различают языки естественные, на которых общаются люди, и искусственные (или формальные), обеспечивающие взаимодействие систем «человек — машина» или «машина — машина». К формальным языкам относятся языки программирования.

Формальный язык задается алфавитом, синтаксисом и семантикой. Теоретические основы методов проектирования языков программирования, конструирования трансляторов рассматриваются в теории формальных языков.

Формальный язык — это язык, обеспечивающий удобное описание конкретных проблем, формулируемых человеком и решаемых с помощью компьютера.

На языке программирования пишется программа, позволяющая при ее выполнении компьютером (вычислительной системой) получить конкретные результаты. Язык программирования состоит из *синтаксиса* и *семантики*.

Алфавит представляет собой совокупность упорядоченных в определенном смысле символов в данном языке или системе. Эти символы называются **буквами**. Только символы, принадлежащие данному алфавиту, можно использовать для построения слов. Буква — это элемент алфавита. Например, алфавит языка Паскаль состоит из латинских букв (причем строчные и прописные буквы не различаются), цифр и специальных символов.

Под *символом* понимается элемент алфавита, имеющий определенное значение. Символ, как правило, записывается в памяти компьютера восемью *битами*, или одним *байтом*.

Синтаксис (от греч. sýntaxis — построение, порядок) — это набор правил построения *слов, конструкций* и *структур* текста в языке или системе. Некоторые авторы включают в синтаксис и алфавит. Ошибки, возникающие при написании программы и касающиеся только синтаксиса, выявляются при синтаксическом анализе, осуществляемом *транслятором*.

В информатике понятие «слово» имеет несколько определений, приведем два из них; слово — это:

— упорядоченный набор символов в заданном алфавите, имеющий определенный смысл;

— единица данных, рассматриваемая как целое при передаче и обработке данных в процессе.

Транслятор (от англ. translator — переводчик) — это программа, производящая трансляцию программы с одного языка программирования в другой.

Под ***семантикой*** (от греч. sēmantikós — обозначающий) понимается смысл каждой синтаксической конструкции в языке или системе.

В язык программирования транслятор превращает синтаксические построения команд, понятные операционной системе и процессору. Смысловые ошибки транслятор не выявляет, их поиск осуществляет человек в процессе отладки, тестирования и даже эксплуатации языка программирования.

Естественный язык возник очень давно. Первобытные люди могли обеспечить себе пропитание (например, добыть круп-

ное животное) только общими усилиями. При этом они должны были общаться между собой, обмениваться информацией (скажем, предупреждать друг друга об опасности). Сначала это делалось с помощью мимики, жестов, возгласов; затем возник язык — важнейшее средство человеческого общения. Он стал средством передачи информации, значительно расширил возможности ее хранения в памяти, стал одним из средств управления поведением человека. Реализуется язык в виде *речи* — языка в действии. В древнейшие времена говорить можно было только с находящимся рядом собеседником или с группой слушателей, так как речь исчезает в момент ее произнесения.

Речь «участвует» не только в передаче сведений от одного человека другому, но и во всех сознательных процессах, происходящих в мозгу человека. Каждый из нас мыслит с помощью речи — словами, которые не произносятся вслух. Даже когда мы молчим, мы говорим сами с собой, мыслим не только образами, но и словами. Воспринимаемые нами предметы в своем сознании мы называем словами-понятиями: *стол, стул, небо, земля, утро, день, вечер, ночь.* Считаем в уме мы тоже с помощью слов, обозначающих числа и вычисления: *один, два, три, четыре, сложить, вычесть, умножить, разделить.* Например: «Девять умножить на три равно двадцати семи», — говорим мы «в уме».

Зрительные, звуковые и другие впечатления человек запоминает в виде образов, а смысловую информацию — в виде слов. Чем раньше человек научится говорить, тем больше у него сохранится ранних детских воспоминаний.

Итак, речь представляет собой самую древнюю природную информационную технологию, которой каждый человек овладевает в первые годы жизни. Для сознательных процессов в мозгу человека эта технология является внутренней, а для передачи своих мыслей другим (с помощью речи) — внешней.

Процесс формирования речи объяснил выдающийся отечественный ученый-физиолог И.П. Павлов (1849—1936), который разработал учение о сигнальных системах. Сигнальные системы условно-рефлекторных связей формируются в коре больших полушарий головного мозга животных и человека при поступлении импульсов от внешних и внутренних раздражителей. Другими словами, мы слышим за спиной голос знакомого человека и тут же оборачиваемся. Голос в этом случае будет раз-

дражителем, а наша реакция на обращение приятеля — реакцией на раздражитель.

Первая сигнальная система формируется при воздействии конкретных раздражителей — света, звука, боли, запаха и т. д. Она является формой непосредственного отражения действительности в виде ощущений.

Вторая сигнальная система, присущая только человеку, формируется при воздействии речевых сигналов, т. е. не непосредственных раздражителей, а их словесных обозначений. Вторая сигнальная система формируется на базе первой в процессе общения между людьми, первоначально между ребенком и родителями. К примеру, грудной ребенок не понимает, что именно говорит ему мать, он лишь реагирует на ее голос. Но когда он начинает понимать смысл сказанного, голос матери для него уже не просто знакомые звуки, но и самое главное средство общения. То есть первая сигнальная система, данная нам от природы, дополняется второй сигнальной системой в процессе обучения. Понятие о второй сигнальной системе ввел в 1932 г. И.П. Павлов.

2.1.2. Формы представления и преобразования информации

При любых формах работы с информацией всегда идет речь о ее представлении в виде определенных символических структур. Наиболее распространены одномерные представления информации, при которых сообщения имеют вид последовательности символов. Так информация представляется в письменных текстах, при передаче по каналам связи, в памяти ЭВМ. Однако широко используется и многомерное представление информации, причем под *многомерностью* понимается не только расположение элементов информации на плоскости или в пространстве в виде рисунков, схем, графов, объемных макетов и т. п., но и множественность признаков используемых символов, например цвет, размер, вид шрифта в тексте.

Формирование представления информации называется ее **кодированием**. В более узком смысле под кодированием понимается переход от исходного представления информации,

удобного для восприятия человеком, к представлению, удобному для ее хранения, передачи и обработки. В этом случае обратный переход к исходному представлению информации называется **декодированием**.

При кодировании информации ставятся следующие цели:

— удобство физической реализации;

— удобство восприятия;

— высокая скорость передачи и обработки;

— экономичность, т. е. уменьшение избыточности сообщений;

— надежность, т. е. защита от случайных искажений;

— сохранность, т. е. защита от нежелательного доступа к информации.

Эти цели часто противоречат друг другу. Стремясь к экономным сообщениям, мы тем самым уменьшаем их надежность и удобство восприятия. Экономные сообщения могут повысить скорость обработки информации (такое сообщение будет передано или прочтено быстрее), но могут и уменьшить ее. А защита от нежелательного доступа уменьшает объем хранимой информации и замедляет работу с ней.

Рассмотрим способы представления информации в ЭВМ. Для записи, хранения и выдачи по запросу информации, обрабатываемой с помощью ЭВМ, предназначено запоминающее устройство (или память) вычислительной машины.

В отличие от обычной словесной формы, принятой в письменной речи, *информация в памяти ЭВМ записывается в форме цифрового двоичного кода*. Это объясняется тем, что электронные элементы, на которых строится память ЭВМ, находятся только в одном из двух устойчивых состояний — их можно интерпретировать как 0 или 1.

Количество информации, которое может помещаться в один элемент памяти (0 или 1), называемый **битом,** очень мало́ и не несет никакой смысловой нагрузки. Однако если соединить несколько таких элементов в ячейку, то можно сохранить в запоминающем устройстве столько информации, сколько потребуется. Последовательность бит, рассматриваемых аппаратной частью ЭВМ как единое целое, называется **машинным словом**. Слово «бит» происходит от английских слов binary — двойной и digit — цифра: BInary + digiT = BIT.

Так как оперативная память ЭВМ состоит из конечной последовательности слов, а слова — из конечной последовательности бит, то объем представляемой в ЭВМ информации ограничен *емкостью памяти,* а числовая информация может быть представлена только с определенной точностью, зависящей от архитектуры памяти данной ЭВМ.

2.1.3. Количество и единицы измерения информации

Для теоретической информатики информация играет такую же роль, как и вещество в физике. И подобно тому как веществу можно приписать довольно большое количество характеристик: массу, заряд, объем и т. д., — так и для информации имеется пусть и не столь большой, но достаточно представительный набор характеристик. Для характеристики как вещества, так и информации имеются единицы измерения, что позволяет некоторой порции информации приписывать числа — *количественные характеристики информации*.

На сегодняшний день наиболее известны объемный, энтропийный и алгоритмический способы измерения информации.

Объемный — самый простой и грубый способ измерения информации. Соответствующую количественную оценку информации естественно назвать **объемом информации**. *Объем информации в сообщении* — это количество символов в нем. Поскольку одно и то же число может быть записано разными способами, т. е. с использованием разных алфавитов (например: двадцать один; 21; 11001; XXI), объемный способ чувствителен к форме представления (записи) сообщения. Повторим: в вычислительной технике вся обрабатываемая и хранимая информация, вне зависимости от ее природы (число, текст, изображение), представлена в двоичной форме (с использованием алфавита, состоящего всего из двух символов: 0 и 1). Такая стандартизация позволила ввести две стандартные единицы измерения: **бит** и **байт**. Восемь бит составляют один байт.

В теории информации и кодирования принят ***энтропийный*** способ измерения информации. Он исходит из следующей модели. Получатель информации (сообщения) имеет определен-

ные представления о возможных наступлениях некоторых событий. Эти представления в общем случае недостоверны и выражаются вероятностями, с которыми он ожидает то или иное событие. Общая мера неопределенности *(энтропия)* характеризуется некоторой математической зависимостью от совокупности этих вероятностей. Количество информации в сообщении определяется тем, насколько уменьшится эта мера после получения сообщения.

Поясним эту идею на примере. Пусть имеется колода из 32 различных карт. Возможность выбора одной карты из колоды — 32. Априори (доопытно, до того, как произведен выбор) естественно предположить, что наши шансы выбрать некоторую определенную карту одинаковы для всех карт колоды. Произведя выбор, мы устраняем эту априорную неопределенность. Нашу априорную неопределенность можно было бы охарактеризовать количеством возможных равновероятных выборов. Если теперь определить количество информации как меру устраненной неопределенности, то и полученную в результате выбора информацию можно охарактеризовать числом 32.

Однако в теории информации получила использование другая количественная оценка, а именно — логарифм от описанной выше оценки по основанию 2:

$$H = \log_2 m, \tag{2.1}$$

где m — число возможных равновероятных выборов (при $m = 2$, $H = 1$).

То есть для выбора из колоды имеем следующую оценку количества информации, получаемую в результате выбора:

$$H = \log_2 32 = 5.$$

Полученная оценка имеет интересную интерпретацию. Она характеризует число двоичных вопросов, ответы на которые позволяют выбрать либо «да», либо «нет». Для выбора дамы пик такими вопросами будут:

1. Карта красной масти?	Ответ «нет» — 0.
2. Трефы?	Ответ «нет» — 0.
3. Одна из четырех старших?	Ответ «да» — 1.
4. Одна из двух старших?	Ответ «нет» — 0.
5. Дама?	Ответ «да» — 1.

Этот выбор можно описать последовательностью из пяти двоичных символов: 00101 (0 — «нет», 1 — «да»).

На первый взгляд может показаться, что эта интерпретация не годится в случае, когда количество выборов не равно степени двойки, так как получается нецелое количество вопросов; к примеру, если взять колоду из 36 карт (добавлены шестерки), то можно заметить, что для того, чтобы выяснить у участника «эксперимента», какую карту он выбрал, в ряде случаев понадобится пять вопросов (как и в рассмотренном случае), а в ряде случаев — шесть. Усреднение по случаям и дает получаемую по формуле нецелую величину.

К. Шеннону принадлежит обобщение H на случай, когда H зависит не только от m, но и от вероятностей возможных выборов (для сообщения — вероятности выбора символов). Для количества собственной индивидуальной информации он предложил соотношение:

$$h_i = \log (1/P_i) = -\log P_i, \qquad (2.2)$$

где P_i — вероятность выбора i-го символа алфавита.

Более удобно пользоваться средним значением количества информации, приходящимся на один символ алфавита:

$$H = -\sum_{j=1}^{m} P_j \log P_j, \quad j = \overline{1, m}. \qquad (2.3)$$

При равновероятных выборах все $P_j = 1/m$, и получается прежняя формула: $H = \log m$.

В алгоритмической теории информации (раздел теории алгоритмов) предлагается ***алгоритмический*** способ оценки информации в сообщении. Этот способ кратко можно охарактеризовать следующими рассуждениями.

Каждый согласится, что слово 0101...01 сложнее слова 00...0, а слово, где 0 и 1 выбираются из эксперимента — бросания монеты (где 0 — герб, 1 — число), сложнее предыдущих.

Компьютерная программа, производящая слово из одних нулей, крайне проста: печатать один и тот же символ. Для получения 0101...01 нужна чуть более сложная программа, печатающая символ, противоположный только что напечатанному. Случайная, не обладающая никакими закономерностями по-

следовательность не может быть произведена никакой «короткой» программой. Длина программы, производящей хаотичную последовательность, должна быть близка к длине последней.

Приведенные рассуждения позволяют предположить, что любому сообщению можно приписать количественную характеристику, отражающую сложность (размер) программы, которая позволяет ее произвести.

Так как имеется много разных вычислительных машин и разных языков программирования (разных способов задания алгоритма), то для определенности используется конкретная вычислительная машина, а сложность слова (сообщения) определяется как минимальное число внутренних состояний машины, требующееся для его воспроизведения. В алгоритмической теории информации применяются и другие способы задания сложности.

Кроме бита и байта, для измерения количества информации используются также более крупные единицы:

1 килобайт = 1024 байт (2^{10} байт);
1 мегабайт = 1024 Кбайт (2^{20} байт);
1 гигабайт = 1024 Мбайт (2^{30} байт);
1 терабайт = 1024 Гбайт (2^{40} байт);
1 петабайт = 1024 Тбайт (2^{50} байт);
1 эксабайт = 1024 Пбайт (2^{60} байт).

Тема 2.2

СИСТЕМЫ СЧИСЛЕНИЯ, ИСПОЛЬЗУЕМЫЕ В КОМПЬЮТЕРЕ

Системой счисления называется совокупность приемов наименования и записи чисел. В любой системе счисления для представления чисел выбираются некоторые символы (слова или знаки), называемые **базисными числами**, а все остальные числа получаются в результате каких-либо операций из базисных чисел данной системы счисления. Символы, используемые для записи чисел, могут быть любыми, только они должны быть разными и значение каждого из них должно быть известно. В современном мире наиболее распространенным является представление чисел посредством арабских цифр: 0, 1, 2, 3, 4, 5,

6, 7, 8, 9 — специальных знаков, используемых для записи чисел. Системы счисления различаются выбором базисных чисел и правилами образования из них остальных чисел. Например, в римской системе счисления базисными являются числа 1, 5, 10, 50, 100, 500, 1000, которые обозначаются знаками, соответственно I, V, X, L, C, D, M, а другие числа получаются путем сложения и вычитания базисных: если цифра справа меньше или равна цифре слева, то эти цифры складываются; если цифра слева меньше, чем цифра справа, то левая цифра вычитается из правой. Так, например, число 146 в римской системе счисления имеет вид: CXLVI (C — 100, XL — 40, VI — 6). Здесь число 40 получается посредством вычитания из 50 числа 10, а 6 — посредством сложения чисел 5 и 1. Системы счисления, в которых любое число получается путем сложения или вычитания базисных чисел, называются **аддитивными**. При таком представлении чисел правила сложения для небольших чисел очевидны и просты, однако если возникает необходимость выполнять операции сложения над большими числами или операции умножения и деления, то римская система счисления оказывается неудобной. В этой ситуации преимущество имеют позиционные системы счисления, хотя в них, как правило, представления чисел далеко не так просты и очевидны, как в римской. Систематичность представления, основанная на «позиционном весе» цифр, обеспечивает простоту выполнения операций умножения и деления.

В римской системе счисления каждый числовой знак в записи любого числа имеет одно и то же значение, т. е. значение числового знака не зависит от его расположения и записи числа. Таким образом, римская система счисления не является позиционной.

2.2.1. Позиционные системы счисления

Для изображения (или представления) чисел в настоящее время используются в основном позиционные системы счисления. Привычной для всех является десятичная система счисления. В этой системе для записи любых чисел используется только десять разных знаков (цифр): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Эти цифры введены для обозначения первых десяти последова-

тельных чисел, а следующие числа (начиная с 10 и т. д.) обозначаются уже без использования новых знаков (цифр). Тем самым сделан важный шаг в построении системы счисления: значение каждой цифры поставлено в зависимость от того места, где она стоит в изображении числа.

Таким образом, система называется **позиционной**, если значение каждой цифры (ее вес) изменяется в зависимости от ее положения (позиции) в последовательности цифр, изображающих число. Если это условие не выполняется, то система счисления является **непозиционной** (например, римская).

Десятичная позиционная система счисления основана на том, что десять единиц каждого разряда объединяются в одну единицу соседнего старшего разряда. Таким образом, каждый разряд имеет вес, равный степени 10. Например, в зависимости числа 343.32 цифра 3 повторена три раза, при этом самая левая цифра 3 означает количество сотен (ее вес равен 10^2); цифра 3, стоящая перед точкой, означает количество единиц (ее вес равен 10^0), а самая правая цифра 3 (после точки) — количество десятых долей единицы (ее вес равен 10^{-1}), так что последовательность цифр 343.32 представляет собой сокращенную запись выражения:

$$3 \times 10^2 + 4 \times 10^1 + 3 \times 10^0 + 3 \times 10^{-1} + 2 \times 10^{-2}.$$

Десятичная запись любого числа X в виде последовательности цифр:

$$a_n a_{n-1} \ldots a_1 a_0 a_{-1} \ldots a_{-m} \ldots \qquad (2.4)$$

основана на представлении этого числа в виде полинома:

$$X = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \ldots + a_1 \times 10^1 + \\ + a_0 \times 10^0 + a_{-1} \times 10^{-1} + \ldots + a_{-m} \times 10^{-m} \ldots, \qquad (2.5)$$

где каждый коэффициент a_i может быть одним из чисел, для обозначения которых введены специальные знаки. Запись числа X в виде (2.4) представляет собой просто перечисление всех коэффициентов этого полинома. Точка, отделяющая целую часть числа от дробной, служит для фиксации конкретных значений каждой позиции в этой последовательности цифр и является началом отсчета.

Число K единиц какого-либо разряда, объединяемых в единицу более старшего разряда, называется **основанием позиционной системы**, а сама система счисления называется ***K*-ичной**. Например, основанием десятичной системы счисления является число 10; двоичной — число 2; троичной — число 3 и т. д. Для записи произвольного числа в K-ичной системе счисления достаточно иметь K разных цифр a_i, $i = 1, ..., K$. Например, в троичной системе счисления любое число может быть выражено посредством цифр 0, 1, 2. Эти цифры служат для обозначения некоторых различных целых чисел, называемых **базисными**.

Числа можно записать как суммы степеней не только числа 10, но и любого другого натурального числа, большего единицы. Например, в Древнем Вавилоне использовалась система счисления с основанием 60. Деление часа на 60 минут, а минуты — на 60 секунд заимствовано именно из этой системы счисления. А то, что человечество выбрало в качестве основания системы счисления число 10, вероятно, связано с тем, что природа наделила людей десятью пальцами.

Запись произвольного числа X в K-ичной позиционной системе счисления основывается на представлении этого числа в виде полинома:

$$X = a_n K^n + a_{n-1} K^{n-1} + ... + a_1 K^1 + \\ + a_0 K^0 + a_{-1} K^{-1} + ... + a_{-m} K^{-m} + ..., \qquad (2.6)$$

где каждый коэффициент a_i может быть одним из базисных чисел и изображается одной цифрой. Как и в десятичной системе счисления, число X, представленное в K-ичной системе счисления, можно кратко записать в виде (2.4) путем перечисления всех коэффициентов полинома (2.6) с указанием позиционной точки. Последовательность цифр, стоящая в (2.4), является изображением числа X в K-ичной системе счисления. Базисные числа должны быть выбраны так, чтобы любое число X могло быть представлено в виде полинома (2.6). Обычно в качестве базисных чисел берутся последовательные целые числа от 0 до $K - 1$ включительно.

Арифметические действия над числами в любой позиционной системе счисления производятся по тем же принципам, что и

в десятичной системе, так как все они основываются на правилах выполнения действий над соответствующими полиномами. При этом нужно только пользоваться теми таблицами сложения и умножения, которые имеют место при данном основании K системы счисления.

Для указания того, в какой системе счисления записано число, условимся при его изображении основание системы счисления указывать в виде нижнего индекса при нем, например: 35.64_8.

2.2.2. Двоичная система счисления

Итак, в современной вычислительной технике, в устройствах автоматики и связи широко применяется двоичная система счисления. Это система счисления с наименьшим возможным основанием. В ней для изображения числа используются только две цифры: 0 и 1.

Произвольное число X в двоичной системе представляется в виде полинома:

$$X = a_n \times 2^n + a_{n-1} \times 2^{n-1} + \ldots + a_1 \times 2^1 + \\ + a_0 \times 2^0 + a_{-1} \times 2^{-1} + \ldots + a_{-m} \times 2^{-m} + \ldots, \qquad (2.7)$$

где каждый коэффициент a_i может быть либо 0, либо 1.

Примеры изображения чисел в двоичной системе счисления:

$1 = 1_2$	$5 = 101_2$	$9 = 1001_2$
$2 = 10_2$	$6 = 110_2$	$10 = 1010_2$
$3 = 11_2$	$7 = 111_2$	$0.5 = 0.1_2$
$4 = 100_2$	$8 = 1000_2$	$0.25 = 0.01_2$

Таблица сложения чисел в двоичной системе счисления:

$0 + 0 = 0$	$1 + 0 = 1$
$0 + 1 = 1$	$1 + 1 = 10$

Таблица умножения чисел в двоичной системе счисления:

$0 \times 0 = 0$	$1 \times 0 = 0$
$0 \times 1 = 0$	$1 \times 1 = 1$

Так как в двоичной системе счисления для изображения любых чисел используются только две цифры, то при построении ЭВМ можно применять лишь элементы, которые могут находиться только в двух состояниях (например, высокое или низкое напряжение в цепи тока, наличие или отсутствие электрического импульса и т. п.). Это обстоятельство, а также простота выполнения арифметических операций являются причиной того, что в большинстве современных ЭВМ используется двоичная система счисления.

2.2.3. Правила выполнения арифметических операций в двоичной системе счисления

Сложение. Операция сложения выполняется так же, как и в десятичной системе счисления. Переполнение разряда приводит к появлению единицы в следующем разряде.

$$0 + 0 = 0; \quad 0 + 1 = 1; \quad 1 + 1 = 10; \quad \begin{array}{r} 11110011 \\ + \quad 111011 \\ \hline 100101110 \end{array}$$

Вычитание. Поскольку большинство современных компьютеров располагает только одним аппаратным сумматором, с помощью которого реализуются арифметические операции, вычитание сводится к сложению с отрицательным числом:

$$15 - 8 = 15 + (-8).$$

Правила вычитания в двоичной системе счисления. Алгоритм операции вычитания путем сложения дополнительных кодов:

1. Преобразовать отрицательное число из формы со знаком в дополнительный код.

2. Выполнить операцию двоичного сложения над всеми разрядами, включая знаковый, игнорируя единицу переноса из самого высокого разряда.

3. Если знаковый разряд суммы равен единице (что означает получение отрицательного результата в форме дополнительного кода), необходимо перевести результат в знаковую форму, используя второе свойство дополнений.

$$13 - 15 = 13 + (-15);$$

1) $-15_{10} = 10001111 \rightarrow 1110000 + 1 \rightarrow 1110001 \rightarrow 11110001;$

2) $\begin{array}{r} 00001101 \\ +\ 11110001 \\ \hline 11111110 \end{array};$

3) $11111110 \rightarrow 0000001 + 1 \rightarrow 10000010 = 2_{10}.$

Таким образом, при выполнении операций сложения и вычитания арифметико-логическому устройству процессора приходится выполнять поразрядное сложение с переносом, инвертирование и проверку на знак двоичных чисел.

В тех случаях, когда необходимо произвести арифметические действия над числами больше 127, они размещаются уже не в одном, а в двух и более регистрах.

Умножение. Если наряду с перечисленными операциями выполнить операции сдвига, то с помощью сумматора можно выполнить и умножение, которое сводится к серии повторных сложений. Если цифра в нулевой позиции множителя равна единице, то множимое переписывается под соответствующими разрядами, умножение на последующие единицы приводит к сдвигу слагаемого влево на одну позицию. Если цифра множителя равна 0, то следующее слагаемое смещается на две позиции влево:

$$15_{10} \times 13_{10} = 195_{10} = 11000011_{2} =$$
$$= 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^1 + 1 \times 2^0 = 195_{10}.$$

Деление. При выполнении операции деления несколько раз производится операция вычитания. Поэтому предварительно следует найти дополнительный код делителя. Деление выполняется путем повторного вычитания и сдвига.

В качестве примера выполним деление числа 195 на 15, или, в двоичной системе, -11000011_{2} на 1111_{2}. Дополнительный код числа $1111 \rightarrow 11110001$. Поскольку по правилам деления каждое промежуточное делимое должно быть больше делителя, выбираем в качестве первого делимого число 11000, т. е. первые пять разрядов, и добавляем слева три нуля, дополняя делимое до восьми разрядов. Затем производим сложение его с дополнительным кодом делимого и заносим в результат единицу. Если следующее делимое после сноса очередной цифры будет

меньше делителя, то в результат заносится нуль и в делимое сносится еще одна цифра из исходного делимого.

$$
\begin{array}{r|l}
00011000011 & 1111 \\
\underline{+\,11110001} & \underline{1101} \\
00010010 & \\
\underline{+\,11110001} & \\
00001111 & \\
\underline{+\,11110001} & \\
00000000 &
\end{array}
$$

Делимое 111 на третьем шаге после сложения и сноски очередного разряда меньше делителя, поэтому записываем в результат нуль и сносим еще один разряд из оставшихся в делимом. После третьего шага результат сложения равен нулю, деление закончено.

Ответ: $00001101_2 = 13_{10}$.

2.2.4. Перевод чисел из одной системы счисления в другую

При решении задач с помощью ЭВМ исходные данные обычно задаются в десятичной системе счисления; в этой же системе, как правило, нужно получить и окончательные результаты. Так как в современных ЭВМ данные кодируются в основном в двоичных кодах, то, в частности, возникает необходимость перевода чисел из десятичной в двоичную систему счисления и наоборот.

При рассмотрении правил перевода чисел из одной системы счисления в другую ограничимся только такими системами счисления, у которых базисными числами являются последовательные целые числа от 0 до $P - 1$ включительно, где P — основание системы счисления.

Задача перевода заключается в следующем. Пусть известна запись числа x в системе счисления с каким-либо основанием P:

$$p_n p_{n-1} \cdots p_1 p_0 p_{-1} p_{-2} \cdots,$$

где p_i — цифры P-ичной системы ($0 \leqslant p_i \leqslant P - 1$). Требуется найти запись этого же числа x в системе счисления с другим основанием Q:

$$q_s q_{s-1} \cdots q_1 q_0 q_{-1} q_{-2} \cdots,$$

где q_i — искомые цифры Q-ичной системы ($0 \leqslant q_i \leqslant Q - 1$). При этом можно ограничиться случаем положительных чисел, так как перевод любого числа сводится к переводу его модуля и приписыванию числу нужного знака.

Перевод числа из десятичной системы счисления в двоичную. Перевод числа из десятичной системы в двоичную осуществляется отдельно для целой и дробной частей числа по следующим алгоритмам:

а) целое десятичное число делится нацело на основание 2, затем на 2 делятся последовательно все частные от целочисленного деления, до тех пор пока частное не станет меньше основания. В результат заносятся последнее частное и все остатки от деления, начиная с последнего (рис. 2.1):

$$227_{10} = 11100011_2;$$

б) десятичная дробь последовательно умножается на основание 2, причем сразу после каждой операции умножения полученная целая часть записывается в результат и в дальнейшем умножении не участвует. Количество операций умножения зависит от требуемой точности; например, $0.64_{10} = 0.10100011_2$:

$$\begin{array}{l} 0.64 \times 2 \\ 1.28 \times 2 \\ 0.56 \times 2 \\ 1.12 \times 2 \\ 0.24 \times 2 \\ 0.48 \times 2 \\ 0.96 \times 2 \\ 1.92 \times 2 \\ 1.84 \times 2 \end{array}$$

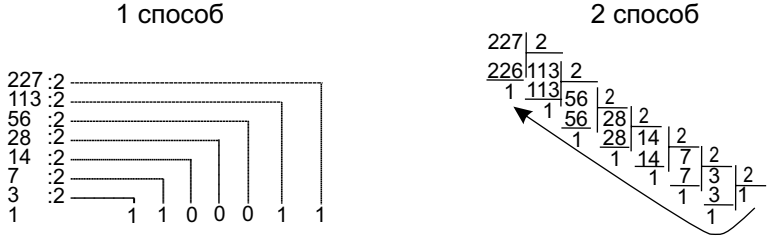


Рис. 2.1. Перевод числа из десятичной системы счисления в двоичную

Перевод числа из двоичной системы счисления в десятичную. Перевод числа из двоичной системы в десятичную можно осуществлять для целой и дробной частей числа по одному алгоритму путем вычисления суммы произведений цифры двоичного числа на вес ее знакоместа:

$$1110\,0011_2 = 1 \times 2^7 + 1 \times 2^6 + 1 \times 2^5 + 0 \times 2^4 + \\ + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = \\ = 128 + 64 + 32 + 2 + 1 = 227_{10};$$

$$0{,}10100001_2 = 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 0 \times 2^{-4} + \\ + 0 \times 2^{-5} + 0 \times 2^{-6} + 0 \times 2^{-7} + 0 \times 2^{-8} = \\ = 0.5 + 0.125 + 0.0078 + 0.0039 = 0.6367_{10}.$$

Представление в компьютере отрицательных чисел. Следует иметь в виду, что в памяти ПЭВМ двоичные числа хранятся в регистрах, состоящих из восьми ячеек, т. е. минимальное двоичное число, которое можно разместить в памяти, должно быть восьмиразрядным. При этом в незаполненные ячейки регистра (в старших разрядах) записываются нули.

В отличие от десятичной, в двоичной системе счисления отсутствуют специальные символы, обозначающие знак числа — положительный («+») или отрицательный («–»), поэтому для представления двоичных отрицательных чисел используются следующие две формы.

Форма значения со знаком: старший (левый) разряд метится как знаковый и содержит информацию только о знаке числа:

1 — число отрицательное;
0 — число положительное.

Остальные разряды отводятся под абсолютную величину числа:

$$-5_{10} = 0000\,0101_2;$$
$$-5_{10} = 1000\,0101_2.$$

Форма обратного дополнительного кода, перевод в которую производится по следующему алгоритму:

1. Инвертировать все разряды числа, кроме знакового разряда.
2. Прибавить единицу к полученному коду.
3. Восстановить единицу в знаковом разряде.

Преобразование числа $-5_{10} = 10000101 \rightarrow 1111010 + 1 \rightarrow$ $\rightarrow 1111011 \rightarrow 11111011$.

Компьютеры обычно строятся таким образом, чтобы отрицательные числа были представлены в дополнительном коде, поскольку это дает существенную экономию времени при выполнении с ними арифметических операций. Основные свойства дополнительных кодов:

— дополнительный код положительного числа — само число;

— преобразование дополнительного кода по приведенному алгоритму перевода приводит к первоначальному виду числа в знаковой форме.

2.2.5. Другие позиционные системы счисления

Неудобство использования двоичной системы счисления заключается в громоздкости записи чисел. Это неудобство не имеет существенного значения для ЭВМ. Однако если возникает необходимость кодировать информацию «вручную», например при составлении программы на машинном языке, то предпочтительнее пользоваться восьмеричной или шестнадцатеричной системами счисления (в силу их свойств, которые будут рассмотрены ниже).

В восьмеричной системе счисления базисными являются числа 0, 1, 2, 3, 4, 5, 6, 7. Запись любого числа в этой системе основывается на его разложении по степеням числа 8 с коэффициентами, которыми являются вышеуказанные базисные числа.

Например, десятичное число 83.5 в восьмеричной системе будет изображаться в виде 123.4. Действительно, эта запись по определению означает представление числа в виде полинома:

$$1 \times 8^2 + 2 \times 8^1 + 3 \times 8^0 + 4 \times 8^{-1} = 64 + 16 + 3 + 4/8 = 83.5.$$

В шестнадцатеричной системе счисления базисными являются числа от 0 до 15. Эта система отличается от рассмотренных тем, что в ней не хватает общепринятых (арабских) цифр для обозначения всех базисных чисел, поэтому приходится вводить в употребление новые символы. Обычно для обозначения первых десяти целых чисел (от 0 до 9) используются арабские цифры, а для следующих целых чисел (от 10 до 15) — буквенные обозначения из латинского алфавита: A, B, C, D, E, F.

Например, десятичное число 175.5 в шестнадцатеричной системе записывается как $AF.8_{16}$. Действительно: $10 \times 16^1 + 15 \times 16^0 + 8 \times 16^{-1} = 160 + 15 + 8/16 = 175.5$.

Соотношение различных систем счисления приведено в табл. 2.1.

Каждая тройка двоичных разрядов соответствует одной восьмеричной цифре, каждая четверка — шестнадцатеричной. Отсюда следует простота преобразований из двоичной системы счисления в восьмеричную и шестнадцатеричную, например:

$$11010011_2 = 1101\ 0011_2 = D3_{16};$$
$$11010011_2 = 011\ 010\ 011_2 = 323_8.$$

Таблица 2.1

Соотношение различных систем счисления

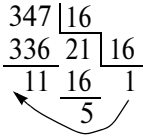
Системы счисления			
Десятичная	Двоичная	Восьмеричная	Шестнадцатеричная
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10

Если исходное количество бит не кратно 3 или 4, добавляются нули слева.

Обратное преобразование аналогично:

$$B9_{16} = 1011\ 1001_{2};$$
$$270_{8} = 10\ 111\ 000_{2}.$$

Перевод из десятичной системы счисления в *K*-ичную производится аналогично переводу в двоичную систему путем целочисленного деления десятичного числа на основание системы *K* до тех пор, пока частное не станет меньше основания. Так, перевод в шестнадцатеричную систему осуществляется следующим образом:



$$347_{10} = 15B_{16}.$$

Перевод из *m*-ичной системы в десятичную производится путем сложения произведений соответствующего десятичного эквивалента символа числа в *m*-ичной системе на вес *i*-го знакоместа.

Пример перевода из шестнадцатеричной системы счисления в десятичную:

$$15B_{16} = 1 \times 16^{2} + 5 \times 16^{1} + 11 \times 16^{0} = 256 + 80 + 11 = 347_{10}.$$

Тема 2.3

ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В ПАМЯТИ ЭВМ

2.3.1. Представление целых чисел без знака и со знаком

Введем для простоты сначала основные понятия на примере четырехбитовых машинных слов. Такой размер слова обеспечивает хранение десятичных чисел только от 0 до 15 и поэтому не представляет практического значения. Однако они менее

громоздки, а основные закономерности, обнаруженные на примере четырехбитовых слов, сохраняют силу для машинного слова любого размера.

Предположим, что процессор ЭВМ способен увеличивать (прибавлять 1) и дополнять (инвертировать) четырехбитовые слова. Например, результатом увеличения слова 1100 является 1101, а результатом дополнения этого слова — 0011. Рассмотрим слово 0000, представляющее десятичное число 0. В результате увеличения содержимое этого слова станет равным 0001, что соответствует десятичному числу 1. Продолжая последовательно увеличивать четырехбитовые слова, придем к ситуации, когда, увеличивая слово 1111 (которое представляет десятичное число 15), получим в результате слово 0000, т. е. 111 + 1 = 0000 (15 + 1 = 0); при этом получили неверную арифметическую операцию и вернулись в исходное состояние. Это произошло из-за того, что слово памяти может состоять только из конечного числа бит. Таким образом, числовая система ЭВМ является *конечной* и *циклической*.

Такой ситуации, приводящей к неверному арифметическому результату, можно избежать, если битовую конфигурацию 1111 принять за код для –1. Тогда 1110 интерпретируется как –2; 1101 — как –3 и т. д. до 1000 — как –8. Тем самым получили другую числовую систему — со знаком, содержащую как положительные, так и отрицательные числа. В этой системе половина четырехбитовых конфигураций, начинающаяся с единицы, интерпретируется как отрицательные числа, а другая половина, начинающаяся с нуля, — как положительные числа или нуль. Поэтому старший бит числа (третий по счету, если нумерацию бит начинать с нуля справа налево) называется **знаковым битом**. Числовая система со знаком также конечна и циклична, однако в этом случае арифметически неверный результат даст попытка увеличить число 8 на единицу. Преимущество введения числовой системы со знаком заключается в возможности представления как положительных, так и отрицательных чисел.

Если знаковый бит равен нулю, то значение числа легко вычисляется — игнорируется знаковый бит, а оставшиеся три бита интерпретируются как двоичный код десятичного числа. Например, слово 0110 представляет двоичное число 110, которое равно десятичному числу 6.

Для оценки отрицательного числа нужно изменить его знак. Рассмотрим четырехбитовое число k в системе со знаком. Тогда $-k = (-1 - k) + 1$, следовательно, для вычисления значения $-k$ необходимо вычесть k из -1 (т. е. из 1111) и затем прибавить единицу (т. е. 0001). Заметим, что операция вычитания всегда возможна, никогда не требует заёма и равнозначна операции инвертирования битов вычитаемого. Например: 1111 – 1011 = 0100; здесь в вычитаемом, равном 1011, единицы перешли в нули, а нуль — в единицу. Инвертирование бит в слове называется **дополнением до единицы**. Для определения отрицательного значения числа k надо к его дополнению до единицы прибавить единицу (согласно вышеприведенному равенству). Инвертирование бит в слове с добавлением единицы к младшему биту называется **дополнением до двух**. Например, требуется найти, какое число закодировано в слове 1001. Для этого сначала выполняем операцию инвертирования: 1001 → 0110, а затем к полученному результату прибавляем единицу: 0110 + 1 = 0111, что является двоичным кодом числа 7. Таким образом, значением 1001 является отрицательное число 7, т. е. –7.

2.3.2. Индикаторы переноса и переполнения

Рассмотрим более подробно ситуацию, приводящую при увеличении четырехбитового числа (т. е. при прибавлении к нему единицы) к неверному арифметическому результату и возникающую из-за конечности числовой системы ЭВМ. В числовой системе без знака такая проблема встает при увеличении слова 1111, при этом имеет место перенос единицы из знакового бита. В случае системы чисел со знаком перенос из старшего бита дает верный результат: 1111 + 0001 и 0000 (что правильно: –1 + + 1 = 0), но увеличение слова 0111 приводит к ошибочной ситуации: 0111 + 1 = 1000 (7 + 1 = –8), при этом имеет место перенос в знаковый бит.

В процессоре ЭВМ (устройстве, в котором выполняются арифметические операции) имеется два индикатора — индикатор переноса и индикатор переполнения. Каждый индикатор содержит один бит информации и может быть процессором установлен (в этом случае ему придается значение, равное еди-

нице) или сброшен (равен нулю). *Индикатор переноса* указывает на перенос из знакового бита, а *индикатор переполнения* — на перенос в знаковый бит. Таким образом, после завершения операции, в которой происходит перенос в старший бит, процессор устанавливает индикатор переполнения; если такого переноса нет, то индикатор переполнения сбрасывается. Индикатор переноса обрабатывается аналогичным образом.

Важно то, что после выполнения операции процессором ЭВМ сигнализирует о состоянии индикаторов, и их можно проверить. Если состояние индикаторов указывает на неправильный арифметический результат, то необходимо исправить эту ситуацию, т. е. пользователю ЭВМ предоставляется возможность контролировать правильность выполнения арифметических операций.

Пример 1. Рассмотрим сложение двух чисел: 0101 + 0011 = = 1000. В результате выполнения операции сложения произошел перенос в знаковый бит, а переноса из знакового бита не было. Таким образом, после завершения этой операции индикатор переноса будет сброшен, а индикатор переполнения установлен. Поэтому если рассматривать эти два числа как целые без знака, то результат является арифметически правильным, так как индикатор переноса сброшен. Если же рассматривать числа в системе со знаком, то бит переполнения показывает, что произошло изменение знака (перенос в знаковый бит есть, а переноса из него — нет), поэтому арифметический результат неправильный.

Пример 2. В результате сложения чисел 1101 + 0101 = 0010 происходит перенос в знаковый бит и из знакового бита. Поэтому будет установлен индикатор переноса, а индикатор переполнения сброшен. Следовательно, в системе чисел без знака результат является арифметически неправильным, а в системе чисел со знаком — правильным.

Пример 3. В результате выполнения операции вычитания: 1001 – 0011 = 1001 + (–0011) = 1001 + (1100 + 1) = 1001 + 1101 = = 0110 происходит перенос из знакового бита, а переноса в знаковый бит нет. Следовательно, индикатор переноса будет сброшен, а индикатор переполнения установлен; это указывает на то, что в данном примере в системе без знака результат арифметически правильный, а в системе со знаком — неправильный.

2.3.3. Представление символьной информации в ЭВМ

В отличие от обычной словесной формы, принятой в письменном виде, символьная информация хранится и обрабатывается в памяти ЭВМ в форме цифрового кода. Например, можно обозначить каждую букву числами, соответствующими ее порядковому номеру в алфавите: А — 01, Б — 02, В — 03, Г — 04, ..., Э — 30, Ю — 31, Я — 32. Точно так же можно договориться обозначать точку числом 33, запятую — числом 34 и т. д. Так как в устройствах автоматической обработки информации используются двоичные коды, обозначения букв надо перевести в двоичную систему. Тогда буквы будут обозначаться следующим образом: А — 000001, Б — 000010, В — 000011, Г — 000100, ..., Э — 011110, Ю — 011111, Я — 100000. При таком кодировании любое слово можно представить в виде последовательности кодовых групп, составленных из 0 и 1. Например, слово «ЭВМ» выглядит так: 011110 000011 001110.

При преобразовании символов (знаков) в цифровой код между множествами символов и кодов должно иметь место взаимноднозначное соответствие, т. е. разным символам должны быть назначены разные цифровые коды, и наоборот. Это условие является единственным необходимым требованием при построении схемы преобразования символов в числа. Однако существует ряд практических соглашений, принимаемых при построении схемы преобразования исходя из соображений наглядности, эффективности, стандартизации. Например, какое бы число ни назначили коду для знака 0 (не следует путать с числом 0), знаку 1 удобно назначить число, на единицу большее, чем код 0, и т. д. до знака 9. Аналогичная ситуации возникает и при кодировке букв алфавита: код для Б на единицу больше кода для А, а код для В — на единицу больше кода для Б и т. д. Таким образом, из соображений наглядности и легкости запоминания целесообразно множества символов, упорядоченных по какому-либо признаку (например, лексико-графическому), кодировать также с помощью упорядоченной последовательности чисел.

Другой важный момент при организации кодировки символьной информации — эффективное использование оперативной памяти ЭВМ. Так как общеупотребительными являют-

ся примерно 100 знаков (сюда, помимо цифр, букв русского и английского алфавитов, знаков препинания, арифметических знаков, входят знаки перевода строки, возврата каретки, возврата на шаг и т. п.), то для взаимнооднозначного преобразования всех знаков в коды достаточно примерно сотни чисел. Значение этого выбора заключается в том, что для размещения числа из этого диапазона в оперативной памяти достаточно одного байта, а не машинного слова. Следовательно, при такой организации кодировки достигается существенная экономия объема памяти.

При назначении кодов знакам надо также учитывать соглашения, касающиеся стандартизации кодировки. Можно назначить знаковые коды по своему выбору, но при этом возникнут трудности, связанные с необходимостью обмена информацией с другими организациями, использующими кодировку, отличную от нашей. В настоящее время существует несколько широко распространенных схем кодирования. Например, двоично-десятичный код BCD (Binary-Coded Decimal) используется для представления чисел, при котором каждая десятичная цифра записывается своим четырехбитовым двоичным эквивалентом. Этот код может оказаться полезным, когда нужно преобразовать строку числовых знаков, например строку из числовых знаков 2537 в число 2537, над которым затем будут производиться арифметические действия. Расширением кода BCD является EBCDIC (Extended Binary-Coded Decimal Interchange Code) — расширенный двоично-десятичный код обмена информацией, который преобразует как числовые, так и буквенные строки.

В ЭВМ применяется код ASCII (American Standart Code for Information Interchange) — Американский стандартный код обмена информацией. Этот код генерируется некоторыми внешними устройствами — принтером, алфавитно-цифровым печатающим устройством (АЦПУ) — и используется для обмена данными между ними и оперативной памятью ЭВМ. Например, когда нажимаем на терминале клавишу <*G*>, то в результате этого действия код ASCII для символа *G* (1000111) передается в ЭВМ. А если надо этот символ распечатать на АЦПУ, то его код ASCII должен быть послан на печатающее устройство.

Отечественной версией кода ASCII является код КОИ-7 (двоичный семибитовый код обмена информацией), который совпадает с ним, за исключением букв русского алфавита.

2.3.4. Форматы данных

Под **данными** будем понимать информацию, представленную в виде, пригодном для обработки автоматическими средствами, например в двоичном коде. Формат представления данных в памяти ЭВМ зависит от ее архитектуры.

Данные, обрабатываемые ЭВМ, подразделяют на три группы: логические коды, числа с фиксированной точкой и числа с плавающей точкой.

Представление логических кодов. Логические коды могут размещаться в отдельных байтах и в словах. Для их представления используются все разряды: для байта — от 0-го до 7-го, для слова — от 0-го до 15-го. Логическими кодами могут быть представлены символьные величины, числа без знака и битовые величины.

Символьные величины задаются в коде ASCII (КОИ-7), каждый символ занимает один байт, разряд 7 которого всегда содержит 0. Символы строки размещаются в последовательно-адресуемых байтах оперативной памяти. Например, символьная строка ABCDE (коды ASCII: A — 101_8, B — 102_8, C — 103_8, D — 104_8, E — 105_8), первый знак которой помещается в ячейку с адресом 1000 (адреса представлены в восьмеричной системе счисления), размещается в оперативной памяти следующим образом: числа без знака для *байта* имеют диапазон представления от 000 до 377_8, для *слова* — от 000000 до 177777_8; битовые величины задают значения отдельных разрядов байта или слова.

Представление чисел в формате с фиксированной точкой. **Фиксированной** называется такая точка, позиция которой фиксируется форматом представления чисел. Это такой способ разделения целой и дробной частей числа, при котором положение точки не меняется в ходе выполнения операций над данным числом. Числа, представленные в формате с фиксированной точкой, могут занимать один байт или слово. Если число с фиксированной точкой занимает один байт, то для его пред-

ставления используются разряды с 0-го по 6-й. Разряд 7 называется **знаковым**. При размещении числа с фиксированной точкой в слове для его представления используются разряды с 0-го по 14-й. Знак числа содержится в разряде 15. Значения знакового разряда: 0 — для положительных чисел; 1 — для отрицательных чисел.

Отрицательные числа в формате с фиксированной точкой представляются в дополнительном коде (посредством операции дополнения до двух).

Примеры представления чисел в формате с фиксированной точкой приведены в табл. 2.2.

Таблица 2.2

Представление чисел в формате с фиксированной точкой

Число	Байт		Слово	
	Восьмеричный код	Двоичный код	Восьмеричный код	Двоичный код
+5	005	00000101	000005	0000000000000101
+63	077	00111111	000077	0000000000111111
–5	373	11111011	177773	1111111111111011
–63	315	11001101	177715	1111111111001101
0	000	00000000	000000	0000000000000000

Диапазон представления чисел в формате с фиксированной точкой: для *байта* — от -128_{10} до $+127_{10}$; для *слова* — от -32768_{10} до $+32767_{10}$. При выполнении операций над числами, представленными в формате с фиксированной точкой, они масштабируются таким образом, чтобы каждое число лежало в интервале (–1, +1). Другими словами, в этом случае ЭВМ оперирует только с числами, по модулю не превосходящими единицу. При этом необходимо следить за тем, чтобы результат не получился большим, чем $2^k - 1$, где k — число разрядов, отведенных для представления в машине. Такая опасность есть при выполнении операций сложения и деления.

Опасность представляют также операции вычитания и умножения. При вычитании может получиться так, что разность станет числом, меньшим, чем представляется в машине, и ре-

зультат исчезнет. При многократном умножении (из-за того, что умножаются числа, меньшие единицы) может произойти то же самое. Поэтому при использовании формата представления чисел с фиксированной точкой приходится следить как за случаями возможного переполнения разрядной сетки машины, так и за случаями, связанными с появлением машинного нуля. Необходимость постоянно следить за тем, чтобы числа в машине не вышли за пределы интервала $(-1, +1)$, а также неизбежное в таких устройствах накопление абсолютной погрешности вычислений из-за перемасштабирования, при котором цифры младших разрядов (а именно в них накапливается абсолютная погрешность) передвигаются в старшие разряды, привели к тому, что в универсальных ЭВМ представление чисел в формате с фиксированной точкой практически перестало применяться. Оно сохраняется в специализированных вычислительных системах, где диапазон изменения чисел заранее проанализирован, в некоторых микропроцессорах и микроЭВМ.

Представление чисел в формате с плавающей точкой. **Плавающей** называется такая точка, позиция которой не фиксируется форматом числа. Любое вещественное число X, представленное в системе счисления с основанием N, можно записать в виде

$$X = \pm m N^{\pm p},$$

где m — мантисса, p — характеристика (или порядок) числа.

Если $|m| < 1$, то запись числа называется **нормализованной слева**. Следующие примеры показывают, как можно представить любое число в формате с плавающей точкой:

а) в десятичной системе счисления:

$$372.95 = 0.37295 \times 10^3;$$
$$25 = 0.025 \times 10^3 = 0.25 \times 10^2;$$
$$0.0000015 = 0.15 \times 10^{-5} = 0.015 \times 10^{-4};$$

б) в двоичной системе счисления:

$$11010{,}1101 = 0{,}0110101101 \times 2^6 = 0{,}110101101 \times 2^5;$$
$$0{,}011011 = 0{,}11011 \times 2^{-1};$$
$$0{,}1 = 0{,}1 \times 2^0$$

(здесь порядок определяет, на сколько разрядов необходимо осуществить сдвиг относительно запятой).

Из этих примеров видно, что представление чисел в формате с плавающей точкой неоднозначно. В ЭВМ с целью минимизации погрешности при вычислениях и эффективного использования памяти применяется процедура нормализации справа.

Число называется **нормализованным справа**, если после запятой в мантиссе стоит не нуль. Например, числа $0{,}00076_{10}$ и $0{,}00011_2$, представленные соответственно в виде $0{,}076 \times 10^{-2}$ и $0{,}011 \times 2^{-2}$, не являются нормализованными справа, а эти же числа, представленные соответственно в виде $0{,}76 \times 10^{-3}$ и $0{,}11 \times 2^{-3}$, являются таковыми.

В дальнейшем под нормализацией числа будем понимать нормализацию справа.

Нормализованное число одинарной точности, представленное в формате с плавающей точкой, записывается в память следующим образом: знак числа — в бите 15 первого слова (0 — для положительных и 1 — для отрицательных чисел); порядок размещается в битах 7—14 первого слова, а мантисса занимает остальные 23 бита в двух словах. Нормализованное число двойной точности записывается в четыре слова памяти и отличается от представления чисел с одинарной точностью только тем, что продолжение мантиссы размещается в следующих за первым словом трех последовательных словах памяти, а всего под мантиссу в этом случае отводится 55 бит.

Порядок числа, представленного в формате с плавающей точкой, изменяется в диапазоне от $-128_{10}(200_8)$ до $+127_{10}(177_8)$ и запоминается увеличенным на $128_{10}(200_8)$. Такой способ представления порядка называется **смещенным**.

Следует иметь в виду, что, хотя для мантиссы отведено 23 разряда для чисел одинарной точности и 55 разрядов — для чисел двойной точности, в операциях участвует 24 и 56 разрядов соответственно, так как старший разряд мантиссы нормализованного числа не хранится, т. е. имеет место так называемый скрытый разряд. Однако при аппаратном выполнении операций этот разряд автоматически восстанавливается и учитывается. Порядок числа также учитывает скрытый старший разряд мантиссы.

Также заметим, что нормализованная мантисса в двоичной системе счисления всегда представляется десятичным числом m, лежащим в диапазоне $0{,}5 \leqslant m < 1$.

Примеры представления чисел в формате с плавающей точкой:

$$1)\ 0{,}1_{10} = 0{,}0\ (6314)_8 = 0{,}000\ (1100)_2 = 0,\ (1100)_2 \times 2^{-3};$$
$$-3 = (-3 + 200)_8 = 175_8 = 01111101_2.$$

Заметим, что в этом примере мантисса представлена бесконечной периодической дробью, поэтому последний учитываемый разряд мантиссы округляется;

$$2)\ -49{,}5_{10} = -61{,}4_8 = -110001{,}100_2 = -0{,}1100011_2 \times 2^6;$$
$$6_{10} = (6 + 200)_8 = 206_8 = 10000110_2.$$

При выполнении арифметических операций над числами, представленными в формате с плавающей точкой, надо отдельно выполнять их для порядков и мантисс. При алгебраическом сложении чисел надо сначала уравнять порядки слагаемых. При умножении порядки надо складывать, а мантиссы — перемножать. При делении из порядка делимого вычитают порядок делителя, а над мантиссами совершают обычную операцию деления. После выполнения операций, если это необходимо, проводят нормализацию результата, что влечет изменение порядков, так как каждый сдвиг на один разряд влево соответствует уменьшению порядка на единицу, а сдвиг вправо — увеличению его на единицу. Введение термина «плавающая точка» как раз и объясняется тем, что двоичный порядок, определяющий фактическое положение точки в изображении числа, корректируется после выполнения каждой арифметической операции, т. е. точка в изображении числа «плавает» (изменяется ее положение) по мере изменения данной величины. А в изображении чисел, представленных в формате с фиксированной точкой, она жестко зафиксирована в определенном месте.

Арифметические операции с числами, представленными в формате с плавающей точкой, намного сложнее таких же операций для чисел, представленных в формате с фиксированной точкой. Но зато плавающая точка позволяет производить операции масштабирования автоматически в самой машине

и избавляет от накопления абсолютной погрешности при вычислениях (хотя не избавляет от накопления относительной погрешности).

2.3.5. Представление цветной и графической информации

Последовательностями нулей и единиц можно закодировать и графическую информацию. Различают три вида компьютерной графики: растровую, векторную и фрактальную.

Рассмотрим наиболее часто используемую при разработке электронных (мультимедийных) и полиграфических изданий **растровую графику**. Основным элементом растрового изображения является **точка**, или **пиксель**.

Для кодирования любого изображения нужно разбить его на точки и цвет каждой точки закодировать. Например, черно-белую картинку можно закодировать, используя два бита: 11 — белый цвет, 10 — светло-серый, 01 — темно-серый и 00 — черный.

Для кодирования 256 различных цветов требуется 8 бит. Однако этого недостаточно для кодирования полноцветных изображений живой природы. Человеческий глаз может различать десятки миллионов цветовых оттенков. В современных компьютерах для кодирования цвета одной точки используется три байта.

Каждый цвет представляет собой комбинацию трех основных цветов: красного, зеленого и синего. Первый байт определяет интенсивность красной составляющей, второй — зеленой, третий — синей.

Белый цвет кодируется полными тремя байтами (255, 255, 255, или — в двоичной системе — 111111111, 11111111, 11111111). Черный цвет — отсутствие всех цветов (0, 0, 0). Красный цвет может быть темным (120, 0, 0) или ярко-красным (255, 0, 0). Такая система кодирования цветной графической информации называется **системой RGB** (Red, Green, Blue) и обеспечивает однозначное определение 16,5 млн (2^{24}) различных цветов и оттенков.

Качество графического изображения зависит от количества точек (пикселей) на единице площади. Этот параметр называется **разрешением** и измеряется в точках на дюйм — dpi.

Расчет объема графической информации сводится к вычислению произведения количества точек на изображении на количество разрядов, необходимых для кодирования цвета одной точки.

Например, для цветной картинки, составленной из 256 цветов в графическом режиме монитора 640×480, требуется объем видеопамяти, равный

$$8 \times 640 \times 480 = 2\,457\,600 \text{ бит} = 307\,200 \text{ байт} = 300 \text{ Кбайт}.$$

Тема 2.4

АЛГЕБРА ЛОГИКИ. ОСНОВНЫЕ ЛОГИЧЕСКИЕ ОПЕРАЦИИ. СЛОЖНЫЕ ВЫСКАЗЫВАНИЯ. ПОСТРОЕНИЕ ТАБЛИЦ ИСТИННОСТИ СЛОЖНЫХ ВЫСКАЗЫВАНИЙ

2.4.1. Понятие об алгебре высказываний

Великий немецкий математик Готфрид Лейбниц (1646—1716) разработал в 1666 г. (в двадцатилетнем возрасте) общий метод, позволяющий свести любую мысль к точным формальным высказываниям. Это открыло возможность перевести логику (Лейбниц называл ее законами мышления) из царства слов в царство математики, где отношения между объектами и высказываниями точны и определенны. Таким образом, Лейбниц стал основателем формальной логики. Он занимался исследованием двоичной системы счисления, наделяя ее неким мистическим смыслом: цифру 1 он ассоциировал с Богом, а 0 — с пустотой. Из этих двух цифр, по мнению Лейбница, произошло все и с их помощью можно выразить любое математическое понятие. Лейбниц первым высказал мысль о том, что двоичная система может стать универсальным логическим языком.

Что такое алгебра логики. Лейбниц мечтал о построении универсальной науки. Он хотел выделить простейшие понятия, с помощью которых по определенным правилам можно сформулировать понятия любой сложности; мечтал о создании универсального языка, на котором можно было бы записывать любые мысли в виде математических формул; думал о машине,

которая могла бы выводить теоремы из аксиом, о превращении логических утверждений в арифметические. В 1673 г. Лейбниц создал новый тип арифмометра — механический калькулятор, который не только складывает и вычитает числа, но и умножает, делит, возводит в степень, извлекает квадратные и кубические корни.

В 1847 г. английский математик Джордж Буль (1815—1864) создал универсальный логический язык. Буль разработал исчисление высказываний (впоследствии названное в его честь **булевой алгеброй)**, которое представляет собой формальную логику, переведенную на строгий язык математики. Формулы булевой алгебры внешне похожи на формулы знакомой нам алгебры, причем это сходство не только внешнее, но и внутреннее. Булева алгебра — вполне равноправная алгебра, подчиняющаяся своду принятых при ее создании законов и правил. Она является системой обозначений, применимой к любым объектам — числам, буквам и предложениям. Пользуясь этой системой, можно закодировать любые утверждения, истинность или ложность которых нужно доказать, а затем манипулировать ими, подобно обычным числам в математике.

Огромную роль в распространении булевой алгебры и ее развитии сыграл американский философ, логик, математик и естествоиспытатель Чарлз Пирс (1839—1914).

Объект рассмотрения в алгебре логики — так называемые *высказывания,* т. е. любые утверждения, о которых можно сказать, что они либо истинны, либо ложны. Например: «Сургут — город в России», «9 — четное число»; первое высказывание истинно, а второе — ложно.

Таким образом, высказывание — это любое утверждение, которое может быть либо истинным, либо ложным, но не может быть истинным и ложным одновременно. В математической логике понятие «высказывание» определяется как повествовательное предложение. В информатике это понятие сужается до определения: **высказывание** — это логическое выражение. Над высказываниями возможны определенные операции.

Итак, раздел математической логики, изучающий высказывания и операции над ними, называется **булевой алгеброй**. Булева алгебра — частный случай алгебры логики.

2.4.2. Основные логические операции

Из простых логических высказываний образовываются сложные высказывания. Для этого используются союзы: И; ИЛИ; ЕСЛИ...ТО; НЕ.

Истинное значение высказывания, полученного с помощью более простых высказываний, полностью определяется истинными значениями исходных высказываний. Поэтому каждой логической операции соответствует функция, принимающая значения 1 или 0, аргументы которой также принимают значения 1 или 0. Такие функции и называются **логическими функциями**.

Рассмотрим основные логические операции.

Конъюнкция (логическое умножение, И) двух высказываний A и B представляет собой сложное высказывание, которое истинно тогда и только тогда, когда истинны оба составляющих его высказывания A и B. Обозначение конъюнкции: 1) $A \wedge B$ (читается: A и B); 2) символ «·»; 3) между перемножаемыми высказываниями знак отсутствует: $A \wedge B = A \cdot B = AB$; 4) &.

Значение истинности логического произведения $A \wedge B$ в зависимости от значений истинности высказываний A и B определяется следующими соотношениями:

$$0 \wedge 0 = 0; \quad 0 \wedge 1 = 0; \quad 1 \wedge 0 = 0; \quad 1 \wedge 1 = 1. \qquad (2.8)$$

Дизъюнкция (логическое сложение, ИЛИ) двух высказываний A и B является сложным высказыванием, которое истинно, когда хотя бы одно из слагаемых A и B истинно. Обозначение дизъюнкции: 1) $A \vee B$ (читается: A или B); 2) в виде матрицы

$$A \vee B = \begin{vmatrix} A \\ B \end{vmatrix}.$$

Значение истинности логического сложения $A \vee B$ в зависимости от значений истинности высказываний A и B определяется следующими соотношениями:

$$0 \vee 0 = 0; \quad 0 \vee 1 = 1; \quad 1 \vee 0 = 1; \quad 1 \vee 1 = 1. \qquad (2.9)$$

Отрицание (НЕ). Отрицанием высказывания A является сложное высказывание A, которое ложно, когда A истинно, и истинно, когда A ложно. Обозначение отрицания: 1) $\overline{A}$ (читается: не A); 2) A'.

Значение истинности отрицания высказывания А определяется следующими соотношениями:

$$1' = 0; \quad 0' = 1. \tag{2.10}$$

Эквивалентность (равнозначность) двух высказываний А и В обозначается символом «⇔» (иногда — «~»). Значение истинности эквивалентных высказываний А и В определяется в зависимости от значений истинности исходных высказываний по следующим соотношениям:

$$0 \Leftrightarrow 0 = 0; \quad 0 \Leftrightarrow 1 = 1; \quad 1 \Leftrightarrow 0 = 1; \quad 1 \Leftrightarrow 1 = 1. \tag{2.11}$$

Импликация двух высказываний обозначается символом «>» (если А, то В) или стрелкой: «→».

2.4.3. Построение таблиц истинности сложных высказываний

В булевой алгебре существует понятие «значение истинности высказывания». Если высказывание истинно, то его значению придается символ 1, а если ложно, — символ 0.

Логические функции могут задаваться таблицами, часто называемыми **таблицами истинности** (табл. 2.3).

Таблица 2.3

Таблица истинности

A	B	$\bar{A}$	$A \wedge B$	$A \vee B$	$A \Leftrightarrow B$	$A \rightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	0	1
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Как читается данная таблица? В качестве примера прочитаем ее верхнюю строку: «Если переменные А и В принимают значения ЛОЖЬ, то отрицание А принимает значение ИСТИНА; выражение (А и В) принимает значение ЛОЖЬ; выражение

(А или В) принимает значение ЛОЖЬ; выражение (А эквивалентно В) принимает значение ИСТИНА; выражение (если А, то В) принимает значение ИСТИНА».

Рассмотренные логические операции тесно связаны с повседневной жизнью. Например, конъюнкцию (логическую функцию И) можно проиллюстрировать на примере работы стереопроигрывателя. Следующее сложное высказывание о стереоэффекте истинно только при условии истинности обоих простых высказываний: «Если работают оба канала стереопроигрывателя, то стереоэффект есть».

Дизъюнкция (логическая функция ИЛИ) может быть показана на примере возможности полета двухмоторного самолета. Это сложное высказывание истинно при истинности хотя бы одного из простых высказываний: «Если работает хотя бы один из двух двигателей — левый или правый, двухмоторный самолет может лететь».

Логическая операция отрицания (НЕ) может быть представлена следующим примером: «Если фотопленка не засвечена, то фотографировать можно, а если фотопленка засвечена, то фотографировать нельзя». Операция НЕ — это функция только одного простого высказывания — состояния фотопленки.

Тема 2.5

ОСНОВНЫЕ ЗАКОНЫ ПРЕОБРАЗОВАНИЯ АЛГЕБРЫ ЛОГИКИ

2.5.1. Закономерности логических операций

В алгебре логики имеют место несколько законов. Из них наиболее часто применяются следующие: сочетательный, переместительный, распределительный, закон двойственности (закон инверсий).

Сочетательный (синоним: **ассоциативный** — зависимость между двумя высказываниями) **закон** для конъюнкции и дизъюнкции соответственно:

— для конъюнкции:

$$A \wedge (B \wedge C) = (A \wedge B) \wedge C = A \wedge B \wedge C; \qquad (2.12)$$

— для дизъюнкции:

$$A \vee (B \vee C) = (A \vee B) \vee C = A \vee B \vee C. \qquad (2.13)$$

Переместительный (синоним: **коммутативный** — свойство операции не изменять значения при перестановке высказываний) **закон**:

— для конъюнкции:

$$A \wedge B = B \wedge A; \qquad (2.14)$$

— для дизъюнкции:

$$A \vee B = B \vee A. \qquad (2.15)$$

Правила (2.12) — (2.15) определяют конъюнкцию и дизъюнкцию в отдельности.

Распределительный (синоним: **дистрибутивный**) **закон**:

— для конъюнкции относительно дизъюнкции:

$$A \wedge (B \wedge C) = (A \wedge B) \wedge (A \wedge C); \qquad (2.16)$$

— для дизъюнкции относительно конъюнкции:

$$A \vee (B \vee C) = (A \vee B) \vee (A \vee C). \qquad (2.17)$$

Правила (2.16) и (2.17) выдают связь между конъюнкцией и дизъюнкцией совместно. Функции конъюнкции и дизъюнкции обладают свойствами, аналогичными свойствам операций умножения и сложения. При этом рассмотренные законы обладают «симметрией» в том смысле, что из любого закона для дизъюнкции (конъюнкции) можно получить тот же закон путем замены знаков дизъюнкции на знаки конъюнкции (дизъюнкции) и наоборот.

Закон двойственности, или **закон инверсий**, позволяет заменять отрицание конъюнкции дизъюнкцией отрицаний и отрицание дизъюнкции — конъюнкцией отрицаний:

$$(A \wedge B)' = A' \vee B'; \qquad (2.18)$$

$$(A \vee B)' = A' \wedge B'. \qquad (2.19)$$

Логическое сложение по модулю 2 (синонимы: ***разноименность, отрицание равнозначности***) высказываний A и B обозначается символом «+». Значение A и B определяется в зависимости от

значений истинности исходных высказываний по следующим соотношениям:

$$0 + 0 = 0;\quad 0 + 1 = 1;\quad 1 + 0 = 1;\quad 1 + 1 = 1. \qquad (2.20)$$

Операцию логического сложения по модулю 2 иногда называют строгой дизъюнкцией и обозначают символом «⊕». В этом случае связка высказывания А и В понимается в смысле «либо... либо», а не в смысле «или». Из соотношения (2.11) видно, что строго разделительное суждение $A \wedge B$ истинно лишь в двух случаях: когда А ложно, а В истинно; когда А истинно, а В ложно.

Для логической операции отрицания равнозначности имеют место переместительный и сочетательный законы, а также распределительный закон относительно операции конъюнкции:

$$A + B = B + A; \qquad (2.21)$$

$$A + (B + C) = (A + B) + C; \qquad (2.22)$$

$$A \wedge (B + C) = (A \wedge B) + (A \wedge C) \qquad (2.23)$$

и соотношения:

$$A \wedge A \wedge ... \wedge A = A; \qquad (2.24)$$

$$A \vee A \vee ... \vee A = A. \qquad (2.25)$$

2.5.2. Решение логических задач с помощью алгебры логики

Булева алгебра допускает множество логических операций, но всего трех из них — И, ИЛИ, НЕ — достаточно для того, чтобы выразить любые логические операции, или функции.

Так, с помощью одного элемента ИЛИ на два входа, двух элементов И на два входа и одного элемента НЕ можно построить логическую схему двоичного полусумматора (рис. 2.2), способного осуществлять операцию двоичного сложения двух одноразрядных двоичных чисел (т. е. выполнять правила двоичной арифметики):

$$0 + 0 = 0;\quad 0 + 1 = 1;\quad 1 + 0 = 1;\quad 1 + 1 = 0.$$

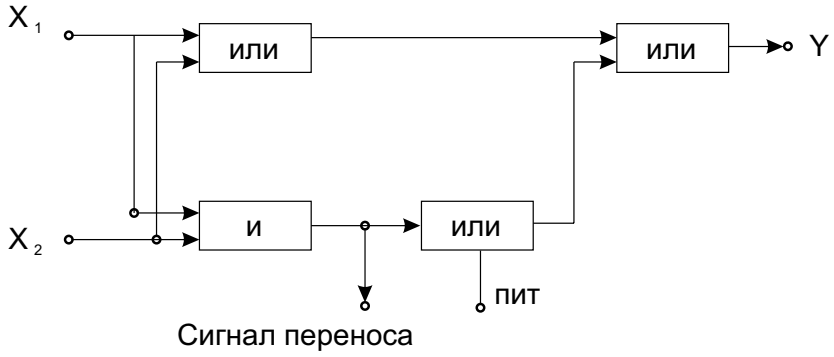


Рис. 2.2. Логическая схема двоичного полусумматора

При этом полусумматор выделяет бит переноса. Однако такая схема не содержит третьего входа, на который можно подавать сигнал переноса от предыдущего разряда суммы двоичных чисел. Поэтому полусумматор используется только в младшем разряде логической схемы суммирования многоразрядных двоичных чисел, где не может быть сигнала переноса от предыдущего двоичного разряда. Полный двоичный сумматор (рис. 2.3) складывает два многоразрядных двоичных числа с учетом сигналов переноса от сложения в предыдущих двоичных разрядах.

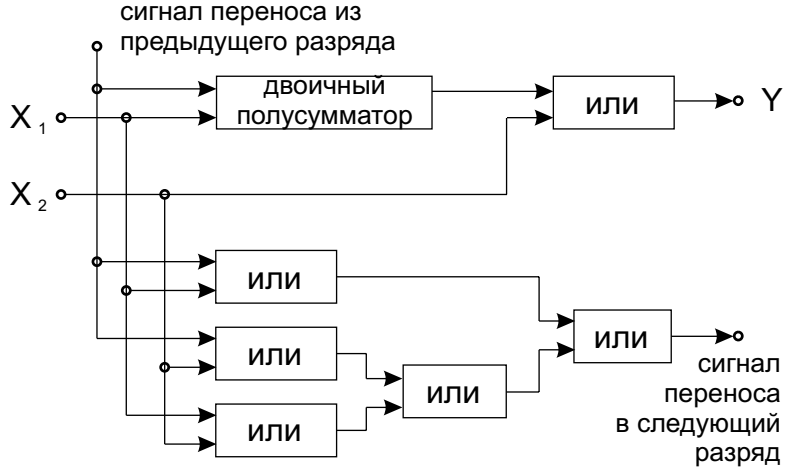


Рис. 2.3. Логическая схема двоичного сумматора

Соединяя двоичные сумматоры в каскад, можно получить логическую схему сумматора для двоичных чисел с любым числом разрядов. С некоторыми изменениями эти логические схемы применяются для вычитания, умножения и деления двоичных чисел. С их помощью построены арифметические устройства современных компьютеров.

Все описанные логические схемы являются однотактными. Значения их выходов однозначно определяются значениями их входов. Фактор времени в них отсутствует. Наряду с ними существуют многотактные логические схемы, в которых значения их выходов определяются не только значениями их входов, но и их состоянием в предыдущем такте. Фактор времени и определяется этими тактами. К таким логическим схемам относятся схемы памяти (рис. 2.4). Они строятся с помощью обратной связи с выхода на вход логической схемы.

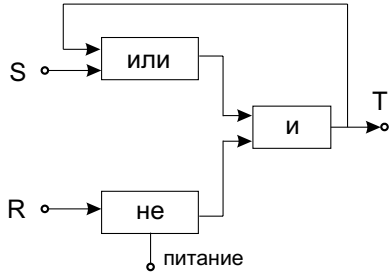


Рис. 2.4. Логическая схема оперативной памяти

Даже простейшая логическая схема памяти на один бит обязательно содержит обратную связь с выхода на вход. Она состоит из трех логических элементов, реализующих логические функции НЕ (отрицание), И, ИЛИ.

В этой схеме с помощью обратной связи образуется замкнутая цепь с выхода на вход для запоминания входного сигнала. Эта цепь сохраняется после снятия входного сигнала неограниченное время, вплоть до появления сигнала стирания.

Такая схема памяти имеет еще и другое название: **триггер с раздельными входами —** для запоминания (*S*) и стирания (*R*). Широко используется в вычислительной технике и **триггер со счетным входом**. Он имеет только один вход и один выход. Такая схема осуществляет деление на 2, т. е. состояние ее выхода изменяется только после подачи подряд двух входных импульсов. Соединяя триггеры со счетным входом в последовательный каскад, можно осуществлять деление на 2, 4, 8, 16, 32, 64 и т. д.

Схема оперативной памяти играет важную роль при построении систем управления машинами повышенной опаснос-

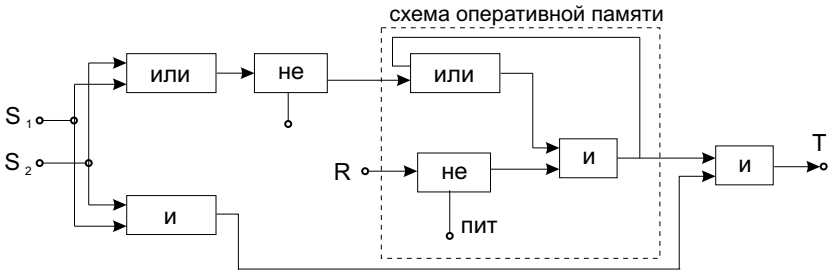


Рис. 2.5. Логическая схема безопасного двуручного управления машинами повышенной опасности

ти, такими, как прессы и одноножевые бумагорезательные машины («гильотины») (рис. 2.5). Чтобы обезопасить руки оператора, такие машины строят с системами двуручного управления. Подобные системы заставляют оператора держать обе руки на кнопках управления во время каждого рабочего цикла машины. Это исключает попадание рук в опасную зону, где происходит прессование детали или резание стопы бумаги.

Приведенный пример еще раз показывает, насколько, казалось бы, абстрактная наука алгебра логики близка к практической жизни. Она позволяет решать самые разные задачи управления.

Входные и выходные сигналы электромагнитных реле, подобно высказываниям в булевой алгебре, также принимают только два значения. Когда обмотка обесточена, входной сигнал равен нулю, а если по обмотке протекает ток, входной сигнал равен единице. Когда контакт реле разомкнут, выходной сигнал равен нулю, а если контакт замкнут, выходной сигнал равен единице.

Именно это сходство между высказываниями в булевой алгебре и поведением электромагнитных реле заметил известный физик Пауль Эренфест. Еще в 1910 г. он предложил использовать булеву алгебру для описания работы релейных схем в телефонных системах. По другой версии идея использования булевой алгебры для описания электрических переключательных схем принадлежит Ч. Пирсу. В 1936 г. основатель современной теории информации Клод Шеннон в своей докторской диссертации объединил двоичную систему счисления, математическую логику и электрические цепи.

Связи между электромагнитными реле в схемах удобно обозначать с помощью логических операций НЕ, И, ИЛИ, повторения (ДА) и т. д. Например, последовательное соединение контактов реле реализует логическую операцию И, а параллельное соединение этих контактов — логическую операцию ИЛИ. Аналогично выполняются операции И, ИЛИ, НЕ в электронных схемах, где роль реле, замыкающих и размыкающих электрические цепи, выполняют бесконтактные полупроводниковые элементы — транзисторы, созданные в 1947—1948 гг. американскими учеными Джоном Бардином, Уильямом Шокли и Уолтером Браттейном.

В современных компьютерах микроскопические транзисторы в кристалле интегральной схемы сгруппированы в системы «вентилей», выполняющих логические операции над двоичными числами. Так, с их помощью построены описанные выше двоичные сумматоры, позволяющие складывать многоразрядные двоичные числа, производить вычитание, умножение, деление и сравнение чисел между собой. Логические «вентили», действуя по определенным правилам, управляют движением данных и выполнением инструкций в компьютере.

Тема 2.6

ЛОГИЧЕСКИЕ ОСНОВЫ ЭВМ. ФУНКЦИОНАЛЬНЫЕ СХЕМЫ ЛОГИЧЕСКИХ УСТРОЙСТВ

Главной причиной, по которой ЭВМ используют не привычную десятичную, а двоичную систему счисления, является то, что в природе встречается множество явлений с двумя устойчивыми состояниями, например: «включено — выключено», «есть напряжение — нет напряжения», «ложное высказывание — истинное высказывание». В то же время явлений с десятью устойчивыми состояниями в природе нет. Почему же десятичная система так широко распространена? Повторим: потому, что у человека на двух руках десять пальцев и их удобно использовать для простого устного счета. Но в ЭВМ гораздо проще использовать двоичную систему счисления всего с двумя устойчивыми состояниями элементов и простейшими таблицами

сложения и умножения. В современных ЭВМ двоичная система используется не только для записи чисел, над которыми нужно производить вычислительные операции, но и для записи самих команд этих вычислений и даже целых программ операций. При этом все вычисления и операции сводятся в компьютере к простейшим арифметическим действиям над двоичными числами.

Таким образом, в двоичной системе (ее часто называют **двоичным кодом**) с помощью двух цифр можно представить любое число. Простое написание числа в двоичной системе, в отличие от десятичной, получается длиннее. Чем больше основание системы счисления, тем запись числа короче, зато таблицы сложения и умножения — длиннее.

Так, таблицы сложения и умножения чисел в двоичной системе счисления — самые короткие из всех возможных; напомним их:

— таблица сложения в двоичной системе счисления:

$0 + 0 = 0$; $0 + 1 = 1$; $1 + 0 = 1$; $1 + 1 = 0$ и 1 переносится в следующий старший двоичный разряд;

— таблица умножения в двоичной системе счисления:

$0 \times 0 = 0$; $0 \times 1 = 0$; $1 \times 0 = 0$; $1 \times 1 = 1$.

Подводя итоги использования алгебры логики в современной ВТ, рассмотрим сводную таблицу логических операций над высказыванием, которые часто называют **логическими связками**.

Логические связки — это операции, преобразующие тот или иной набор высказываний в более сложные высказывания. То есть логические связки — это *логические операции* над высказываниями, рассматриваемыми целиком, без учета их внутренней структуры. В формализованных языках логические связки выполняют функции, аналогичные функциям союзов и союзных слов в естественных языках. Так, *отрицание, конъюнкция, дизъюнкция, импликация, эквивалентность* истолковываются соответственно как частица *не*, союз *и*, союз *или*, союз *если...то*, оборот *тогда и только тогда*.

Правда, соответствие между логическими связками и союзными словами естественного языка не является вполне адекватным, точным, так как в формализованных языках игнорируются любые смысловые оттенки высказываний и прини-

маются во внимание лишь их логические значения «истинно» и «ложно». Кроме того, не для всех логических связок существуют соответствующие союзы и союзные слова естественного языка. Различные подходы к истолкованию, уточнению смысла союзов и союзных слов естественного языка, особенно союзов *или* и *если...то,* явились одной из причин развития, наряду с классическими логиками, ряда *классических логик*. В *алгебре логики* логические связки рассматривают как *булевы функции,* т. е. операции на множестве из двух значений («истинно» и «ложно» или 1 и 0). Константы 0 и 1 рассматриваются как нульместные логические связки. Основной одноместной логической связкой является *отрицание*. Существует 16 двухместных (бинарных) логических связок. Только 10 из них существенно зависят от обоих аргументов (табл. 2.4), остальные шесть равнозначны нульместным или одноместным логическим связкам.

Таблица 2.4

Логические связки

Обозначение логической связки	Другие обозначения и представления	Таблица истинности					Название связки	Как читать
		x	0	0	1	1		
		y	0	1	0	1		
$x \,\&\, y$	$x \wedge y$ $x \cdot y$ xy		0	0	0	1	Конъюнкция, логическое произведение, логическое И, функция совпадения	x и y
$x \vee y$			0	1	1	1	Дизъюнкция, логическая сумма, логическое ИЛИ, функция разделения	x или y; либо x, либо y, либо (x и y)
$x \rightarrow y$	$x \supset y$		1	1	0	1	Материальная импликация	если x, то y; x влечет y; x имплицирует y
$x \leftrightarrow y$	$x \equiv y$ $x \sim y$		1	0	0	1	Эквивалентность, функция равнозначности	x тогда и только тогда y, когда x эквивалентен y

Обозначение логической связки	Другие обозначения и представления	Таблица истинности					Название связки	Как читать
		x	0	0	1	1		
		y	0	1	0	1		
$x + y$	$x \vee y$ $\neg(x \leftrightarrow y)$ $\neg(x \equiv y)$		0	1	1	0	Сумма по модулю 2, разделительная дизъюнкция, отрицание эквивалентности, функция неравнозначности	либо x, либо y; x не эквивалентен y
$x \,/\, y$	$\overline{x \wedge y}$ $\neg(x \,\&\, y)$ $x \vee y$		1	1	1	0	Штрих Шеффера, отрицание конъюнкции, антиконъюнкция	x и y несовместны; неверно, что x и y
$x \downarrow y$	$\overline{x \vee y}$ $\neg(x \vee y)$ $\bar{x} \wedge \bar{y}$		1	0	0	0	Стрелка Пирса, отрицание дизъюнкции, антидизъюнкция	ни x, ни y
$x \not\rightarrow y$	$x \not\supset y$ $\neg(x \rightarrow y)$ $\neg(x \supset y)$ $x \,\&\, \bar{y}$		0	0	1	0	Отрицание материальной импликации, материальная антиимпликация	x, но не y; неверно, что x влечет y
$x \leftarrow y$	$x \subset y$ $y \rightarrow x$ $y \supset x$		1	0	1	0	Обратная импликация	x, если y; если y, то x; y влечет x
$x \not\leftarrow y$	$x \not\subset y$ $\neg(y \rightarrow x)$ $\neg(y \supset x)$ $\bar{x} \,\&\, y$		0	1	0	0	Отрицание обратной импликации, обратная антиимпликация	не x, но y; неверно, что y влечет x

Логические операции в ЭВМ реализуются соответствующим набором логических элементов.

Логическими элементами ЭВМ называются технические устройства, осуществляющие элементарные логические операции над входными сигналами. Простейшими логическими элементами служат элементы, выполняющие логические операции

И, ИЛИ, НЕ, называемые соответственно *конъюнктор, дизъюнктор* и *инвертор*. Логический элемент имеет один выход, а количество его входов равно числу аргументов логической функции. На основе логических элементов строят более сложные логические схемы. **Логическая схема** — это совокупность логических элементов, реализующих какую-либо булеву функцию.

Конъюнктор (от лат. conjunctio — связь) — это логический элемент, реализующий логическую функцию конъюнкции И (рис. 2.6), или логическое произведение. Конъюнктор выдает сигнал на выходе при наличии сигнала хотя бы на одном из входов.

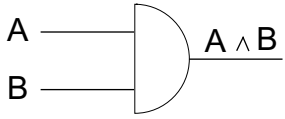


Рис. 2.6. Элемент И (конъюнктор)

Дизъюнктор (от лат. disjunctio — разобщение, обособление) — это логический элемент, реализующий логическую функцию ИЛИ (рис. 2.7), или логическую сумму. Дизъюнктор выдает сигнал на выходе при одновременной подаче сигналов на все его выходы и не выдает выходной сигнал, если не возбужден хотя бы один из его входов.

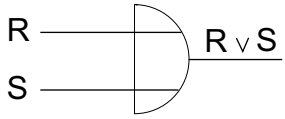


Рис. 2.7. Элемент ИЛИ (дизъюнктор)

Инвертор (от лат. inverto — поворачиваю, переворачиваю) — это логический элемент, реализующий логическую функцию НЕ (отрицание) (рис. 2.8). Различают потенциальные и импульсные инверторы. В потенциальном инверторе высокий уровень входного напряжения преобразуется в низкий и наоборот. В импульсном инверторе в момент подачи сигнала на вход появляется сигнал противоположной полярности на его выходе.

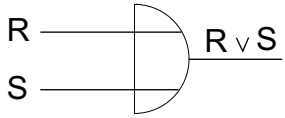


Рис. 2.8. Элемент НЕ (инвертор)

Таким образом, логический элемент — это электронная схема, реализующая элементарную логическую функцию. На основе логических элементов строят более сложные электронные схемы: триггеры, регистры, переключатели, счетчики, сумматоры, шифраторы, дешифраторы и др.

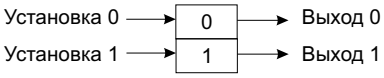


Рис. 2.9. Триггер

Триггер (от англ. trigger — защелка, спусковой механизм) — логическая схема, обеспечивающая два устойчивых состояния (рис. 2.9). Переход триггера из одного устойчивого состояния в другое осуществляется скачком и только под воздействием пускового импульса, подаваемого на один из входов триггера. Так же скачком изменяется уровень напряжения на выходе электронной схемы. Каждый триггер используется для запоминания одного разряда двоичного числа (0 или 1). Триггер можно построить как самостоятельную электронную схему, а можно реализовать на основе элементарных логических элементов.

Регистр (от лат. registrum — список, перечень) — совокупность элементов, предназначенная для приема, временного (оперативного) хранения и выдачи информации в виде мощного слова (здесь термин «слово» используется только для обозначения ячейки памяти ЭВМ).

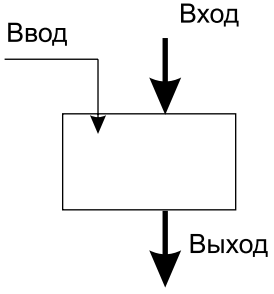


Рис. 2.10. Параллельный регистр

Чаще всего информация в регистры вводится параллельно, т. е. все биты записываются одновременно. *Параллельный регистр* изображен на рис. 2.10 (жирной стрелкой показано наличие нескольких информационных входов, служащих для ввода информации в регистр).

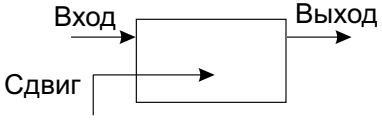


Рис. 2.11. Последовательный регистр

Последовательный регистр может принимать и передавать в каждый момент времени только 1 бит информации. Графическое изображение последовательного регистра приведено на рис. 2.11.

Промежуточным вариантом является *последовательно-параллельный регистр*, в котором запись и считывание информации происходят в один и тот же интервал времени. Так как

символ состоит из нескольких бит, то говорят, что регистр является последовательным по символу и параллельным по битам. Графическое изображение такого регистра представлено на рис. 2.12.

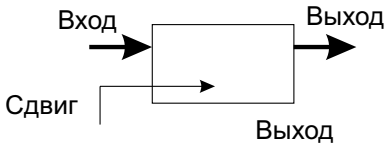


Рис. 2.12. Последовательно-параллельный регистр

Переключатель — это логическая схема, которая разрешает или запрещает прохождение сигнала через линию или группу линий. При этом переключателем можно управлять дистанционно; если переключатель является автоматическим устройством на основе механического или электромеханического прибора, то он называется **реле**. Первые вычислительные машины строились в основном на реле. Хотя реле недорогое устройство и обладает свойством двухпозиционности, его скорость слишком мала для построения на его основе быстродействующих вычислительных машин.

В *электронном переключателе* один электронный сигнал разрешает или запрещает прохождение по сигнальной линии другого электронного сигнала. Оба эти сигнала по своей природе являются электронными, и трудно провести различие между ними. Соединяя два элемента И так, как это показано на рис. 2.13, получим двухканальный переключатель. Заметим, что выходной сигнал наблюдается только при наличии сигналов *A* и *B* или пары сигналов *C* и *B*.

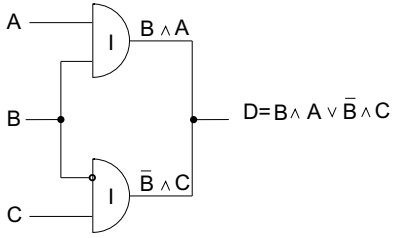


Рис. 2.13. Двухканальный переключатель

Добавленные в схему на рис. 2.13 элементы ИЛИ преобразуют ее в схему, в которой не происходит взаимодействия элементов И (рис. 2.14).

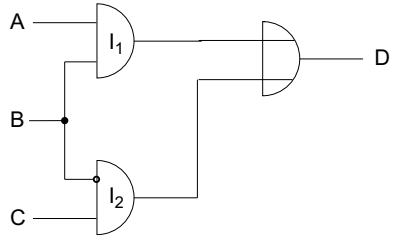


Рис. 2.14. Усовершенствованный двухканальный переключатель

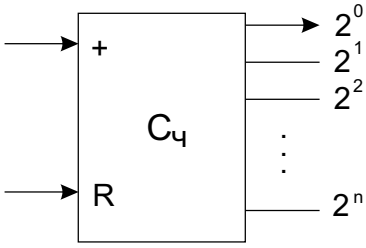


Рис. 2.15. Счетчик

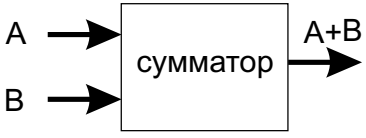


Рис. 2.16. Сумматор

Счетчик — это логическая схема, осуществляющая счет сигналов (импульсов). Счетным элементом двоичного счетчика служит триггер со счетным входом (рис. 2.15). Двоичный счетчик позволяет подсчитывать число сигналов по модулю 2. Максимальное количество состояний счетчика $N = 2^n$ однозначно определяет его емкость $S = N - 1 = 2^n - 1$ и зависит от числа разрядов n. Счетчик имеет предельную емкость, в случае превышения которой показания сбрасываются.

Сумматор — это логическая схема, реализующая операции сложения двух чисел и хранения результатов (рис. 2.16). Сумматор строится на основе цепочки триггеров и является сердцем арифметического блока.

Шифратор — это логическая схема, которая является преобразователем. Шифратор вырабатывает группу бит, соответствующую символу в коде пользователя. Для каждого преобразуемого символа имеется отдельная входная линия. Выходным сигналом является специфический код, соответствующий данному символу на входе (рис. 2.17). Этот код представляется совокупностью сигналов на выходных линиях шифратора. При подаче на вход нескольких символов одновременно эта совокупность выходных сигналов будет ложной.

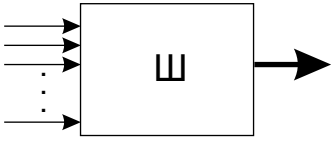


Рис. 2.17. Шифратор

Рис. 2.18. Дешифратор

Дешифратор выполняет функцию, обратную функции шифратора. В дешифраторе код, т. е. группа бит, преобразуется в соответствующий знак (рис. 2.18). На входные кодовые линии могут

поступать различные комбинации сигналов, однако выходной сигнал образуется только на одной выходной линии. Если некоторый набор бит оказался ложным и ему не отвечает ни один символ из выбранного набора символов, то это может быть указано с помощью отдельной выходной линии.

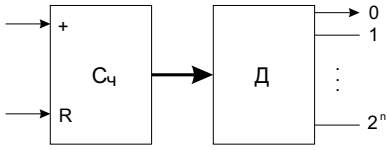


Рис. 2.19. Сочетание счетчика с дешифратором

Часто счетчик используется совместно с дешифратором, как это показано на рис. 2.19, так что результаты счета появляются в виде выходного сигнала на одном из выходов дешифратора.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ТЕМАМ 2.5—2.6

1. Что такое алгебра логики?

2. Какой вклад в развитие алгебры логики внесли Г. Лейбниц, Дж. Буль, Ч. Пирс?

3. Какие логические операции вы знаете?

4. Раскройте понятия «конъюнкция», «дизъюнкция», «отрицание», «эквивалентность», «импликация».

5. Что такое таблица истинности?

6. Как вы понимаете законы алгебры логики?

7. Приведите примеры построения логических схем из логических элементов.

8. Расскажите о причинах использования в ЭВМ двоичной системы счисления, а не, например, десятичной.

9. Что называют логическим элементом?

10. Назовите известные вам логические элементы.

11. Раскройте понятия «конъюнктор», «дизъюнктор», «инвертор», «триггер», «регистр», «переключатель», «счетчик», «сумматор», «шифратор», «дешифратор».

Задания для самостоятельной работы

1. Преобразуйте десятичные числа в восьмеричные и шестнадцатеричные: 35; 1024; 1135.

2. Переведите в восьмеричную и шестнадцатеричную системы счисления следующие двоичные числа:

а) 11110101000100000100111100101000;

б) 10001010101011001100110000000111.

3. Используя двоичную систему счисления, произведите сложение двух чисел:

а) 75 + 44; б) 158 + 36; в) 144 + 56.

Проверьте результат вычислений путем перевода его в десятичную систему.

4. Используя двоичную систему счисления, произведите вычитание путем сложения дополнений до двух:

а) 75 + 44; б) –15 – 36; в) 14 – 56.

Проверьте результат вычислений путем перевода его в десятичную систему счисления.

5. Используя двоичную систему счисления, произведите деление:

а) 75 : 5; б) 54 : 6; в) 56 : 14.

Проверьте результат вычислений путем перевода его в десятичную систему счисления.

6. Рассчитайте объем памяти, необходимый для хранения следующих чисел:

а) 35_{10}; б) 1024_{10}; в) 1135_{8}; г) $10AF_{16}$.

7. Рассчитайте объем памяти, необходимый для хранения следующих чисел с одинарной и двойной точностью:

а) 12,123456789; б) 1456123,23.

8. Подсчитайте количество информации, содержащееся в записи полного адреса вашего учебного заведения при использовании различных кодировок.

9. Вычислите объем памяти, который займет при двоичном кодировании цветная картинка:

а) размером 2 × 4 см, при использовании 256 цветовых оттенков;

б) размером 5 × 6 см, при использовании 15 000 цветовых оттенков.

Учтите, что в каждом квадратном сантиметре содержится 24 × 24 точки.

10. Какой объем адресуемой оперативной памяти имеют ОЗУ с шестнадцатибитовой адресной организацией?

КОНТРОЛЬНЫЕ ВОПРОСЫ К РАЗДЕЛУ 2

1. Назовите основные способы измерения информации.

2. В чем состоит суть энтропийного подхода к измерению информации?

3. Что такое бит и байт?

4. Что такое машинное слово?

5. Что такое элемент и ячейка памяти?

6. Что называется основанием системы счисления?

7. Какие числа называются базисными?

8. Какой смысл вкладывается в понятие записи и представления числа в *K*-ичной системе счисления?

9. Что такое позиционная система счисления?

10. В чем состоит отличие позиционной системы счисления от непозиционной? Приведите примеры.

11. Вспомните общее правило перевода чисел из любой системы счисления в десятичную систему.

12. Назовите правила перевода чисел из десятичной системы счисления в любую другую систему.

13. Сформулируйте основные свойства двоичной системы счисления в ЭВМ.

14. Какие операции с двоичными числами может выполнять процессор вычислительного устройства?

15. Какие существуют формы представления отрицательного числа в двоичной системе счисления?

16. Как представляются целые и действительные числа в ЭВМ? Приведите примеры.

17. Почему при выполнении арифметических операций на ЭВМ возникают ситуации, приводящие к неверному результату?

18. Что такое дополнение до единицы и дополнение до двух?

19. Зачем нужны индикаторы переноса и переполнения?

20. Сформулируйте правило, определяющее выполнение операции сложения в разных числовых системах ЭВМ (со знаком и без знака).

21. Как представляются целые и вещественные числа в памяти ЭВМ?

22. Какой способ представления порядка числа с плавающей точкой называется смещенным?

23. Как представляются символьные данные в памяти ЭВМ?

24. Что такое пиксель?

25. Какие данные хранятся в файлах, содержащих растровые изображения?

Раздел 3

КОМПЬЮТЕР

В результате изучения материала третьего раздела студент должен:

знать:

— общую функциональную схему компьютера;

— назначение и основные характеристики устройств компьютера;

— назначение и основные функции операционной системы;

уметь:

— работать с файлами (создавать, копировать, переименовывать, осуществлять поиск файлов);

— вводить и выводить данные;

— работать с носителями информации;

— пользоваться антивирусными программами;

— соблюдать правила техники безопасности при работе на ЭВМ;

— перечислять состав и назначение программного обеспечения компьютера;

иметь представление:

— об истории развития вычислительной техники;

— о поколениях ЭВМ;

— о перспективах развития персональных компьютеров.

Методические указания

Материал данного раздела составляет основу применения уже имеющихся знаний к решению практических задач на ЭВМ. Студенты должны усвоить, что ЭВМ, наряду с программными средствами, являются инструментом информатики.

Тема 3.1

ОСНОВНЫЕ УСТРОЙСТВА КОМПЬЮТЕРА

3.1.1. Общая схема функционирования компьютера

Создание и функционирование информационных систем в управлении экономикой тесно связано с развитием информационной технологии — главной составной части автоматизированной информационной системы.

Автоматизированная информационная технология (АИТ) — это системно организованная для решения задач управления совокупность методов и средств реализации операций сбора, регистрации, передачи, накопления, поиска, обработки и защиты информации на базе применения развитого программного обеспечения, используемых средств вычислительной техники и связи, а также способов, с помощью которых информация предлагается пользователю.

Все возрастающий спрос на информационные услуги привел к тому, что современная технология обработки информации ориентирована на применение самого широкого спектра технических средств и прежде всего электронных вычислительных машин и средств коммуникаций. На их основе создаются вычислительные системы и сети различных конфигураций с целью не только накопления, хранения, переработки информации, но и максимального приближения терминальных устройств к рабочему месту специалиста или принимающего решения руководителя.

Появление ЭВМ и стремительное совершенствование их эксплуатационных возможностей создало реальные предпосылки для автоматизации управленческого труда, формирования рынка информационных продуктов и услуг. Развитие АИТ шло параллельно с появлением новых видов технических средств обработки и передачи информации, совершенствованием организационных форм использования ЭВМ и ПЭВМ, насыщением инфраструктуры экономики новым средством коммуникации.

ЭВМ появились, когда возникла острая необходимость в проведении трудоемких и точных расчетов. Уровень прогресса в таких областях науки и техники, как, например, атомная энерге-

тика, аэрокосмические исследования, во многом зависел от возможности выполнения сложных расчетов, которые нельзя было осуществить на электромеханических счетных машинах. Требовался переход к вычислительным машинам, работающим с большой производительностью.

Принципиальная схема работы компьютера показана на рис. 3.1. Упрощенная схема вычислительного процесса может быть описана следующим образом. По указанию устройства управления (УУ) управляющая информация (команда) считывается из запоминающего устройства, передается в УУ и расшифровывается. Команда определяет, какая операция и над

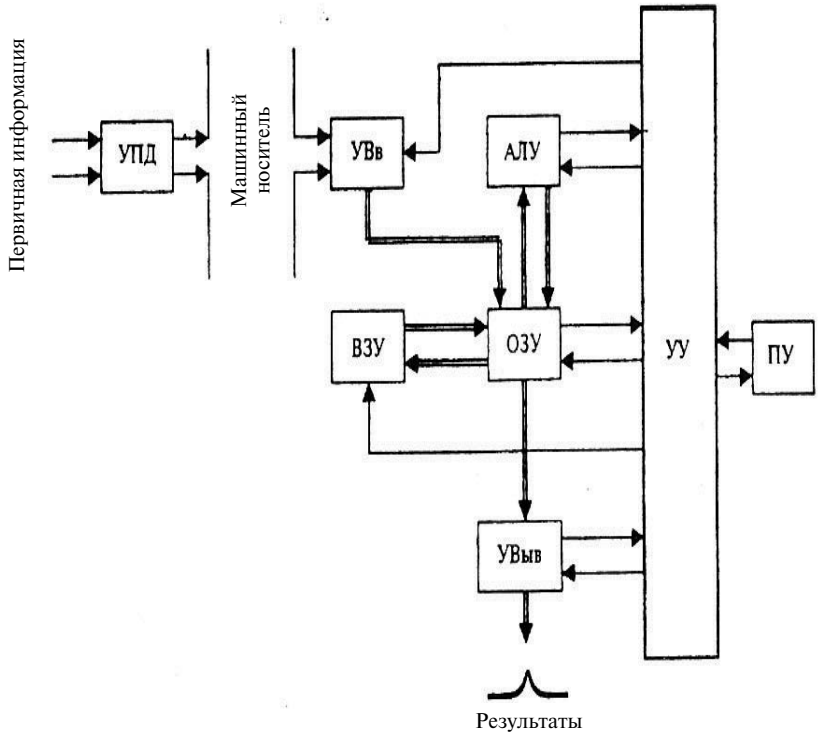


Рис. 3.1. Обобщенная структурная схема ЭВМ:
УПД — устройство подготовки данных; УВв — устройство ввода информации; ОЗУ — оперативное запоминающее устройство; ВЗУ — внешнее запоминающее устройство; АЛУ — арифметико-логическое устройство; УУ — устройство управления; ПУ — пульт управления; УВыв — устройство вывода информации

какими данными должна выполняться в арифметико-логическом устройстве (АЛУ). Получив соответствующие указания и адреса, запоминающее устройство (ЗУ) выдает требуемые числа в АЛУ, где они преобразуются. Результаты обработки пересылаются в оперативное запоминающее устройство (ОЗУ) на хранение. Окончательная результатная информация из ОЗУ с помощью устройств вывода поступает на дисплей, печатающее устройство или на машинный носитель.

3.1.2. Основные блоки и устройства компьютера

Чтобы понять назначение и взаимосвязь основных устройств ЭВМ, необходимо рассмотреть последовательность прохождения информации при ее обработке.

В основе организации вычислительного процесса на ЭВМ лежит *принцип программного управления*. Для решения задачи необходимо составить программу. **Программа** — это определенная последовательность команд (инструкций), которая обеспечивает выполнение задачи. Пользователь записывает программы на каком-либо алгоритмическом языке. Однако компьютер работает под управлением программы, переведенной с алгоритмического на машинный язык, который является собственным языком программирования. *Машинным языком* принято считать способ представления программ и исходных данных в доступном для ЭВМ виде, т. е. это формальный язык, предназначенный для описания алгоритма решения задачи, содержание и правила которого реализуются аппаратными средствами конкретной машины. Машинный язык строится на базе системы команд, характерной для данной ЭВМ.

Программа кодируется в цифровой форме, как и любая информация. Записанная на бланках программа, а также исходная числовая и алфавитная информация переносятся на промежуточный машинный носитель (например, на магнитную ленту) либо вводятся вручную с клавиатуры. Машинный носитель может быть получен на *устройстве подготовки данных* (УПД), которое не связано непосредственно с ЭВМ.

Основной функцией УПД является преобразование информации (текстовой, числовой, графической и пр.) в данные,

записанные в символах входного алфавита ЭВМ. Наиболее распространены устройства подготовки данных на перфоносителях (перфокартах и перфолентах) и магнитных носителях (магнитной ленте, гибких магнитных дисках).

Информация, зафиксированная на машинных носителях, вводится в ЭВМ через устройство ввода, с помощью которого закодированная на носителе информация преобразуется в электрические сигналы и передается в запоминающее устройство. Назначение *устройства ввода информации (УВв)* — преобразование информации, представленной в символах входного алфавита, в символы внутреннего алфавита, в котором осуществляются обработка и хранение информации в ЭВМ. В качестве устройства ввода данных в ЭВМ используются клавиатура, считыватели информации с бумажных перфокарт и перфолент, магнитных дисков и лент. В компьютерах информация часто вводится с *клавиатуры*. Известны устройства ввода информации непосредственно с документа, а также с устройства *ввода речевой информации*.

Устройства хранения информации в ЭВМ. Основной функцией запоминающих устройств (памяти ЭВМ) является хранение программ и данных, необходимых для решения задачи, а также промежуточных и окончательных результатов вычисления. Пользователя ЭВМ интересуют прежде всего такие характеристики, как емкость памяти и ее быстродействие. *Емкость памяти* определяется максимально возможным количеством кодов чисел и команд определенной разрядности, которые могут одновременно храниться в ЗУ. *Быстродействие памяти* характеризуется временем, которое необходимо для поиска (выборки), записи или считывания информации.

На практике емкость запоминающих устройств ЭВМ обычно измеряется в килобайтах и мегабайтах.

Время обращения — интервал времени между началом и окончанием ввода (вывода) информации в память (из памяти). В отношении устройств памяти вместо термина «ввод-вывод» используют термин «запись-чтение». Таким образом, время обращения характеризует затраты времени на поиск места в памяти и запись (чтение) информации.

Различают оперативные и внешние запоминающие устройства (ЗУ).

Оперативные запоминающие устройства (ОЗУ) характеризуются малым значением времени обращения, которое обычно составляет 10^{-7}—10^{-5} с (100 нс — 10 мкс), при емкости от сотен мегабайт до десятков гигабайт.

Внешние запоминающие устройства (ВЗУ) характеризуются в сравнении с ОЗУ бóльшим временем обращения (10^{-2} с и более), но емкость их может быть практически неограниченной, так как число блоков внешней памяти можно наращивать по мере необходимости.

В современных ЭВМ устройства оперативной памяти реализуются на быстродействующих электронных элементах. В качестве ВЗУ обычно используются запоминающие устройства на магнитных дисках и лентах.

С понятием «поиск места в памяти» связано понятие «адрес информации». Память ЭВМ можно представить состоящей из отдельных ячеек, в каждой из которых хранится определенное количество информации: один байт, два байта, четыре байта и т. д. Все ячейки в памяти пронумерованы. Номера ячеек называются **адресами**.

В современных ЭВМ минимальные адресуемые элементы памяти обычно имеют емкость один байт. Если информация занимает в памяти несколько последовательных элементов, то она составляет *поле памяти*. Местонахождение информации в этом случае задается адресом первого байта и количеством занимаемых ею байт памяти.

Различают запоминающие устройства произвольного и последовательного доступа. *Устройства произвольного доступа* позволяют по заданному адресу одинаково быстро извлекать информацию из любого элемента памяти. Такими устройствами являются используемые в ЭВМ оперативные ЗУ. Примером запоминающих *устройств последовательного доступа* являются ЗУ на магнитных лентах (МЛ). Данные на МЛ записываются байт за байтом по всей длине. Чтобы считать информацию, необходимо перематывать ленту до тех пор, пока участок с записью требуемой информации не будет подведен к считывающему устройству. Время обращения при этом зависит от того, как «далеко» на ленте находится требуемая информация.

Арифметико-логическое устройство (АЛУ) выполняет арифметические и логические операции, предусмотренные системой команд ЭВМ, и характеризуется:

— набором выполняемых элементарных операций (сложение, вычитание, умножение, деление, сравнение, логические операции);

— временем выполнения элементарных операций;

— средним быстродействием, т. е. количеством арифметических и логических операций, выполняемых в единицу времени (секунду). Быстродействие АЛУ современных ЭВМ изменяется в довольно широких пределах — от сотен тысяч до нескольких миллиардов операций в секунду.

В состав АЛУ входят регистры *РгА, ГгВ* и *РгС. Регистры,* как и ячейки ЗУ, являются элементами памяти. Перед выполнением операции исходные данные *А* и *В* считываются из ОЗУ ЭВМ и помещаются на временное хранение в регистры *РгА* и *РгВ*. Из устройства управления ЭВМ в блок управления *АЛУ* поступает команда на выполнение заданной операции. Операционная часть *АЛУ,* получив необходимые команды от блока управления, выполняет заданную операцию над исходными данными *А* и *В* и результат засылает на временное хранение в регистр результата *РгС*. Затем, если это требуется, результат из *РгС* помещается для хранения в ячейку памяти ОЗУ.

Быстродействие ЭВМ связано со временем обращения ОЗУ. Так, в ЭВМ с быстродействием 200 тыс. оп/с одна операция выполняется за 5 мкс, в течение которых необходимо не только выполнить операцию, но и трижды обратиться к оперативной памяти: считать два операнда и записать результат. Для обеспечения заданного быстродействия время обращения для ОЗУ данной ЭВМ должно иметь порядок 1 мкс (10^{-6} с).

Устройство управления (УУ) предназначено для автоматического управления ходом вычислительного процесса и обеспечивает необходимое взаимодействие всех устройств машины по автоматическому выполнению заданного алгоритма решения задачи. Функции УУ, как наиболее важные для понимания автоматической работы ЭВМ, описываются далее при рассмотрении реализации в ЭВМ принципа программного управления.

Устройства вывода предназначены для вывода информации из машины и преобразования результатов решения задач из символов внутреннего алфавита в выходную информацию, представленную в символах выходного алфавита.

В качестве устройств вывода ЭВМ применяют печатающие устройства, графопостроители, устройства вывода информации на перфокарты и перфоленты, алфавитно-цифровые и графические дисплеи, различные звукогенерирующие устройства, позволяющие выводить из ЭВМ звуковую информацию — от простейших звуковых сигналов типа «щелчок» до более сложных: музыкальных мелодий, искусственно синтезированной речи и т. д. В качестве устройства вывода ЭВМ можно рассматривать и накопители информации на магнитных лентах и гибких магнитных дисках.

Печатающие, графические и отображающие устройства, как правило, выдают информацию в виде, удобном для человека. Устройства, фиксирующие информацию на магнитных лентах и перфоносителях, служат для выдачи данных, которые в последующем предполагается обрабатывать на ЭВМ или использовать для управления какими-либо автоматами (например, задания для станков с числовым программным управлением). Специализированные ЭВМ формируют на выходе электрические сигналы, управляющие работой различных исполнительных устройств (механизмов), и выдают другую информацию, форма представления которой определяется характером управляемой системы. Рассмотренную структурную систему ЭВМ (см. рис. 3.1) можно считать классической. Она получила название **компьютера фон Неймана** — по имени американского математика, предложившего такую структуру машины для автоматической обработки данных. В современных ЭВМ комплекс устройств, охватывающий АЛУ и УУ, принято называть **процессором**.

Скорость работы внешних устройств ЭВМ, включающих устройства ввода и вывода информации, значительно ниже скорости работы процессора. Поэтому для более эффективного использования возможностей процессора к нему через специальные устройства, называемые **каналами**, подключают несколько одновременно обслуживаемых внешних устройств.

Каналы, через которые осуществляется ввод-вывод информации, представляют собой специализированные процессоры для выполнения операций ввода-вывода. Структуры связи устройств таких ЭВМ можно лишь условно назвать электронными вычислительными машинами. По существу, это уже не маши-

ны, а сложные вычислительные системы, которые могут включать в свой состав несколько обрабатывающих процессоров, обеспечивать возможность обмена информацией с другими ЭВМ, выполнять передаваемые по каналам связи задания удаленных пользователей и другие сложные функции по обработке данных.

3.1.3. Архитектура ЭВМ

Обработка информации и представление ее результатов в удобном для человека виде производится с помощью вычислительных средств. Научно-технический прогресс привел к созданию разнообразных вычислительных средств: электронных вычислительных машин (ЭВМ), вычислительных систем (ВС), вычислительных сетей (ВСт). Они различаются структурной организацией и функциональными возможностями.

Дать определение такому явлению, как ЭВМ, представляется сложным. Достаточно сказать, что само по себе название «ЭВМ», т. е. «электронные вычислительные машины», не отражает полностью сущность концепции. Слово «электронные» подразумевало электронные лампы в качестве элементной базы, современные ЭВМ правильнее следовало бы называть микроэлектронными. Слово «вычислительный» подразумевает, что устройство предназначено для проведения вычислений, однако анализ программ показывает, что современные ЭВМ не более 10—15 % времени тратят на чисто вычислительную работу — сложение, вычитание, умножение и т. д. Основное время затрачивается на выполнение операций пересылки данных, сравнения, ввода-вывода и т. д. То же самое относится и к англоязычному термину «компьютер», т. е. вычислитель. К понятию «ЭВМ» можно подходить с нескольких точек зрения.

Представляется разумным определить ЭВМ с точки зрения ее функционирования. Целесообразно описать минимальный набор устройств, который входит в состав любой ЭВМ, и тем самым определить состав минимальной ЭВМ, а также сформулировать принципы работы отдельных блоков ЭВМ и принципы организации ЭВМ как системы, состоящей из взаимосвязанных функциональных блоков.

Если же рассматривать ЭВМ как ядро некоей информационно-вычислительной системы, может оказаться полезным показать информационную модель ЭВМ — определить ее в виде совокупности блоков переработки информации и множества информационных потоков между этими блоками.

Принципы фон Неймана. Большинство современных ЭВМ строится на базе принципов, сформулированных американским ученым, одним из «отцов» кибернетики Джоном фон Нейманом (1903—1957). Впервые эти принципы были опубликованы фон Нейманом в 1945 г. в его предложениях по машине EDVAC. Эта ЭВМ была одной из первых машин с хранимой программой, т. е. с программой, находящейся в памяти машины, а не считываемой с перфокарты или другого подобного устройства. В целом эти принципы сводятся к следующему:

1) основными блоками фоннеймановской машины являются блок управления, арифметико-логическое устройство, память и устройство ввода-вывода (см. рис. 3.1);

2) информация кодируется в двоичной форме и разделяется на единицы, называемые **словами**;

3) алгоритм представляется в форме последовательности управляющих слов, которые определяют смысл операции. Эти управляющие слова называются **командами**. Совокупность команд, представляющая алгоритм, называется **программой**;

4) программы и данные хранятся в одной и той же памяти. Разнотипные слова различаются по способу использования, но не по способу кодирования;

5) устройство управления и АЛУ обычно объединяются в одно устройство, называемое **центральным процессором** (**ЦП**). ЦП определяет действия, подлежащие выполнению, путем считывания команд из оперативной памяти. Обработка информации, предписанная алгоритмом, сводится к последовательному выполнению команд в порядке, однозначно определяемом программой.

Принципы фон Неймана практически можно реализовать множеством различных способов. В этой теме приведем один из них: ЭВМ с шинной, или магистрально-модульной, организацией. Введем несколько определений.

Архитектура ЭВМ — абстрактное определение машины в терминах основных функциональных модулей, языка, структур

данных. Архитектура не определяет особенности реализации аппаратной части ЭВМ, время выполнения команд, степень параллелизма, ширину шины и другие аналогичные характеристики. Архитектура отображает структуру ЭВМ, которая является видимой для пользователя: систему команд, режимы адресации, форматы данных, набор программно-доступных регистров. Одним словом, термин «архитектура» используется для описания возможностей, представляемых ЭВМ. Весьма часто употребляется термин «конфигурация» ЭВМ, под которым понимается компоновка вычислительного устройства с четким определением характера, количества взаимосвязей и основных характеристик его функциональных элементов. Термин «организация» ЭВМ определяет, как реализованы возможности ЭВМ.

Команда — совокупность сведений, необходимых процессору для осуществления определенного действия при выполнении программы. Команда состоит из *кода операции,* содержащего указание на операцию, которую необходимо выполнить, и нескольких *адресных полей,* содержащих указание на места расположения операндов команды. Способ вычисления адреса по информации, содержащейся в адресном поле команды, называется **режимом адресации**. Множество команд, реализованных в данной ЭВМ, образует ее *систему команд.*

3.1.4. Магистрально-модульный принцип построения компьютера

Магистрально-модульная (шинная) организация является простейшей формой организации ЭВМ. В соответствии с приведенными выше принципами фон Неймана подобная ЭВМ имеет в своем составе следующие функциональные блоки (рис. 3.2).

Магистраль (наиболее часто употребляется термин «шина») — это совокупность электрических линий для обмена данными между устройствами компьютера. Тип шины определяет и сигналы, которые передаются по этим линиям. В персональном компьютере типы шин определяются материнской платой. Основные характеристики шин — разрядность передаваемых данных и скорость передачи данных в Мбайт/с. Наибольший

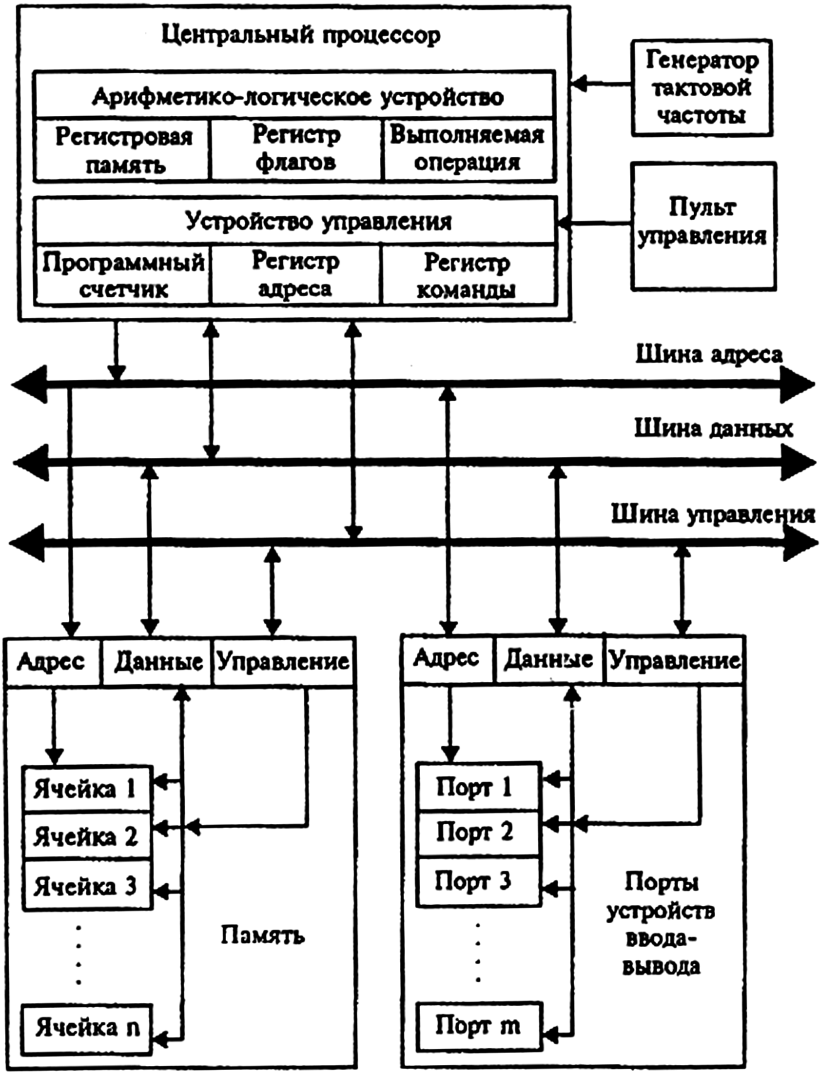


Рис. 3.2. Упрощенная схема ЭВМ с шинной организацией

интерес вызывают два типа шин: системный и локальный. *Системная шина* предназначена для обеспечения передачи данных между периферийными устройствами и ЦП, а также оперативной памятью (ОП). *Локальная шина* — это шина, непосредственно подключенная к контактам микропроцессора, т. е. шина процессора.

Модуль — функциональная часть технического обеспечения вычислительной системы, выполненная в одном блоке и имеющая узлы сопряжения с другими модулями. Модули применяются для удобной эксплуатации ВС, их можно быстро заменить при выходе из строя и при модернизации.

Центральный процессор (ЦП) — функциональная часть ЭВМ, выполняющая основные операции по обработке данных и управлению работой других блоков. Это наиболее сложный компонент ЭВМ как с точки зрения электроники, так и с точки зрения функциональных возможностей. Центральный процессор состоит из следующих взаимосвязанных составных элементов: арифметико-логического устройства, устройства управления и регистров.

Арифметико-логическое устройство выполняет основную работу по переработке информации, хранимой в оперативной памяти. В нем выполняются арифметические и логические операции. Кроме того, АЛУ вырабатывает управляющие сигналы, позволяющие ЭВМ автоматически выбирать путь вычислительного процесса в зависимости от получаемых результатов. Операции выполняются с помощью электронных схем, каждая из которых состоит из нескольких тысяч элементов. Микросхемы имеют высокую плотность и быстродействие. На современном технологическом уровне все АЛУ можно разместить на одном кристалле полупроводникового элемента размером с конторскую скрепку. АЛУ формирует по двум входным переменным одну — выходную, выполняя заданную функцию (сложение, вычитание, сдвиг и т. д.). Выполняемая функция определяется микрокомандой, получаемой от устройства управления. АЛУ содержит в своем составе устройство, хранящее характеристику результата выполнения операции над данными и называемое **флаговым регистром**. Отметим пока, что отдельные разряды этого регистра указывают на равенство результата операции нулю, знак результата операции («+» или «–»), правильность

выполнения операции (наличие переноса за пределы разрядной сетки или переполнения). Программный анализ флагов позволяет производить операции ветвления программы в зависимости от конкретных значений данных.

Кроме того, в АЛУ имеется набор программно-доступных быстродействующих ячеек памяти, которые называются **регистрами**. Регистры составляют основу архитектуры процессора.

Среди обязательного набора регистров можно отметить следующие. *Регистр данных* — служит для временного хранения промежуточных результатов при выполнении операций. *Регистр-аккумулятор* — регистр временного хранения, который используется в процессе вычислений (например, в нем формируется результат выполнения команды умножения). *Регистр-указатель стека* — используется при операциях со стеком, т. е. с такой структурой данных, которая работает по принципу «последним вошел — первым вышел», т. е. последнее записанное в него значение извлекается из него первым. Пока отметим только, что стеки применяются для организации подпрограмм. *Индексные, указательные* и *базовые регистры* используются для хранения и вычисления адресов операндов в памяти, *регистры-счетчики* — для организации циклических участков в программах. *Регистры общего назначения,* имеющиеся во многих ЭВМ, могут использоваться для любых целей; точное назначение такого регистра определяет программист при написании программы. Они могут служить для временного хранения данных, в качестве аккумуляторов, а также индексных, базовых и указательных реестров. Количество регистров и связей между ними оказывает существенное влияние на сложность и стоимость процессора. С другой стороны, наличие большого количества регистров с богатым набором возможностей упрощает программирование и повышает гибкость программного обеспечения. Кроме перечисленных регистров, в состав АЛУ могут входить *внутренние системные регистры,* недоступные программно и используемые во время внутренних пересылок информации при выполнении команд.

Устройство управления (УУ) — часть центрального процессора. Оно вырабатывает распределенную во времени и пространстве последовательность внутренних и внешних управляющих сигналов, обеспечивающих выборку и выполнение

команд. На этапе цикла выборки команды УУ интерпретирует команду, выбранную из программной памяти. На этапе выполнения команды в соответствии с типом реализуемой операции УУ формирует требуемый набор команд низкого уровня для АЛУ и других устройств. Эти команды задают последовательность простейших низкоуровневых операций, таких, как пересылка данных, сдвиг данных, установка и анализ признаков, запоминание результатов и др. Такие элементарные низкоуровневые операции называются **микрооперациями**, а команды, формируемые устройством управления, — **микрокомандами**. Последовательность микрокоманд, соответствующая одной команде, называется **микропрограммой**.

В простейшем случае УУ имеет в своем составе три устройства — *регистр команды,* который содержит код команды во время ее выполнения; *программный счетчик,* в котором содержится адрес очередной подлежащей выполнению команды; *регистр адреса,* в котором вычисляются адреса операндов, находящихся в памяти. Для связи пользователя с ЭВМ предусмотрен *пульт управления,* который позволяет выполнять такие действия, как сброс ЭВМ в начальное состояние, просмотр регистра или ячейки памяти, запись адреса в программный счетчик, пошаговое выполнение программы при ее отладке и т. д.

Память (ПАМ) — устройство, предназначенное для запоминания, хранения и выборки программ и данных. Память состоит из конечного числа ячеек, каждая из которых имеет свой уникальный номер, или адрес. Доступ к ячейке осуществляется указанием ее адреса. Память способна выполнять два вида операций над данными — чтение с сохранением содержимого и запись нового значения со стиранием предыдущего. Как уже говорилось выше, каждая ячейка памяти может использоваться для хранения либо порции данных, либо команды. В большинстве современных ЭВМ минимально адресуемым элементом памяти является *байт* — поле из восьми бит. Совокупность бит, которые АЛУ может одновременно поместить в регистр или обработать, называется обычно **машинным словом**.

Оперативная память (ОП) — функциональный блок, хранящий информацию для УУ (команды) и АЛУ (данные). Задачи, решаемые с помощью ЭВМ, требуют хранения в памяти различного количества информации, зависящего от сложности

реализуемого алгоритма, количества исходных данных и т. п. Поэтому память должна вмещать достаточно большое количество информации, т. е. должна иметь большую емкость. С другой стороны, память должна обладать достаточным быстродействием, соответствующим быстродействию других устройств ЭВМ. Чем больше емкость памяти, тем медленнее доступ к ней, так как время доступа (т. е. быстродействие) определяется временем, необходимым для выборки из памяти или записи в нее информации. Поэтому в ЭВМ существует несколько запоминающих устройств, различающихся емкостью и быстродействием (табл. 3.1).

Таблица 3.1

Виды запоминающих устройств

Устройства памяти	Время доступа, с	Емкость, бит
Регистры	$(2—20) \times 10^{-9}$	$10^3—10^4$
Оперативная память	$(0{,}2—20) \times 10^{-6}$	$10^6—10^8$
Внешняя память	10—100	$10^{11}—0^{12}$

Оперативная память собирается на ферритовых сердечниках или полупроводниковых микросхемах и состоит из отдельных ячеек.

Периферийные устройства (ПУ). В их число входят *устройства внешней памяти*, предназначенные для долговременного хранения данных большого объема и программ, и *коммуникационные устройства*, предназначенные для связи ЭВМ с внешним миром (с пользователем, другими ЭВМ и т. д.). Обмен данными с внешним устройством осуществляется через *порты ввода-вывода*. «Порт» (от англ. port — ворота, дверь, отверстие) — абстрактное понятие. По аналогии с ячейками памяти порты можно рассматривать как ячейки, через которые можно записать информацию в ПУ или, наоборот, прочитать ее из него. Так же как и ячейки памяти, порты имеют уникальные номера — адреса портов ввода-вывода.

Система шин. Объединение функциональных блоков в ЭВМ осуществляется посредством следующей системы шин: *шины данных,* по которой осуществляется обмен информацией между

блоками ЭВМ; *шины адреса*, используемой для передачи адресов (номеров ячеек памяти или портов ввода-вывода, к которым производится обращение), и *шины управления,* которая служит для передачи управляющих сигналов. Совокупность этих трех шин называется **системной шиной**, **системной магистралью** или **системным интерфейсом**. Состав и назначение шин, правила их использования, виды передаваемых по шине сигналов и другие характеристики шины могут существенно различаться у разных видов ЭВМ.

Однако есть принципиально общие закономерности организации шин. Шина состоит из отдельных проводников *(линий)*. Сигналы по линиям шины могут передаваться либо импульсами (наличие импульса соответствует логической единице, а отсутствие импульса — нулю), либо уровнем напряжения (например, высокий уровень — логическая единица, низкий — нуль). **Шириной шины** называется количество линий (проводников), входящих в состав шины. Ширина шины адреса определяет размер адресного пространства ЭВМ. Если, например, количество линий адреса, используемых для адресации памяти, равно 20, то общее количество адресуемых ячеек памяти составит 20, т. е. немногим более 1 млн (точнее — 1 048 576) ячеек. Обычно на шине в любой момент можно выделить два активных устройства. Одно из них называется **задатчиком** и инициирует операцию обмена данными (формирует адреса и управляющие сигналы), другое — **исполнителем** и выполняет операцию (формирует адреса и управляющие сигналы и принимает или передает данные). В большинстве случаев задатчиком является ЦП. Память всегда выступает только в качестве исполнителя.

Функционирование ЭВМ с шинной структурой можно описать следующим обобщенным алгоритмом (рис. 3.3):

1. Инициализация: после включения ЭВМ или операции сброса в регистры центрального процессора заносятся некоторые начальные значения. Обычно в процессе инициализации в память ЭВМ помещается программа, называемая **первичным загрузчиком**. Основное назначение первичного загрузчика — загрузить в память с устройства внешней памяти операционную систему. Эта программа может быть размещена в энергонезависимом устройстве памяти или автоматически считываться

с некоторого устройства внешней памяти. Мы не будем здесь подробно останавливаться на механизмах загрузки операционной системы, тем более что они могут существенно различаться для разных типов ЭВМ. Пока будем полагать, что в памяти некоторым образом оказалась первая из подлежащих выполнению программ. Программному счетчику присваивается начальное значение, равное адресу первой команды программы, указанной выше.

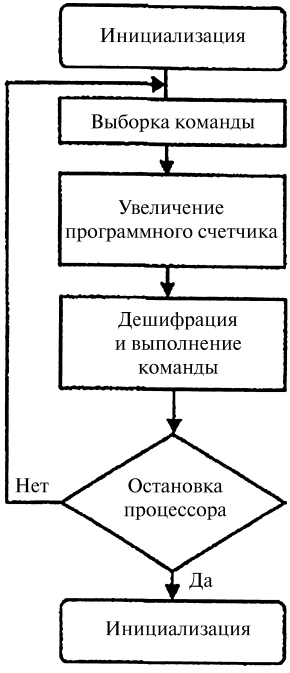


Рис. 3.3. Обобщенный алгоритм фоннеймановской ЭВМ

2. Центральный процессор производит операцию считывания команды из памяти. В качестве адреса ячейки памяти используется содержимое программного счетчика.

3. Содержимое считанной ячейки памяти интерпретируется процессором как команда и помещается в регистр команды. Устройство управления приступает к интерпретации прочитанной команды. По полю команды операции из первого слова команды устройство управления определяет ее длину и, если это необходимо, организует дополнительные операции считывания, пока вся команда полностью не будет прочитана процессором. Вычисленная длина команды прибавляется к исходному содержимому программного счетчика, и когда команда полностью прочитана, программный счетчик будет хранить адрес следующей команды.

4. По адресным полям команды устройство управления определяет, имеет ли команда операнды в памяти. Если это так, то на основе указанных в адресных полях режимов адресации вычисляются адреса операндов и производятся операции чтения памяти для считывания операндов.

5. Устройство управления и арифметико-логическое устройство выполняют операцию, указанную в поле кода операции

команды. Во флаговом регистре процессора запоминаются признаки результата операции (равно нулю или нет, знак результата, наличие переполнения и т. д.).

6. Если это необходимо, устройство управления выполняет операцию записи, для того чтобы поместить результат выполнения команды в память.

7. Если последняя команда не была командой ОСТАНОВИТЬ ПРОЦЕССОР, то описанная последовательность действий повторяется, начиная с шага 1. Описанная последовательность действий центрального процессора с шага 1 до шага 6 называется **циклом процессора**.

Большинство мини- и микроЭВМ имеют шинную организацию, и их поведение описывается приведенным выше алгоритмом. В различных конкретных ЭВМ реализация этого алгоритма может несколько отличаться. Так, например, по-разному может осуществляться синхронизация задатчиков и исполнителей, процессор может считывать из памяти не одну команду, а сразу несколько и хранить их в специальной очереди команд. Команды и данные, часто используемые программой, могут храниться не в основной памяти ЭВМ, а в быстродействующей буферной памяти и т. д. Таким образом, функционирование любой фон-неймановской ЭВМ описывается алгоритмом, близким к рассмотренному выше, и представляет собой последовательность достаточно простых действий.

3.1.5. Персональные ЭВМ

С 1975 г. в США было начато серийное производство персонального компьютера, или ПЭВМ, или ПК. Это событие сегодня в мире часто называют второй информационной революцией (первой информационной революцией считается появление печатного станка и книгопечатания — 1445 г.). ПК появился на базе мини- и микроЭВМ и обеспечивает персональные вычисления, т. е. режим работы специалиста в данной предметной области, непосредственно на рабочем месте. За дисплей ПЭВМ смог сесть пользователь — непрофессионал в программировании. С 1981 г. стали выпускаться ПЭВМ, имеющие блочно-модульную конструкцию. Эти простые в эксплуатации и сравни-

тельно дешевые машины предназначались для потребителей, не обладающих знаниями в области вычислительной техники и программирования. Широкое распространение мини-ЭВМ в начале 1970-х гг. определялось необходимостью приблизить компьютер к пользователю. Мини-ЭВМ устанавливались непосредственно на предприятиях и в организациях, где использование больших ЭВМ было экономически невыгодным.

Таким образом, ПК — это компьютер, предназначенный для индивидуального использования. В настоящее время это мощный универсальный компьютер; он успешно работает как дома, так и на рабочих местах в офисах, легко подключается к различным вычислительным сетям.

Основные критерии отнесения компьютера к классу ПК — малые размеры, отсутствие необходимости в обслуживании, низкая цена, функциональная универсальность и простота модернизации.

Краткая характеристика и классификация ПК. Так как технической основой ПК служит микропроцессор (МП), то развитие технологии МП определило смену *поколений* ЭВМ:

— 8-разрядный МП (1975—1980 гг.) — первое поколение;

— 16-разрядный МП (1981—1985 гг.) — второе поколение;

— 32-разрядный МП (1986—1992 гг.) — третье поколение;

— 64-разрядный МП (1993 г. — по настоящее время) — четвертое поколение.

Важную роль в развитии ПЭВМ сыграло появление компьютера IBM PC, произведенного корпорацией IBM (США) на базе МП Intel-8086 в 1981 г. Этот персональный компьютер занял ведущее место на рынке ПЭВМ. Его основное преимущество — так называемая открытая архитектура, благодаря которой пользователи могут расширять возможности приобретенной ПЭВМ, добавляя различные периферийные устройства и модернизируя компьютер. В дальнейшем другие фирмы начали создавать свои ПЭВМ, но компьютер IBM PC стал как бы стандартом класса ПЭВМ. В наши дни около 85 % всех продаваемых ПЭВМ базируется на архитектуре IBM PC.

По назначению ПЭВМ классифицируют на бытовые, общего назначения и профессиональные.

Бытовые ПЭВМ предназначены для массового потребителя, поэтому они должны быть достаточно дешевыми, надежны-

ми и иметь, как правило, простейшую базовую конфигурацию. Бытовые ПЭВМ используют в домашних условиях для развлечений (видеоигры), для обучения и тренировки, управления бытовой техникой. Вместе с тем архитектура этих машин позволяет подключать их к каналам связи, расширять набор периферийного оборудования. При некоторой модернизации эти модели (например, отечественную ПЭВМ «Амата») можно применять для индивидуальной обработки текста, решения небольших научных и инженерных задач. Бытовые ПЭВМ снабжены пакетом игр, программным обеспечением локальной сети и др. Фирмы предлагают за дополнительную плату нарастить комплектность компьютера накопителями на жестких магнитных дисках (НЖМД) типа «винчестер», музыкальной картой, монитором и т. д. Модель «Амата» легко превращается в ПЭВМ общего назначения.

ПЭВМ *общего назначения* применяются для решения научно-технических и экономических задач, для обучения и тренировки. Машины этого класса получили наибольшее распространение на мировом рынке. Они обладают достаточно большой емкостью оперативной памяти, имеют внешнюю память на гибких и жестких магнитных дисках, собственный дисплей. Интерфейсы позволяют подключать большое количество периферийных устройств, средства для работы в составе вычислительных сетей.

С ПЭВМ общего назначения работают прежде всего пользователи-непрофессионалы. Поэтому такие ПЭВМ снабжены развитым программным обеспечением, включающим операционные системы, трансляторы с алгоритмических языков, пакеты прикладных программ. В состав аппаратуры входят устройства для вывода как текстового, так и графического материала, принтеры с высоким качеством печати.

Профессиональные ПЭВМ применяются в научной сфере, для решения сложных информационных и производственных задач, где требуются высокое быстродействие, эффективная передача больших массивов информации, достаточно большая емкость оперативной памяти. Потребителями профессиональных ПЭВМ, как правило, являются профессионалы-программисты, поэтому программное обеспечение должно быть достаточно богатым, гибким, включать инструментальные программные средства.

Благодаря подключению широкой номенклатуры периферийных устройств, функциональные возможности ПЭВМ значительно расширяются. Они могут работать в многозадачном режиме, с алгоритмическими языками высокого уровня, в составе вычислительных сетей. По своим функциональным возможностям многопроцессорные профессиональные ПЭВМ не только приближаются к большим ЭВМ предыдущего поколения, но и вполне могут конкурировать с ними.

В настоящее время появился новый признак классификации ПЭВМ — по конструктивному исполнению, связанный с микроминиатюризацией изделий. Результатом снижения массы и уменьшения габаритов стали компьютеры LAPTOP («наколенный» компьютер), NOTEBOOK (компьютер-«блокнот») и HANDHELD («ручной» компьютер: от hand — рука и held — держать).

В LAPTOP-компьютере клавиатура и системный блок выполнены в одном корпусе, который сверху, как крышкой, закрывается жидкокристаллическим дисплеем, неразъемносоединенным со своим электронным основанием. Соединительные провода между дисплеем и ЭВМ скрыты в корпусе. Пользователь может легко переносить компьютер и держать его на коленях. Эти модели уступают по своим техническим параметрам профессиональным ПЭВМ. Они построены на базе МП i80386, имеют встроенные накопители на гибких магнитных дисках (НГМД) и НЖМД. Компьютеры класса LAPTOP не должны весить более 3,5 кг.

Модели NOTEBOOK имеют размер листа бумаги стандарта А4 (297 × 210), снабжены неполной клавиатурой (около 80 клавиш), имеют НЖМД (например, дисковод емкостью от 80 Гбайт) и НГМД. В комплекте с NOTEBOOK применяются модемы или факс-модемы, подключаемые к компьютерам и телефонной сети. Компьютеры NOTEBOOK могут использоваться в деловых поездках. Они не требуют места на рабочем столе, их можно хранить их в ящике для бумаг, в портфеле.

Размер моделей HANDHELD — меньше листа бумаги формата А4 (например, размер модели Hewlett Packard 95 LX составляет 160 × 136 × 25 мм). Такой компьютер всегда под рукой (в кармане), в готовом к работе состоянии. Эти модели могут работать независимо от электросети. При автономной работе

программы вводятся с помощью твердой карточки ROM CARD большой емкости. (Аббревиатура ROM образована от Read Only Memory — память только для чтения.) Карточки можно перепрограммировать. Для хранения результатов расчетов, введенного текста, составленных электронных таблиц и других результатов работы пользователь применяет ROM CARD размером 2×5 мм со встроенной батарейкой, что позволяет вставлять их в специальные отверстия в корпусе ПК для чтения с них программ, данных или записи результатов работы. По мере надобности результаты работы можно перенести по кабелю в настольный компьютер. В конструкциях моделей HANDHELD предусмотрен гораздо больший объем постоянной памяти, чем в конструкциях настольных ПЭВМ.

Миниатюрные компьютеры можно включать в вычислительные сети без проводов (с помощью радио), что потребует минимальных затрат. Такая технология получила название полевой компьютеризации (Field Computing).

3.1.6. Характеристика основных блоков персональных компьютеров

ПЭВМ включает три основных устройства: системный блок, клавиатуру и дисплей (монитор). Однако для расширения их функциональных возможностей можно подключить различные дополнительные периферийные устройства, в частности принтеры, накопители на магнитной ленте (стримеры), различные манипуляторы (мышь, джойстик, трекбол, световое перо), устройства оптического считывания изображений (сканеры), графопостроители (плоттеры) и др.

Такие устройства подсоединяют к системному блоку с помощью кабелей через специальные гнезда (разъемы), которые размещены обычно на его задней стенке. В некоторых моделях ПЭВМ при наличии свободных гнезд дополнительные устройства вставляются непосредственно в системный блок (например, модем для обмена информацией с другими ПЭВМ через телефонную связь). ПЭВМ, как правило, имеет модульную структуру (рис. 3.4). Все модули связаны с системной магистралью (шиной).

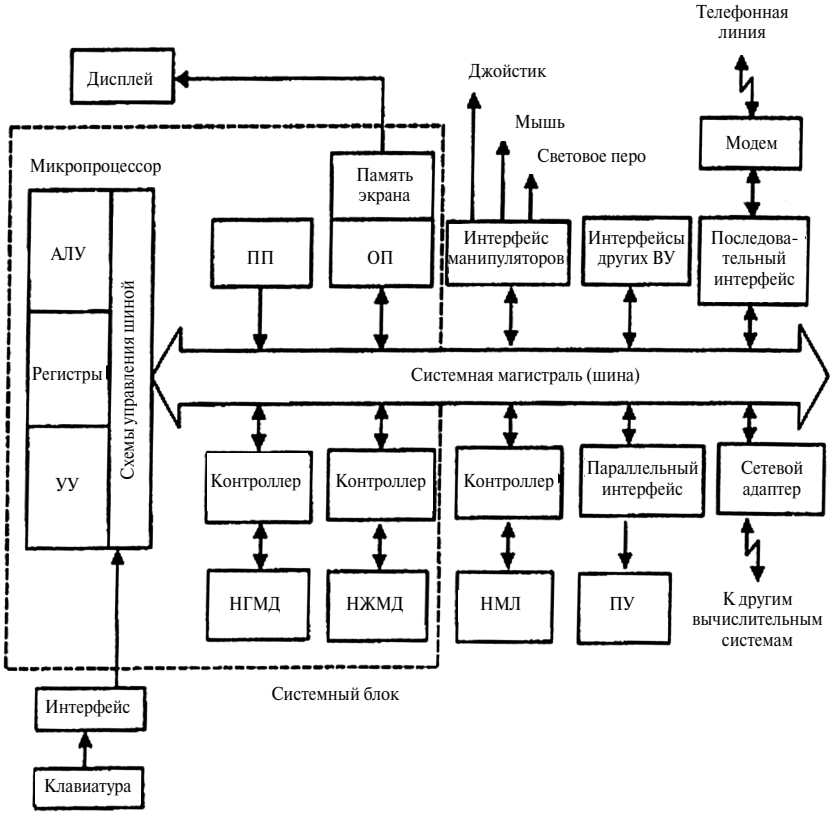


Рис. 3.4. Структурная схема ПЭВМ с периферийными устройствами: ПП — постоянная память; ОП — оперативная память; ВУ — внешнее устройство; НГМД — накопитель на гибких магнитных дисках; НЖМД — накопитель на жестких магнитных дисках; НМЛ — накопитель на магнитной ленте; ПУ — печатающее устройство

Системный блок. Это главный блок ПЭВМ, он включает в свой состав центральный микропроцессор, сопроцессор, модули оперативной и постоянной памяти, контроллеры, накопители на магнитных и оптических дисках и другие функциональные модули. Набор модулей определяется типом ПЭВМ. Пользователи по своему желанию могут изменять конфигурацию ПЭВМ, подключая дополнительные периферийные устройства. Очень часто при описании системного блока применяется понятие «материнская (системная) плата»; это печатная плата, на которой установлены основные компоненты ПК:

МП, память (постоянная, оперативная, видео), адаптеры ввода-вывода и др. Системная плата определяет шины, используемые в ПК. Печатная плата — это пластина из диэлектрика (например, из текстолита), на которой специальными методами, например травлением или электрохимическим осаждением, создают проводники, соединяющие электронные устройства (транзисторы, интегральные схемы и др.), закрепленные на этой пластине.

В системный блок может быть встроено звуковое устройство, с помощью которого пользователю удобно следить за работой машины, вовремя заметить сбой в работе отдельных устройств или необычные ситуации, возникающие при решении задачи на ПЭВМ. Со звуковым устройством часто связан таймер, позволяющий вести отсчет времени работы машины, фиксировать календарное время, указывать на окончание заданного промежутка времени при выполнении той или иной задачи.

Микропроцессор (МП). Центральный микропроцессор является ядром любой ПЭВМ. Он выполняет функции обработки информации и управления работой всех блоков ПК. Конструктивно МП, как правило, выполнен на одном кристалле (на одной сверхбольшой интегральной схеме (СБИС)). В состав МП входят:

— арифметико-логическое устройство;
— центральное устройство управления;
— внутренняя регистровая память;
— кэш-память;
— схема обращения к оперативной памяти;
— схема управления системной шиной и др.

Рассмотрим структуру и функционирование микропроцессора на примере разработанной фирмой Intel модели i486.

АЛУ выполняет логические операции, а также арифметические операции в двоичной системе счисления и в двоично-десятичном коде, причем арифметические операции над числами, представленными в формате с плавающей точкой, реализуются в специальном блоке. В некоторых конфигурациях с этой целью используется *арифметический сопроцессор* (например, i80387). Он имеет собственные регистры данных и управления, работает параллельно с центральным МП, обрабатывает данные, представленные в формате с плавающей точкой.

Память микропроцессора состоит из функциональных регистров: регистров общего назначения, указателя команд, регистров флагов и регистров сегментов.

Восемь 32-разрядных регистров общего назначения используются для хранения данных и адресов. Они обеспечивают работу с данными разрядностью 1, 8, 16, 32 и 64 бита и адресами размером 16 и 32 бита. Каждый из таких регистров имеет свое имя, например EAX или ESP.

32-разрядный указатель команд содержит смещение при определении адреса следующей команды; 32-разрядный регистр флагов указывает признаки результата выполнения команды.

Регистры сегментов содержат значения секторов сегментов, определяющих текущие адресуемые сегменты памяти.

Кроме вышеуказанных, регистровая память МП содержит регистры процессора обработки чисел, представленных в формате с плавающей точкой, системные и некоторые другие регистры.

Производительность микропроцессора значительно повышается за счет буферизации часто используемых команд и данных во внутренней кэш-памяти размером (в данном случае) 8 Кбайт. При этом сокращается число обращений к внешней памяти. Внутренняя кэш-память имеет несколько режимов работы, что обеспечивает гибкость отладки и выполнения рабочих программ.

Устройство управления микропроцессорного типа обеспечивает конвейерную обработку данных с помощью блока предварительной выборки (очереди команд), а также многозадачность, т. е. способ организации работы ПЭВМ, при котором в ее памяти одновременно содержатся программы и данные для выполнения нескольких задач. В составе МП i486 имеются аппаратно-программные средства, позволяющие эффективно организовать многозадачный режим, в том числе системы прерываний и защиты памяти.

Система прерываний обрабатывает запросы на прерывание как от внешних устройств, так и от внутренних блоков МП. Поступление запроса на прерывание от внутреннего блока МП свидетельствует о возникновении исключительной ситуации, например о переполнении разрядной сетки. Внешнее прерывание может быть связано с обслуживанием запросов от периферийных устройств. Требуя своевременного обслуживания, внеш-

нее устройство посылает запрос прерывания микропроцессору. В ответ он приостанавливает нормальное выполнение текущей программы и переходит на обработку этого запроса, чтобы затем выполнить определенные действия по вводу-выводу данных. После совершения таких действий происходит возврат к прерванной программе. МП i486 способен обрабатывать до 256 различных типов прерываний, причем первые 32 типа отведены для внутрисистемных целей и недоступны пользователю.

Защита памяти от несанкционированного доступа в многозадачном режиме осуществляется с помощью системы привилегий, регулирующей доступ к тому или иному сегменту памяти в зависимости от уровня его защищенности и степени важности.

Обмен информацией между блоками МП происходит через *магистраль микропроцессора,* включающую 32-разрядную шину адреса, 32-разрядную двунаправленную шину данных и шину управления.

Шина адреса используется для передачи адресов ячеек памяти и регистров для обмена информацией с внешними устройствами.

Шина данных обеспечивает передачу информации между МП, памятью и периферийными устройствами. По этой шине возможна пересылка 32-, 16- и 8-разрядных данных. Шина двунаправленная, т. е. позволяет осуществлять пересылку данных как в прямом, так и в обратном направлении.

Шина управления предназначена для передачи управляющих сигналов — управления памятью, управления обменом данных, запросов на прерывание и т. д.

Системная магистраль представляет собой совокупность шин (кабелей), используемых для передачи данных, адресов и управляющих сигналов. Количество линий в адресно-информационной шине определяется разрядностью кодов адреса и данных, а количество линий в шине управления — числом управляющих сигналов, используемых в ПЭВМ.

Внутренняя память ПЭВМ состоит из оперативной памяти и постоянной памяти (ПП).

Оперативная память (ОП) ПЭВМ. ОП построена на больших интегральных схемах (БИС) или сверхбольших интегральных схемах (СБИС) и является энергозависимой: при отключе-

нии питания информация в ОП теряется. В оперативной памяти хранятся исполняемые машинные программы, исходные и промежуточные данные и результаты. Емкость ОП в ПЭВМ измеряется в килобайтах и мегабайтах. Иногда адресное пространство увеличивается до нескольких гигабайт. В наиболее распространенных конфигурациях ПЭВМ емкость ОП составляет 512—1024 Мбайт и более.

В ОП обычно выделяется область, называемая **стеком**. Обращение к *стековой памяти* возможно только в той ячейке, которая адресуется указателем стека. Стек удобен при организации прерываний и обращении к подпрограммам.

Постоянная память (ПП) ПЭВМ. ПП является энергозависимой, используется для хранения системных программ, в частности так называемой базовой системы ввода-вывода (BIOS — Basic Input and Output System), вспомогательных программ и т. п. Программы, хранящиеся в ПП, предназначены для постоянного использования микропроцессором.

Контроллеры (К) служат для управления внешними устройствами (ВУ). Каждому ВУ соответствует свой контроллер. Электронные модули-контроллеры реализуются на отдельных печатных платах, вставляемых внутрь системного блока. Такие платы часто называют **адаптерами ВУ** (от «адаптировать» — приспосабливать). После получения команды от МП контроллер функционирует автономно, освобождая МП от выполнения специфических функций, требуемых для того или другого конкретного ВУ.

Контроллер содержит регистры двух типов — регистр состояния (управления) и регистр данных. Эти регистры часто называют **портами** ввода-вывода. За каждым портом закреплен определенный номер — адрес порта. Через порты пользователь может управлять ВУ, используя команды ввода-вывода. Программа, выполняющая по запросу из основной выполняемой программы операции ввода-вывода для конкретного устройства или группы устройств ПЭВМ, входит в состав ядра операционной системы ПЭВМ.

Для ускорения обмена информацией между МП и внешними устройствами в ПЭВМ используется прямой доступ к памяти (ПДП). Контроллер ПДП, получив сигнал запроса от внешнего устройства, принимает управление обменом на себя и обес-

печивает обмен данными с ОП, минуя центральный МП. В это время микропроцессор продолжает без прерывания выполнять текущую программу. Прямой доступ к памяти, с одной стороны, освобождает МП от непосредственного обмена между памятью и внешними устройствами, а с другой — позволяет значительно быстрее, по сравнению с режимом прерывания, удовлетворить запросы на обмен.

Внешние запоминающие устройства (ВЗУ) ПЭВМ. В качестве ВЗУ в ПК в основном используются НГМД и НЖМД типа «винчестер».

Накопители на гибких магнитных дисках служат для хранения программ и данных небольшого объема и удобны для перенесения информации с одной ПЭВМ на другую.

На рабочей поверхности диска (дискеты) по концентрическим окружностям, размещенным на определенном расстоянии от центрального отверстия, записываются данные. Стандартный формат дискеты для IBM PC и совместимых с ней ПЭВМ имеет 40 (80) дорожек. Каждая дорожка разделена на части, называемые **секторами** или **записями**. Секторы представляют собой основную единицу хранения информации на дискете. При чтении или записи устройство всегда считывает или записывает целое число секторов независимо от объема запрашиваемой информации.

Емкость сектора (число байт или слов) — основная характеристика формата данных на носителе. Она определяется наименьшим количеством данных, которое может быть считано или записано на дискету за одну операцию ввода-вывода.

Существует два способа разбивки (разметки) дорожек на секторы — фиксированный (или аппаратный) и программный. Если размер сектора задан жестко и определяется механическими характеристиками устройства, разметка называется **фиксированной**. При такой разметке индексные отверстия, расположенные по кругу, обозначают начало каждого сектора; следовательно, его положение на дискете точно определено.

Для дискет ПЭВМ расположение дорожек на дискете и число сторон неизменны: они определяются характеристиками самих дискет. Однако количество секторов на дорожке и их размер могут определяться программно в процессе разметки (форматирования). Именно поэтому гибкие диски называют также

дисками с программной разметкой секторов (soft-sector). Форматирование выполняется либо программами операционной системы, либо программами BIOS (Basic Input/Output System) — базовой системы ввода-вывода.

Широкое распространение получили НГМД диаметром 3 дюйма. Их емкость достигает 1,44 Мбайт. Достоинством этих НГМД является не только большая компактность, но и наличие жесткого пластикового корпуса со специальной металлической сдвигающейся крышкой, защищающего рабочие поверхности дискеты от загрязнения и механических повреждений. Специальные сдвигающиеся рычажки на корпусе дискеты обеспечивают ее механическую защиту от записи.

Накопители на жестких магнитных дисках содержат несколько дисков, объединенных в пакет, чаще всего из четырех — шести дисков. НЖМД несменяем, располагается внутри системного блока.

В НЖМД магнитные головки, объединенные в блок, перемещаются одновременно в радиальном направлении по отношению к дискам. Дорожки с одинаковыми номерами на разных поверхностях дисков образуют цилиндр, который имеет тот же номер, что и объединенные им дорожки.

Любой диск имеет физический и логический формат. Физический формат диска определяет размер сектора (в байтах), число секторов на дорожке (или — для жестких дисков — в цилиндре), число дорожек (цилиндров) и число сторон. Логический формат диска задает способ организации информации на нем и фиксирует размещение информации различных типов.

Важным параметром для пользователя является *время доступа,* характеризующее скорость чтения и записи информации на диски. Для наиболее распространенных НЖМД оно колеблется от 14 до 70 мкс. Реальная скорость работы НЖМД существенно зависит от типа используемой программы. Так, обработка больших массивов информации, требующая многократного поиска одиночных сведений, может неожиданно для пользователя занять весьма значительное время. Еще более продолжительной может оказаться обработка сложных изображений.

Главный недостаток магнитной записи — ее недолговечность и чувствительность к внешним магнитным полям.

Оптические диски — это устройства для хранения информации на оптических (лазерных) дисках. Их емкость измеряется гигабайтами и даже десятками гигабайт; однако в большинстве случаев такие диски не допускают перезаписывания, поэтому используются большей частью для хранения постоянной информации (например, сложных компьютерных игр с высокоразвитой графикой).

Отличие оптической записи от магнитной заключается в полном отсутствии физического контакта механизма дисковода с поверхностью оптического диска. Запись и считывание информации производятся бесконтактно, с помощью лазерного луча. К тому же луч фокусируется не на поверхности, а в глубине прозрачного диска. Поэтому оптической записи не страшны неглубокие царапины на поверхности оптического диска. Следовательно, он обеспечивает очень высокую долговечность и надежность хранения информации. К тому же оптические диски совершенно нечувствительны к внешним магнитным полям, однако боятся глубоких царапин и прямых солнечных лучей.

Основная возможность увеличения емкости оптических дисков — уменьшение расстояний между дорожками и размеров пит за счет уменьшения длины волны лазерных диодов. Пит (от англ. pit — углубление) — это микровпадина на рабочей поверхности компакт-диска, которая определяет значение бита.

К оптическим относятся CD-R(RW)- и DVD-R(RW)-диски. При массовом производстве они изготавливаются с помощью штампов и предназначены для чтения и записи информации.

Аудиокомпакт-диски могут проигрываться как в музыкальных центрах, CD-проигрывателях и плеерах, так и с помощью дисководов ПК. Время звучания этих дисков может составлять несколько часов. Время просмотра видеофильмов на мониторе ПК в стандарте VideoCD составляет 74 мин. В настоящее время разработана технология сжатия информации в формате MPEG, с помощью которой обеспечивается коэффициент сжатия до 200 : 1 и более.

Кроме дисков «только для чтения информации», существуют диски для однократной (CD-R) и многократной (CD-RW) записи информации.

Только для чтения — CD-ROM. Технология, объединяющая текст и графику со звуком и движущимися изображениями, на-

зывается **мультимедиа**. При этом подразумевается обратная связь — действия пользователя должны напрямую и существенно влиять на происходящее в системе мультимедиа.

В качестве носителей информации в мультимедийных компьютерах используются оптические компакт-диски CD-ROM (Compact Disk Read Only Memory, т. е. память на компакт-диске «только для чтения»). Внешне они не отличаются от звуковых компакт-дисков, используемых в проигрывателях и музыкальных центрах. Емкость одного CD-ROM достигает 700 Мбайт. Для чтения компакт-дисков используется CD-дисковод. Скорость чтения данных в нем зависит от скорости вращения диска. Сейчас применяются уже 24-, 32-, 40- и 50-скоростные дисководы, а скорость считывания информации при этом приближается к скорости считывания с «винчестера». Компакт-диск так же легко сменить, как и дискету. Информация на него записывается один раз. Кстати, с помощью CD-дисковода можно проигрывать и звуковые компакт-диски (разумеется, при наличии в ПК звуковой карты и звуковых колонок).

Замечание для пользователей CD-ROM: безудержное повышение скорости считывания носит чисто рекламный характер. Для считывания практически любых дисков вполне достаточно 10—16-скоростных дисководов. На бóльших скоростях начинают сказываться малейшие искривления дисков, известны даже случаи их разрушения.

Диск CD-ROM содержит три слоя — подложку из поликарбоната с отштампованным рельефом диска, напыленное на нее отражающее покрытие из алюминия, серебра или золота и тонкий защитный слой из поликарбоната или лака — на него наносятся рисунки и подписи. Некоторые «пиратские» диски имеют слишком тонкий защитный слой либо лишены его совсем, поэтому их легко повредить. Информация на диске кодируется чередованием пит и промежутков между ними, расположенных вдоль дорожки.

Что представляет собой дисковод CD-ROM? В его состав входят плата электроники, шпиндельный двигатель, устройство загрузки диска и система считывающей оптической головки.

Во вращение диск приводит шпиндельный двигатель. В состав системы считывающей головки входят сама оптическая головка и система ее перемещения. В головке находятся лазер-

ный излучатель (на основе лазерного инфракрасного светодиода), система фокусировки лазерного луча (объектив), фотоприемник и предварительный усилитель. Система перемещения головки содержит собственный двигатель, который с помощью червячной или зубчатой передачи приводит в движение каретку с оптической считывающей головкой.

На одной из международных книжных ярмарок в 1999 г. было представлено московское издательство «МЦФ», специализирующееся на издании CD-ROM. Изданы диски с записями собраний сочинений А.С. Пушкина, А.П. Чехова, Ф.М. Достоевского, энциклопедий фантастики и истории России, музыкального цикла «Бранденбургские концерты» И.С. Баха и др.

Записываемые оптические диски — CD-R (R — от англ. Record — запись). Принцип однократной записи на диске CD-R (CD-Recordable) основан на «выжигании» лучом лазера бит информации на записывающем слое диска, состоящем из органического красителя. Краситель имеет свойство однократно изменять отражающую способность диска. При считывании лазерным лучом и фиксируется это изменение отражательной способности.

Перезаписываемые оптические диски — CD-RW. Многократная запись на диске CD-RW (CD-ReWritable) (ReWritable — от англ. Rewrite — перезапись) производится несколько иначе. В данном случае применяется специальный комбинированный слой, который при нагреве лазерным лучом способен многократно менять характеристики. При этом вещество такого слоя может многократно переходить из кристаллического состояния в аморфное и обратно. В кристаллическом состоянии вещество комбинированного слоя хорошо отражает свет лазера, в аморфном — плохо. Изменение отражающей способности фиксируется лазерным лучом при считывании информации с диска.

На компакт-диск можно записать любую понравившуюся музыку, причем прослушивать ее можно не только на компьютере, но и на любом CD-плеере или музыкальном центре с CD-дисководом. Если записи делаются для длительного пользования и хранения, то для этого больше подходят более дешевые записываемые диски CD-R.

Записываемый диск CD-R читается с помощью любого дисковода CD-ROM. Запись информации на диски CD-R пред-

ставляет собой самый дешевый и оперативный способ хранения больших объемов данных.

Для оперативного хранения информации больше подходят перезаписываемые диски CD-RW.

Цифровой диск общего назначения — DVD. На смену существующим компакт-дискам пришел новый стандарт носителей информации — DVD (Digital Versatile Disc), или цифровой диск общего назначения. Его внешний вид и геометрические размеры не отличаются от компакт-дисков. Основное отличие DVD-диска — более высокая плотность записи информации. Он вмещает в семь — десять раз больше информации, чем CD. Этого удалось достичь благодаря более короткой длине волны лазера и меньшему размеру пятна сфокусированного луча, что дало возможность вдвое уменьшить расстояние между дорожками. Кроме того, DVD-диски могут иметь один или два слоя информации. К ним можно обращаться, регулируя положение лазерной головки. У DVD-диска каждый слой информации вдвое тоньше, чем у CD-диска. Поэтому можно соединять два диска толщиной 0,6 мм в один со стандартной толщиной 1,2 мм. При этом емкость удваивается.

В 1998 г. фирма Sony выпустила на рынок компакт-диски емкостью 2,6 Гбайт и записываемые компакт-диски емкостью 650 Мбайт с возможностью однократной (DVD-R) и даже многократной (DVD-RAM) записи информации. RAM (от англ. Random Access Memory) — память прямого доступа.

Дисководы DVD представляют собой несколько усовершенствованные дисководы CD-ROM.

Внешние устройства ПЭВМ. Эффективность использования ПЭВМ в большой степени определяется количеством и типами внешних устройств, которые могут применяться в ее составе. Внешние устройства обеспечивают взаимодействие пользователя с ПЭВМ. Широкая номенклатура внешних устройств, разнообразие их технико-эксплуатационных и экономических характеристик дают возможность пользователю выбрать такие конфигурации ЭВМ, которые в наибольшей мере соответствуют его потребностям и обеспечивают рациональное решение его задач.

Внешние устройства составляют до 80 % стоимости ПЭВМ и оказывают значительное (иногда даже решающее) влияние на характеристики машины в целом.

Конструктивно каждая модель ПЭВМ имеет так называемый базовый набор внешних устройств: клавиатуру, дисплей, НЖМД и один или два НГМД, — составляющий вместе с системным блоком «базовую конфигурацию» данной модели. Пользователь, как правило, сам подбирает печатающее устройство. В случае необходимости к ПЭВМ могут подключаться и дополнительные внешние устройства, например сканеры, плоттеры или диджитайзеры. В последние годы разрабатываются совершенно новые виды внешних устройств, ориентированные на стремительно растущие запросы пользователей, в частности для приложений в области мультимедиа.

Клавиатура (клавишное устройство) реализует диалоговое общение пользователя с ПЭВМ:

— ввод команд пользователя, обеспечивающих доступ к ресурсам ПЭВМ;

— запись, корректировку и отладку программ;

— ввод данных и команд в процессе решения задач.

Центральную часть клавиатуры обычно занимают клавиши букв латинского и русского алфавитов, служебных знаков («!», ««», «:», «%» и др.), а также цифровые клавиши. В большинстве случаев одна клавиша используется для ввода нескольких разных знаков, причем различие между ними производится путем одновременного нажатия соответствующей клавиши и одной и/или двух служебных функциональных клавиш (обычно — клавиш <*Shift*>, <*Alt*>, <*Ctrl*>). В большинстве моделей клавиатуры (за исключением клавиатуры ПЭВМ классов LAPTOP, NOTEBOOK, HANDHELD) с правой стороны размещается дополнительная цифровая клавиатура, что создает удобства при необходимости частого ввода чисел. По периферии клавиатуры размещаются служебные функциональные клавиши: <*Enter*>, <*Esc*>, <*Delete*>, <*Insert*>, <*Tab*> и др., а также «программируемые» функциональные клавиши (<*F1*> — <*F12*>). Функциональные клавиши в программах выполняют в основном специальные операции. К примеру, клавиша <*Esc*> обычно означает *Отменить* или *Вернуть*, клавиша <*Insert*> — *Вставить* и т. п. Назначение программируемых функциональных клавиш <*F1*> — <*F12*> более гибко; как правило, оно определяется в соответствующих программах и приводится в их документации. Служебные клавиши (<*Shift*>, <*Ctrl*>, <*Alt*>) и индикаторы режимов (<*Print screen*>, <*Caps Lock*>, <*Break*>) служат для переключе-

ния назначения алфавитно-цифровых клавиш, вывода «образа экрана дисплея» на принтер, изменения режима работы и прерывания программ. Клавиши управления (<←>, <↑>, <↓>, <→>) необходимы для позиционирования курсора на экране дисплея. Ряд клавиш обеспечивают перемещение курсора в начальную или конечную позицию на строке экрана дисплея (соответственно <*Home*> и <*End*>), а также на страницу вперед или назад (соответственно <*Pg Up*> и <*Pg Dn*>).

Типовые размеры клавиатуры: $40 \times 450 \times 180$ мм. Она должна обеспечить минимальное количество нажатий на клавиши пользователем. Это достигается изменением значений отдельных клавиш программой. Клавиатура ПЭВМ передает МП не код символа, а порядковый номер нажатой клавиши и продолжительность времени каждого нажатия. Интерпретация смысла нажатой клавиши выполняется программой. Таким образом, кодировка клавиши не зависит от кодировки символов, что значительно упрощает работу с клавиатурой.

Дисплей (монитор) (от англ. display — показывать) — основное устройство для отображения информации, выводимой на него во время работы программ на ПЭВМ. Дисплеи могут существенно различаться по своим характеристикам, от которых зависят возможности машин и используемого программного обеспечения. Одни дисплеи выводят только алфавитно-цифровую информацию, другие — только графическую. Не менее важные признаки — возможность поддержки цветного или только монохромного изображения, текстовый формат и разрешающая способность изображения. Текстовый формат (в текстовом режиме) характеризуется числом символов в строке и числом текстовых строк на экране. В графическом режиме разрешающая способность задается числом точек по горизонтали и числом точечных строк по вертикали. Существенными техническими параметрами дисплеев являются количество поддерживаемых уровней яркости в монохромном режиме и количество цветов — при цветном изображении, а также размер экрана: он определяет различимость изображения в целом и четкость его отдельных элементов, в том числе букв и цифр.

Перечисленные параметры зависят как от конструкции экрана, так и от схемы управления, сосредоточенной в системном блоке. В настоящее время в большинстве случаев применяется схема формирования изображения на основе растровой памяти

(bit mapping). Каждый элемент изображения — одна точка на экране дисплея — формируется из фрагмента растровой памяти, состоящего из одного, двух или четырех бит. Информация, записанная в указанных битах, управляет яркостью (или цветом) точки на экране, а также ее миганием и другими возможными атрибутами.

Объем растровой памяти прямо связан с разрешающей способностью дисплея. К примеру, дисплею с двумя уровнями яркости и разрешающей способностью 640 × 480 точек требуется 26 Кбайт растровой памяти. Если же при этом необходимо управлять 16 цветами для каждой точки, требуемый объем растровой памяти составит не менее 64 Кбайт, а при двухцветном экране с разрешающей способностью 1024 × 768 потребуется уже 132 Кбайт растровой памяти. При таком методе управления изображением знаки выводятся на экран с помощью специальных знакогенераторов — особых электронных схем, управляемых точечными матрицами, на которых формируется изображение каждого символа.

Большинство профессиональных ПЭВМ снабжено дисплеями, основанными на монохромных или цветных *электронно-лучевых трубках (ЭЛТ)*. Наиболее часто в IBM-совместимых ПЭВМ используются мониторы типа VGA или SVGA, а в более ранних моделях — CGA, EGA, Hercules.

В профессиональных ПЭВМ широко применяются цветные мониторы с очень высоким разрешением (1024 × 768 и 1600 × × 1200 точек) и возможностью получения изображений из 4096 базовых цветов, что обеспечивает до 16 млн оттенков.

Пользователи ПЭВМ проводят в непосредственной близости от работающих дисплеев многие часы подряд. В связи с этим фирмы — производители дисплеев усилили внимание к оснащению их экранов специальными средствами защиты от всех видов негативных воздействий на организм пользователя. Так, фирма Samsung выпускает дисплеи «Low Radiation» с нанесенным на экран специальным покрытием, снижающим уровень жесткого излучения.

Кроме мониторов с ЭЛТ, существуют мониторы с жидкокристаллическим экраном и плазменные.

Действие *жидкокристаллического* монитора LCD (Liquid Crystal Display) или любого индикатора, например часов или калькуляторов, основано на использовании вещества, находя-

щегося в жидком состоянии, но при этом обладающего некоторыми свойствами кристаллических тел. Молекулы таких жидких кристаллов под действием электрического поля способны изменять свою ориентацию и свойства проходящего сквозь них светового луча. Так, в жидкокристаллических индикаторах изображение создается в результате изменения электрического напряжения и ориентации молекул.

Действие *плазменного* монитора PDP (Plasma Display Panels) похоже на работу неоновой лампы. Каждая ячейка плазменной панели выполнена в виде плоской стеклянной трубки, заполненной инертным газом под низким давлением. Внутри трубки помещено два электрода. При подаче напряжения между ними возникает электрический (так называемый тлеющий) разряд и вызывает свечение. В плазменных экранах пространство между двумя стеклянными поверхностями заполняется, как и в неоновой лампе, инертным газом — аргоном или неоном. На стеклянную поверхность помещают маленькие прозрачные электроды, на которые подается высокочастотное напряжение: образуется целое поле миниатюрных точечных неоновых лампочек. Под действием напряжения в газовой области, прилегающей к электроду, возникает электрический разряд. Плазма этого разряда излучает свет в ультрафиолетовом диапазоне спектра, а он, в свою очередь, вызывает свечение частиц люминофора в видимой человеком части спектра. То есть каждый пиксель на экране работает подобно маленькой лампе дневного света.

Общение пользователя с ПЭВМ облегчается с помощью различных манипуляторов. Наиболее распространенный из них — так называемая **мышь**: небольшая коробочка с двумя или тремя кнопками (иногда их называют клавишами) и утопленным, свободно вращающимся в любом направлении шариком на нижней поверхности. «Коробочка» подключается к компьютеру с помощью специального кабеля. Пользователь, перемещая мышь по поверхности стола (обычно ее размещают на специальном резиновом коврике), позиционирует указатель мыши (стрелку, прямоугольник) на экране дисплея, а нажатием соответствующих кнопок выполняет определенное действие (например, определенный пункт меню). Мышь требует специальной программной поддержки.

В портативных ПЭВМ мышь обычно заменяют ***трекболом*** — встроенным в клавиатуру шариком на подставке с двумя кноп-

ками (клавишами) по бокам. Позиционирование указателя трекбола на экране дисплея производится вращением этого шарика. Кнопки трекбола имеют то же назначение, что и кнопки мыши. Несмотря на наличие трекбола, пользователь портативной ПЭВМ может использовать и обычную мышь, подключив ее к соответствующему порту.

Для непосредственного считывания графической информации с бумажного или иного носителя в ПЭВМ применяются оптические ***сканеры*** — настольные и ручные. *Настольные* сканеры позволяют обрабатывать весь лист бумаги или пленки целиком, *ручные* — проводят над нужными рисунками или текстом, обеспечивая их считывание. Введенный с использованием сканера рисунок распознается ПЭВМ с помощью специального программного обеспечения. Рисунок может быть не только сохранен, но и откорректирован по желанию пользователя соответствующими графическими пакетами программ. В настоящее время выпускают черно-белые и цветные сканеры с точностью разрешения до 8000 точек на дюйм (более 300 точек на 1 мм), однако эти устройства весьма дороги. Использование сканеров для непосредственного ввода в ПЭВМ текстовой информации с ее последующим редактированием затруднено и значительной сложностью программного обеспечения, необходимого для правильного распознавания и интерпретации отдельных символов.

Для ввода рисунков в ПЭВМ применяют также ***световое перо*** и различные ***диджитайзеры***.

К ручным манипуляторам относится и ***джойстик*** (от англ. joystick) — подвижная рукоять с одной или двумя кнопками, с помощью которой можно позиционировать указатель на экране дисплея. Кнопки джойстика имеют то же назначение, что и кнопки мыши. Джойстик чаще применяется в бытовых ПЭВМ, в первую очередь для компьютерных игр.

Печатающие устройства ПЭВМ. В ПЭВМ используются матричные, лепестковые, струйные и лазерные принтеры (от англ. printer — печатник).

Матричные принтеры наиболее распространены. Печатаемые знаки синтезируются в них с помощью игольчатой матрицы (головки), двигающейся вдоль каждой печатаемой строки по специальной направляющей и ударяющей по красящей ленте. Чаще всего применяются принтеры с 9- и 24-игольчатыми головками. Эти принтеры позволяют получить вполне прием-

лемое для большинства приложений качество печати, в том числе за счет многократных проходов при печати одной строки с небольшими смещениями. Однако это снижает и без того невысокую скорость печати. Недостатком матричных принтеров следует считать и довольно значительный уровень производимого при печати шума.

Лазерные принтеры располагают многообразными возможностями печати, обеспечивают ее высокое качество при значительной скорости. Они имеют собственный расширяемый блок памяти, позволяют масштабировать шрифты. «Паспортная» скорость печати у различных моделей лазерных принтеров, как правило, колеблется от 4 до 64 страниц в минуту. Вместе с тем она зависит от объема собственной памяти принтера (и может заметно сокращаться при ее недостатке для конкретной печатаемой информации) и сложности выводимого изображения.

Особенно эффективны лазерные принтеры при изготовлении оригинал-макетов книг и брошюр, рекламных проспектов, деловых писем и иных материалов, требующих высокого качества. Они позволяют с большой скоростью печатать графики и рисунки. В последние годы появилась целая гамма лазерных принтеров, обеспечивающих не только черно-белую, но и многокрасочную цветную печать.

Недостатком лазерных принтеров являются довольно жесткие требования к качеству бумаги — она должна быть достаточно плотной и нерыхлой, недопустима печать на бумаге с пластиковым покрытием и т. д.

Даже самые простые модели лазерных принтеров в пять — десять раз дороже средних моделей матричных принтеров, а цена цветных лазерных принтеров значительно превосходит цену матричных. Весьма дороги и сменные картриджи, содержащие красящий порошок. Все это делает лазерные принтеры малопригодными для изготовления значительных тиражей, поскольку печать одного листа обходится существенно дороже ксерокопии.

Струйные принтеры в последние годы получают все более широкое распространение среди пользователей ПЭВМ. Этот тип принтера занимает промежуточное положение между матричным и лазерным. Струйные принтеры, как и матричные, печатают построчно, но при этом обеспечивают качество пе-

чати, приближающееся к лазерным принтерам. Они просты в эксплуатации и работают практически бесшумно, а под управлением соответствующих программных средств позволяют печатать вполне удовлетворительные по качеству графические материалы. Вместе с тем струйные принтеры ненамного превосходят матричные по скорости печати, хотя стоят в два-три раза дороже последних. Струйные принтеры вполне успешно применяются во всех случаях, когда скорость и качество печати не являются критическими факторами. Красящая жидкость («чернила») для них выпускается в специальных компактных картриджах; производится нескольких цветов таких «чернил», так что простой заменой картриджа можно обеспечить печать многоцветных изображений.

Графопостроители (плоттеры) применяются для вывода графической информации из ПЭВМ. Плоттеры значительно дешевле, чем лазерные принтеры, хотя их скорость вывода изображений значительно ниже. Достоинством плоттеров, по сравнению с лазерными принтерами, является возможность использования для печати крупноформатной бумаги и пленки (вплоть до формата A0). Плоттеры выпускаются двух типов — рулонные и планшетные. В *рулонных* плоттерах бумажный лист перемещается транспортирующим валиком в вертикальном направлении, а пишущий узел — в горизонтальном. Рулонные плоттеры позволяют получать полноцветные изображения хорошего качества. В *планшетных* плоттерах лист бумаги фиксируется горизонтально на плоском столе, а пишущий узел (одно или несколько разноцветных перьев) перемещается по направляющим в двух направлениях — по осям X и Y. Планшетные плоттеры обеспечивают более высокую, по сравнению с рулонными, точность печати рисунков и графиков.

3.1.7. Правила техники безопасности при работе на компьютере

Компьютер, при работе с которым организм человека не подвергается опасности, называется **безопасным**.

Считается, что основное вредное воздействие на пользователя оказывают монитор на базе электронно-лучевой трубки и электромагнитные поля, генерируемые компьютером.

Для ограничения вредных воздействий компьютера Департамент труда Швеции в 1987 г. принял стандарт MPR I. В 1990 г. принят более жесткий стандарт — MPR II. На смену MPR II пришли еще более жесткие требования шведского объединения профсоюзов TCO'92 (1992 г.), TCO'95 (1995 г.) и TCO'99 (1999 г.). Эти стандарты приняты многими странами. Мониторы, удовлетворяющие этим стандартам, имеют марку LR (Low Radiation — низкое излучение).

Госсанэпиднадзор России в 1996 г. выпустил Санитарные правила и нормы 2.2.2.542—96 «Гигиенические требования к видеодисплейным терминалам, персональным электронно-вычислительным машинам и организации работ», которые по многим параметрам соответствуют шведским.

По этим правилам продолжительность непрерывной работы взрослого пользователя компьютера не должна превышать двух часов, ребенка — от 10 до 20 мин в зависимости от возраста. Минимальный перерыв определен в 15 мин. Расстояние от глаз пользователя до экрана монитора должно составлять не менее 50 см, оптимально 60—70 см. Расстояние от экрана монитора до задней стенки монитора соседнего ряда должно быть не менее 2 м, а расстояние между боковыми стенками — не менее 1,2 м. Площадь рабочего пространства, приходящаяся на одного взрослого пользователя, должна составлять не менее 6 $м^2$.

Тема 3.2

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ КОМПЬЮТЕРА. ОПЕРАЦИОННАЯ СИСТЕМА

3.2.1. Программное обеспечение компьютера

Функционирование любого компьютера требует различных видов обеспечения: математического, алгоритмического, информационного, программного, технологического, лингвистического, организационного, эргономического, правового и др. Важнейшее значение имеет программное обеспечение (ПО) ЭВМ. **Программное обеспечение** (от англ. software) — это совокупность программных средств для обеспечения нормальной

работы ЭВМ. ПО любого компьютера в основном определяет его интеллектуальные возможности, профессиональную направленность, широту и полноту осуществления устройств (блоков). Комплекс ПО должен охватывать множество функций ЭВМ, таких, как обеспечение организации решения функциональных задач пользователя, организация диалога пользователя и ЭВМ, управление базами данных, трансляция программ, осуществление сервисных программ, облегчающих работу на компьютере. Развитость ПО определяет функциональную полноту и разнообразие сервисной поддержки пользователя.

ПО любой ЭВМ подразделяют на системное (общее) и прикладное (синонимы — приложение, специальное ПО). Основные элементы ***системного ПО*** обычно поставляются вместе с ПЭВМ. К ним относятся операционные системы (ОС) и оболочки, программные средства (ПС) ведения баз данных, организации диалога, а также программы, расширяющие возможности ОС. Главное предназначение этой части ПО — управление работой процессора, организация интерфейса между пользователем и ПЭВМ, организация доступа к памяти, периферийным устройствам и сети, управление файлами, запуск прикладных программ и управление процессом их выполнения, трансляция и выполнение программ, подготовленных на различных алгоритмических языках.

Прикладное ПО, как правило, состоит из уникальных программ и функциональных пакетов прикладных программ. Именно от функционального ПО зависят вид, содержание и конкретная специализация пользователя. Учитывая, что прикладное ПО в конечном счете определяет область применения ПЭВМ и состав решаемых пользователем задач, оно должно создаваться на основе инструментальных программных средств диалоговых систем, ориентированных на решение конкретного класса задач со схожими функционально-техническими особенностями обработки информации.

Прикладное ПО ЭВМ должно обладать свойствами адаптивности (приспосабливаемости), гибкости, модифицируемости и настраиваемости на конкретное применение в соответствии с требованиями пользователя.

3.2.2. Системное и прикладное программное обеспечение

Принципиальный состав ПО ПЭВМ приведен на рис 3.5. Рассмотрим основные его компоненты.

Операционная система (ОС) — это комплекс программ, обеспечивающий управление ЭВМ как единым целым (тогда как на самом деле компьютер состоит из многих частей), осуществляющий взаимодействие ЭВМ с окружающей средой, а именно: с человеком, прикладными программами, другими системами. ОС — главная часть системного ПО, она управляется командами. Более подробно назначение и основные функции ОС рассматриваются в подтеме 3.2.3.

Оболочка — это программа (или комплекс программ), управляющая работой с основной программой. Иногда используются термины «среда», «окружение». Например, работать со

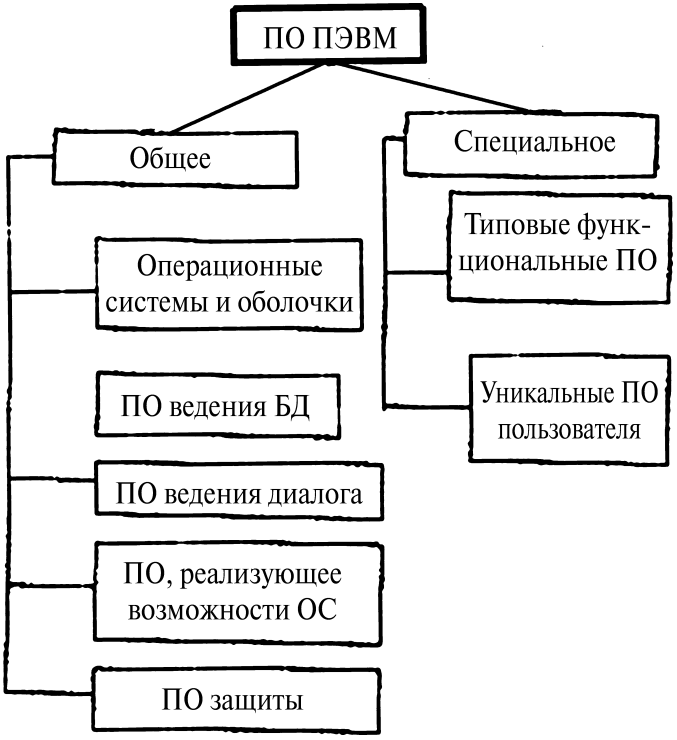


Рис. 3.5. Состав программного обеспечения ПЭВМ

старой средой ОС MS-DOS (Microsoft Disk Operation System — дисковая операционная система, разработанная в 1981 г. фирмой Microsoft) достаточно сложно — эта система управляется с помощью команд. И эти команды пользователь должен помнить. Кроме того, их нужно правильно набрать. Разработано много оболочек над данной ОС, позволяющих упростить управление ею. В первую очередь это знаменитая оболочка Norton Commander (коммандер Нортон). Широко используются оболочки для создания среды программирования. Под *средой программирования* понимается совокупность технических и программных средств, в которых функционирует система (объект). Например, фирма Borland для работы с языками Паскаль, Си, Пролог разработала оболочку и включила ее в состав языка программирования, что значительно упрощает процесс программирования. Имеется тенденция так разработать ОП, чтобы оболочки не были нужны.

Программные средства ведения баз данных. Ведение БД — это процесс поддержания базы данных, системы в актуальном состоянии, т. е. постоянное, полное и своевременное внесение в БД всех изменений. *Программное средство (ПС)* (от англ. software tools) — это программы, которые обеспечивают работу с компьютером и включают в себя утилиты, редакторы, компиляторы и пр.

Программные средства ведения диалога — это ПС, осуществляющие режим прямого взаимодействия между пользователем и компьютером, между компьютерами в сети или между компьютером и периферийным устройством, при котором связь между взаимодействующими системами не прерывается. Диалоговый режим часто называют **интерактивным режимом** или **режимом on-line**.

Программные средства, расширяющие возможности ОС, — это ПС, расширяющие возможности таких операционных систем, которые обеспечивают функционирование ЭВМ различных конфигураций. К ним относятся пакеты, обеспечивающие работу многомашинных комплексов типовых конфигураций, диалоговые системы, системы для работы в реальном масштабе времени, удаленную пакетную обработку.

Программные средства защиты информации — это развитые программные комплексы, обеспечивающие защиту информа-

ции от всех видов постороннего вмешательства. Подробно средства защиты информации рассмотрены в теме 3.4.

Наиболее динамично развивающаяся часть прикладного ПО — **пакеты прикладных программ (ППП)**. Круг решаемых с их помощью задач постоянно расширяется. Во многом внедрение компьютеров практически во все сферы деятельности стало возможным благодаря появлению новых и совершенствованию существующих ППП.

Достижения в области микроэлектроники, появление более мощных по своим функциональным возможностям компьютеров также стимулируют создание новых ППП. В свою очередь, необходимость улучшения характеристик ППП при решении конкретных задач пользователя стимулирует совершенствование архитектуры и элементной базы компьютеров и периферийных устройств.

Структура и *принципы построения* ППП зависят от класса ЭВМ и операционной системы, в рамках которой данный пакет будет функционировать. Наибольшее количество разнообразных ППП создано для IBM PC-совместимых компьютеров с операционными системами MS-DOS и Windows.

Каждая группа ППП имеет свои проблемы организации, трудности разработки и создания. Каждый пакет реализуется в соответствии с предъявляемыми к нему требованиями и возможностями конкретного языка программирования. Все ППП подразделяют на пакеты:

— расширяющие возможности операционных систем;

— общего назначения;

— ориентированные на работу в автоматизированных системах управления.

ППП общего назначения включают набор программ для алфавитно-цифровых и графических дисплеев, графопостроителей, систем программирования, а также для научно-технических расчетов, математического программирования, обработки матриц, различного вида моделирования, решения задач теории массового обслуживания и т. д.

ППП, ориентированные на работу в АСУ, включают набор программ для общецелевых систем обработки банков данных, информационных систем общего назначения, систем обработки документов.

Существующие ППП охватывают почти все сферы человеческой деятельности, связанные с обработкой информации. Развитие и совершенствование ППП — поступательный процесс, поэтому следует ожидать появления новых пакетов, возможности которых превзойдут достижения существующих.

По функциональному признаку ППП подразделяют на проблемно-ориентированные и интегрированные.

Проблемно-ориентированные ППП — наиболее развитая в плане реализуемых функций и многочисленная по количеству созданных пакетов часть ПО. Она включает следующие проблемно-ориентированные программные продукты: текстовые процессоры, издательские системы, графические редакторы, демонстрационную графику, системы мультимедиа, организаторы работ, электронные таблицы (табличные процессоры), системы управления базами данных, программы распознавания символов, финансовые и аналитико-статистические программы. Большое число проблемно-ориентированных ППП рассмотрено в разделе 4.

Интегрированные ППП — немногочисленная по количеству наименований продуктов, но довольно мощная в вычислительном плане и активно развивающаяся часть ПО. Интегрированные ППП подразделяют на две группы: традиционные, или полносвязанные, и объектно-связанные.

Идея создания интегрированных программных комплексов была реализована на всех поколениях ЭВМ.

Традиционные, или *полносвязанные, интегрированные комплексы* представляют собой многофункциональный автономный пакет, в котором в одно целое соединены функции и возможности различных специализированных (проблемно-ориентированных) пакетов, родственных в смысле технологии обработки данных на отдельном рабочем месте. Типичными представителями таких программ являются пакеты FrameWork, Symphony, а также пакеты нового поколения Microsoft Word, Lotus Works.

В настоящее время активно реализуется другой подход к интеграции программных средств: объединение специализированных пакетов в рамках единой ресурсной базы, обеспечение взаимодействия приложений (программ пакета) на уровне объектов и единого упрощенного центра — переключения меж-

ду приложениями. Интеграция в этом случае носит *объектно-связанный характер.*

Типичные и наиболее мощные пакеты данного типа — Borland Office for Windows, Lotus SmartSuite for Windows, Microsoft Office, в профессиональной редакции которых присутствуют четыре приложения: текстовый редактор, СУБД, табличный процессор, программы демонстрационной графики. Целесообразность создания таких пакетов, очевидно, связана с желанием получить дополнительный эффект от интеграции по отношению к простой сумме составляющих его компонентов. Этот эффект должен достигаться за счет согласованного взаимодействия компонентов в процессе работы пользователя.

3.2.3. Операционная система: назначение и основные функции

Программное и аппаратное обеспечение в компьютере работает в неразрывной связи и в непрерывном взаимодействии. Основные управляющие программы, рассматриваемые как единое целое, представляют собой операционную систему (ОС).

Операционная система — это совокупность программных средств, обеспечивающая управление аппаратной частью компьютера и прикладными программами, а также их взаимодействие между собой и пользователем. Операционная система ЭВМ:

— образует автономную среду, не связанную ни с одним языком программирования;

— работает на конкретной аппаратной платформе, например IBM PC, однако для одной и той же платформы может существовать несколько ОС;

— управляет работой конкретных прикладных программ, называемых **приложениями**.

Обычно файлы операционной системы хранятся на системном диске, который может быть реализован на любом внешнем носителе. При включении компьютера ОС автоматически загружается с диска в оперативную память и занимает там определенное место.

Функции операционной системы. ОС ЭВМ предназначена для:

— обеспечения нескольких видов интерфейса — аппаратно-программного (между программным и аппаратным обеспечением), программного (между разными видами программного обеспечения) и пользовательского (между пользователем и программно-аппаратными средствами);

— организации и хранения информации на внешних носителях информации.

Интерфейс (от англ. inter — между, face — лицо) — это средства и способы установления и поддерживания информационного обмена между исполнительными устройствами автоматической системы и человеком-пользователем.

Операционная система ПЭВМ представляет собой программу, которая автоматически загружается при включении ПК и содержит базовый набор команд, с помощью которых пользователь может осуществлять общение с ЭВМ и выполнять ряд действий: запуск программ, форматирование дискеты, копирование файлов и т. д. ОС обеспечивает интерфейс пользователя с ПК и обработку данных.

Операционные системы подразделяют на однопрограммные, многозадачные и многопользовательские. К *однопрограммным ОС* относятся SCP, MS-DOS и др. *Многозадачные ОС* (Unix, Windows начиная с версии 3.1, DOS 7.0, OS/2 и др.) предусматривают одновременное выполнение нескольких приложений. Если однопрограммные системы работают или в пакетном, или в диалоговом режиме, то многозадачные могут совмещать эти режимы, обеспечивая пакетную и диалоговую технологии обработки данных.

Многопользовательские ОС отвечают требованиям пользователей различных категорий (неквалифицированных пользователей, прикладных и системных программистов) и профессий. Эти ОС реализуются также сетевыми ОС Novell Netware, обеспечивают сетевые, пакетные и диалоговые технологии.

Разнообразие ПЭВМ и ОС привело к появлению нового понятия — «платформа». *Платформа* определяет тип компьютера и ОС, на которых устанавливается та или иная информационная технология. Платформа имеет сложную структуру. Главными ее компонентами являются тип компьютера, опре-

деляемый типом процессора (Intel, Motorola, Atary, Sincer и др.), и ОС, работающая на том или ином процессоре. Например, Windows NT работает на многих типах процессоров: MIPS, ALPHA, Power PC.

Тема 3.3

ФАЙЛОВАЯ СИСТЕМА. РАБОТА С НОСИТЕЛЯМИ ИНФОРМАЦИИ

3.3.1. Файловая система

Понятие файла. Для обеспечения удобства работы с записанными на диск сведениями их размещают в файлах. **Файл** — это логически связанная совокупность данных, для которой во внешней памяти отводится поименованная область (*данные* — это любая информация, включающая программу и исходные данные для их выполнения, результаты выполнения программ, тексты, иллюстрации и т. п.).

Обычно в отдельном файле хранят данные, относящиеся к одному типу. Тип данных определяет тип файла. Файл хранится в виде последовательности произвольного числа байт, обладающей уникальным именем. Файл может содержать любое число байт или быть пустым (0 байт); создать файл — значит присвоить ему имя. Уникальность имени файла гарантирует однозначность доступа к данным.

Правила задания имени файла. Составное (полное) имя файла представляет собой совокупность собственно имени файла и расширения имени файла. Имя от расширения отделяется точкой.

Расширение имени файла передает операционной системе информацию о том, к какому типу относятся данные, содержащиеся в файле, и о формате, в котором они записаны.

В семействе операционных систем MS-DOS на имя файла отводится восемь символов, а на расширение — три. При именовании файла допускается использовать цифры и символы латинского алфавита. Соглашение 8.3 назовем «коротким» именем файла.

В семействе операционных систем Windows имя файла может содержать 256 символов — «длинное» имя, любые символы,

причем кроме специальных: «/», «\», «:», «*», «"», «<», «>», «|», — можно также использовать пробелы и несколько точек. Расширением имени считаются все символы, идущие после последней точки.

Чтобы обеспечить возможность работы с файлом на других рабочих местах, лучше использовать «короткое» имя файла.

Параметры (свойства), характеризующие файл:

— полное имя;

— объем в байтах;

— дата создания;

— время создания;

— атрибуты файла, которые определяют степень доступа к нему: R (Read only) — только для чтения, H (Hidden) — скрытый, S (System) — системный файл, A (Archive) — архивированный файл.

Понятие файловой системы. **Файловая система** — это функциональная часть операционной системы, обеспечивающая хранение данных на дисках и доступ к ним.

Принцип организации файловой системы в семействах операционных систем MS-DOS и Windows — табличный. Поверхность диска рассматривается как трехмерная матрица, измерениями которой являются номера поверхности, цилиндра и сектора. Данные о том, в каком месте диска записан тот или иной файл, хранятся в системной области диска в специальных таблицах размещения файлов (FAT-таблицах).

Файловая система определяет способы организации и средства обслуживания *файловой структуры,* преобразуя FAT-таблицы в иерархическую структуру для обеспечения быстрого и удобного доступа к данным, простого и понятного пользователю способа задания адреса данных. Операции, выполняемые операционной системой по обслуживанию файловой структуры:

— создание файлов и присвоение им имен;

— создание каталогов (папок) и присвоение им имен;

— переименование файлов и каталогов (папок);

— копирование и перемещение файлов и каталогов (папок);

— удаление файлов и каталогов (папок);

— навигация по файловой структуре с целью доступа к заданному файлу, каталогу (папке);

— управление атрибутами файлов.

Для обеспечения удобного доступа к файлам файловая система позволяет объединять их в каталоги (папки).

Каталогом называется специальный файл, в котором регистрируются другие файлы и каталоги. Если файл зарегистрирован в каталоге, это означает, что в последнем содержатся вся характеризующая файл информация и сведения о том, в каком месте диска файл расположен. Сам же файл хранится как последовательность байт без каких-либо дополнительных справочных сведений.

Правила именования каталогов совпадают с правилами именования файлов, однако расширения практически не используются. Каталоги низких уровней вкладываются в каталоги более высоких уровней и являются для них вложенными. Верхним уровнем вложенности иерархической структуры является *корневой каталог.*

На каждом диске всегда имеется единственный корневой каталог (он именуется символом «\»), в который могут входить другие каталоги и файлы. Корневой каталог создается при форматировании (разметке) диска, хранится во вполне определенной области дисковой памяти, имеет ограниченный размер и не может быть удален никакими средствами. Пользователь не имеет возможности что-либо сделать с корневым каталогом, за исключением помещения в него файлов и других каталогов, а также удаления их из него.

Каждый каталог хранит свою файловую структуру, которая формируется по следующим правилам:

— каталог или файл может входить только в один каталог;

— допускается вхождение в различные каталоги каталогов и файлов с одинаковыми именами (но, конечно, не в один каталог);

— на порядок следования файлов и каталогов в каталоге никаких ограничений (за исключением корневого каталога системного диска) не накладывается;

— глубина вложенности каталогов не ограничивается.

В один каталог обычно объединяют группу файлов (каталогов), связанных между собой по какому-либо признаку, например файлы и каталоги одного владельца, функционально подобные файлы (каталоги), файлы, имеющие однотипное содержимое (тексты, исходные программы и т. п.).

С понятием файла и каталога в ОС связано понятие логического диска. **Логический диск** создается и управляется специальной программой, имеет уникальное имя в виде одной латинской буквы (например: C, D, E, F и т. д.), может быть реализован на жестком и гибком дисках, на CD-ROM, в оперативной памяти (электронный диск). На одном физическом диске может быть создано несколько логических дисков.

Различают два состояния логического диска — текущее и пассивное. *Текущий диск* — это диск, на котором пользователь работает в текущее машинное время, *пассивным* является диск, с которым в данный момент времени связь отсутствует. Каталог также может быть текущим и пассивным. Операционная система помнит текущий каталог на каждом логическом диске.

Различают еще одно состояние каталога — активное. *Активный* каталог определяется как текущий каталог текущего диска, т. е. с каталогом установлена связь в настоящий момент времени.

Способы обращения к файлу, **группе файлов**. Для обеспечения доступа к существующему файлу или определения места размещения файла в файловой структуре в общем случае требуется задать:

— имя привода, на котором установлен диск, содержащий искомый файл или предназначенный для размещения нового файла;

— путь к файлу по файловой структуре этого диска;

— составное имя файла (имя файла и расширение имени файла).

Данные сведения указываются в спецификации файла, которая имеет следующий синтаксис (представление, форму, структуру): *[имя носителя]: [\маршрут \] имя_файла.[расширение].*

Маршрут (путь) — это цепочка соподчиненных каталогов, которую надо пройти по иерархической структуре к каталогу, где зарегистрирован искомый файл. При задании пути имена каталогов записываются в порядке следования и отделяются друг от друга символом «\».

Здесь необязательные элементы заключены в квадратные скобки, так как операционная система хранит информацию о текущем диске в текущем каталоге.

В случае когда те или иные элементы отсутствуют, они восстанавливаются по следующим правилам:

— если привод не задан, то выбирается текущий привод;

— если маршрут начинается с символа (указан полный маршрут), то поиск каталога, где должен содержаться файл, осуществляется, начиная с корневого каталога диска на выбранном дисководе;

— если условие предыдущего пункта не выполняется, то поиск каталога, где должен содержаться файл, осуществляется, начиная с текущего каталога диска на выбранном дисководе;

— если маршрут не задан, то считается, что файл содержится в текущем каталоге диска на выбранном дисководе.

Достаточно часто бывает необходимо выполнить одни и те же действия над несколькими файлами, например *Скопировать, Переместить* или *Удалить*. Для работы с несколькими файлами одновременно операционная система позволяет с помощью шаблона имени файла объединять их в группу.

Шаблон имени файла — это специальная форма, в которой в полях имени и типа файла используются символы «*» или «?».

Символ «*» служит для замены любой последовательности символов. В шаблоне в поле имени и типе файла может быть использовано по одному символу «*».

Символ «?» служит для замены одного символа. В шаблоне может быть использовано несколько таких символов.

Рассмотрим на примере файловой структуры диска [C:] способы обращения к файлу, к группе файлов:

Пример 1. Условие: файл b.doc зарегистрирован в активном каталоге K5. Что необходимо указать для доступа к данному файлу?

В этом случае для доступа к файлу достаточно указать его имя: b.doc.

Пример 2. Условие: диск [C:] в данный момент времени пассивный.

а) Что необходимо указать для доступа к файлу с именем file.doc?

В этом случае для доступа к файлу следует указать его полную спецификацию: C:\K2\K4\file.doc.

б) Что необходимо указать для доступа ко всем файлам каталога с именем K5?

В этом случае для доступа к группе файлов указывают следующую спецификацию: C:\K1\K2\K5*.*.

Пример 3. Условие: каталог с именем K2 активный.

а) Что необходимо указать для доступа ко всем файлам каталога K4, имеющим расширение .doc?

В этом случае для доступа к группе файлов указывают следующую спецификацию: K4*.doc.

б) Что необходимо указать для доступа ко всем файлам каталога K4, у которых расширение начинается с символа d и содержит максимально возможное количество символов?

В этом случае для доступа к группе файлов указывают следующую спецификацию: K4*.d?.

в) Что необходимо указать для доступа к файлу с именем a.doc каталога K4?

В этом случае для доступа к файлу указывают следующую спецификацию: K4\a.doc.

Режимы работы операционной системы. Операционная система имеет два режима работы: пакетный и диалоговый. *Пакетный* режим состоит в том, что ОС автоматически исполняет заданную последовательность команд. В отличие от пакетного, при *диалоговом* режиме операционная система находится в состоянии ожидания команды пользователя; получив ее, приступает к исполнению, а исполнив — возвращает отклик и ждет очередной команды.

3.3.2. Графические пользовательские интерфейсы

Виды интерфейса пользователя. По типу пользовательского интерфейса различают символьные (линейные) и графические операционные системы.

Линейные ОС реализуют интерфейс командной строки. Основным устройством управления в данном случае является клавиатура. Команда набирается на клавиатуре и отображается на экране дисплея. Окончанием ввода команды служит нажатие клавиши <*Enter*>. Для работы с операционными системами, имеющими линейный интерфейс, необходимо овладеть командным языком данной среды, т. е. совокупностью команд, структура которых определяется синтаксисом этого языка.

Графические ОС реализуют интерфейс, основанный на взаимодействии активных и пассивных графических экранных элементов управления. Устройствами управления в данном случае являются *клавиатура* и *мышь*. *Активным* элементом управления является указатель мыши — графический объект, перемещение которого на экране синхронизировано с перемещением мыши. *Пассивные* элементы управления — это графические элементы управления приложений (экранные кнопки, значки, переключатели, флажки, раскрывающиеся списки, строки меню и т. д.).

Операционные системы семейства Windows (от англ. windows — окно) являются графическими операционными системами компьютеров платформы IBM PC. Системы Windows 95, Windows 98 в основном предназначены для управления автономным компьютером, но также поддерживают создание небольшой компьютерной сети, например в пределах одного учебного класса (локальная, или одноранговая, сеть), и имеют средства для интеграции компьютера во всемирную сеть Internet. Системы Windows 2000, Windows XP, Windows Vista являются мощными сетевыми операционными системами, поддерживающими управление глобальными сетями.

Основным понятием операционной системы Windows является *объект,* его свойства и действия, которые он может выполнить в зависимости от запроса.

Объектами Windows являются:

— файлы;

— каталоги, называемые в системной среде Windows папками;

— папки логических устройств компьютера (диски, принтер, модем и т. д.).

Понятие *«Рабочий стол»*. Стартовый экран Windows представляет собой системный объект, называемый ***Рабочим столом***. *Рабочий стол* — это графическая среда, на которой отображаются объекты Windows и элементы управления Windows.

На стандартном *Рабочем столе* расположены несколько экранных графических значков (рис. 3.6) и *Панель задач* — один из основных элементов управления. В ее центральной части располагаются кнопки приложений или документов, с которыми пользователь работает в текущем сеансе. В левой части находится кнопка вызова *Главного меню* Windows *(Пуск),* а в правой части — *Панель индикации.*

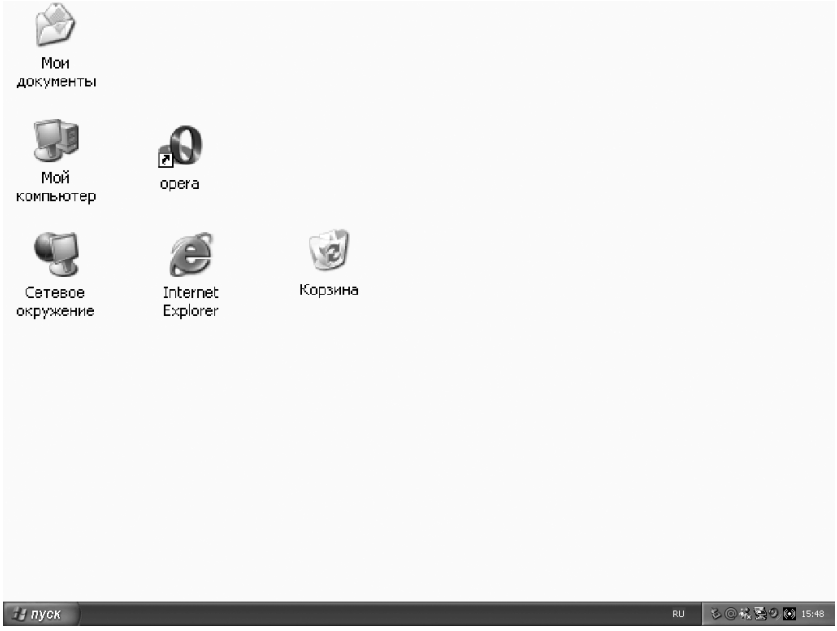


Рис. 3.6. Окно папки *Рабочий стол*

Каждый объект Windows имеет свой графический значок. Каждый значок имеет надпись, поясняющую его функциональное назначение или принадлежность к конкретной папке либо файлу. Значок, принадлежащий файлу, как правило, отражает приложение, в котором данный файл создан, указывает его тип.

Приемы управления. В системной среде Windows большинство операций можно выполнить многими различными способами, например через строку меню, через панель инструментов, через контекстное меню, через систему окон или программу *Проводник,* используя основные элементы управления.

При намерении что-либо сделать в системной среде Windows необходимо придерживаться определенной последовательности действий:

— выбрать (выделить) объект;

— из совокупности операций, которые можно выполнить над объектом, выбрать необходимую, например *Открыть, Скопировать, Отправить, Просмотреть свойства, Удалить* и т. д.

Основные приемы управления с помощью мыши. Основным устройством управления в Windows является мышь, так как большинство команд можно выполнить с ее помощью.

С мышью связан активный элемент управления — указатель мыши, который двигается по *Рабочему столу* синхронно перемещению мыши, его можно позиционировать на значках объектов или пассивных элементах управления.

Окна как объекты графического интерфейса. Основу графического интерфейса пользователя в системной среде Windows составляет организованная система окон и других графических объектов, при создании которой разработчики стремились к максимальной стандартизации всех элементов и приемов работы. Благодаря такой максимальной унификации структуры окон пользовательский интерфейс очень удобен.

Окно — это обрамленная прямоугольная область на экране монитора, в которой отображаются приложения, документ, сообщение. Окно будет активным, если с ним в данный момент работает пользователь. Все операции, которые мы выполняем, работая с компьютером под управлением операционной системы Windows, происходят или на *Рабочем столе*, или в каком-либо окне.

Система окон Мой компьютер. Windows относится к классу дисковых операционных систем, поэтому при построении файловой структуры сохраняются все основные правила подчиненности каталогов, а также размещения программ и данных в файлах. Однако в системной среде Windows понятие «каталог» заменяется понятием «объект-папка».

На верхнем уровне этой структуры находится единственный объект — *Рабочий стол*. На втором уровне располагаются объекты, расположенные на *Рабочем столе*. К таким объектам стандартно относятся системные папки *Мой компьютер* и *Корзина*. Эти папки нельзя удалить, переместить. Они, как и другие папки, служат хранилищами объектов Windows.

Папка *Мой компьютер* предоставляет доступ ко всем папкам и файлам в компьютере. В ней находятся системные папки дисковых устройств, папки *Принтер, Панель управления* и др. В папке *Мои документы* хранятся документы (файлы) пользователя. Папка *Корзина* предназначена для хранения удаленных файлов и папок, которые можно при необходимости восстановить.

Типовая структура окна. Перечислим стандартные элементы, обязательные для окон всех типов, на примере окна папки (рис. 3.7).

Строка заголовка. Указывает название объекта, которому принадлежит окно.

Системный значок. С его помощью вызываются команды изменения размеров окна и его перемещения.

Кнопки управления размером: Закрыть, Свернуть, Развернуть. Дублируют команды системного меню, служат для ускорения их вызова.

Строка меню. Содержит имена ниспадающих меню — группы команд, объединенных по функциональному признаку. Набор команд в строке меню определяется типом окна.

Панель инструментов. Содержит командные кнопки для выполнения наиболее часто встречающихся операций. В работе она удобнее, чем строка меню, но ограничена по количеству команд. В окнах современных приложений панель инструментов часто бывает настраиваемой. Пользователь сам может разместить на ней те командные кнопки, которыми он пользуется наиболее часто.

Адресная строка. Указывает путь доступа к текущей папке, что удобно для ориентации в файловой структуре. Позволяет выполнить быстрый переход к другим разделам файловой структуры с помощью раскрывающей кнопки на правом конце строки.

Рабочая область. Отображает значки объектов, хранящихся в папке, причем способом отображения можно управлять. В окнах приложений в рабочей области размещаются окна документов и рабочие панели.

Полосы прокрутки. Если количество объектов слишком велико (или размер окна слишком мал), по правому и нижнему краям рабочей области могут отображаться полосы прокрутки, с помощью которых можно «прокручивать» содержимое папки в рабочей области.

Строка состояния. Здесь выводится дополнительная, часто немаловажная, информация. Так, если среди объектов, представленных в окне, есть скрытые или системные, то они могут не отображаться при просмотре, но в строке состояния имеется специальная запись об их наличии. В окнах приложений

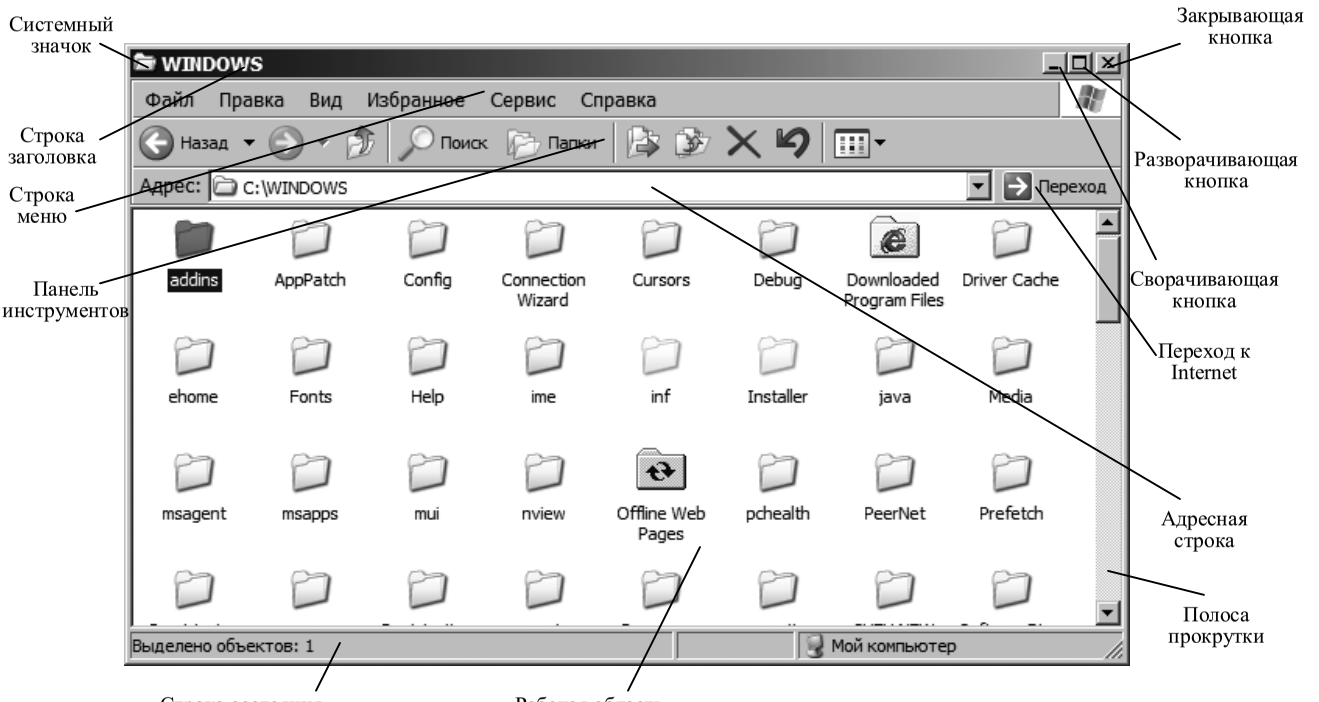


Рис. 3.7. Окно папки WIN2000

строка состояния содержит информацию о режимах работы приложения.

Программа Проводник. Служебная программа, предназначенная для навигации по файловой структуре компьютера и ее обслуживания; очень глубоко интегрирована в операционную систему Windows.

Окно программы *Проводник* представлено на рис. 3.8. По элементам управления оно похоже на окна папок. Основное отличие в том, что окно *Проводник* имеет не одну рабочую область, а две: левую панель, называемую **панелью папок**, и правую, называемую **панелью содержимого**.

Навигация по файловой структуре. Цель навигации состоит в обеспечении доступа к нужной папке и ее содержимому. Навигацию по файловой структуре выполняют на левой панели окна *Проводник,* где показана структура папок. Папки могут быть развернуты или свернуты, раскрыты или закрыты. Если папка имеет вложенные папки, то на левой панели рядом с папкой отображается узел, отмеченный значком «+». Щелчок на узле разворачивает папку, при этом значок узла меняется на «–». Таким же образом папки сворачиваются.

Чтобы раскрыть папку, надо щелкнуть левой кнопкой мыши на ее значке. Содержимое раскрытой папки отображается на правой панели. Одна из папок на левой панели раскрыта всегда. Закрыть папку щелчком на ее значке невозможно — она закроется автоматически при раскрытии любой другой папки.

Запуск программ и открытие документов. Эта операция выполняется двойным щелчком (левой кнопкой мыши) на значке программы или документа на правой панели *Проводника*. Если нужный объект на правой панели не показан, надо выполнить навигацию на левой панели и найти папку, в которой он находится.

Создание папок. Чтобы создать новую папку, сначала на левой панели окна *Проводник* надо раскрыть папку, внутри которой будет создана эта новая папка. Затем следует перейти на правую панель, щелкнуть правой кнопкой мыши на свободном от значков месте и выбрать в контекстном меню пункт *Создать → Папку*. На правой панели появится значок папки с названием *Новая папка*. После того как название выделено, его можно редактировать. Когда папка будет создана, она войдет в состав файловой структуры, отображаемой на левой панели.

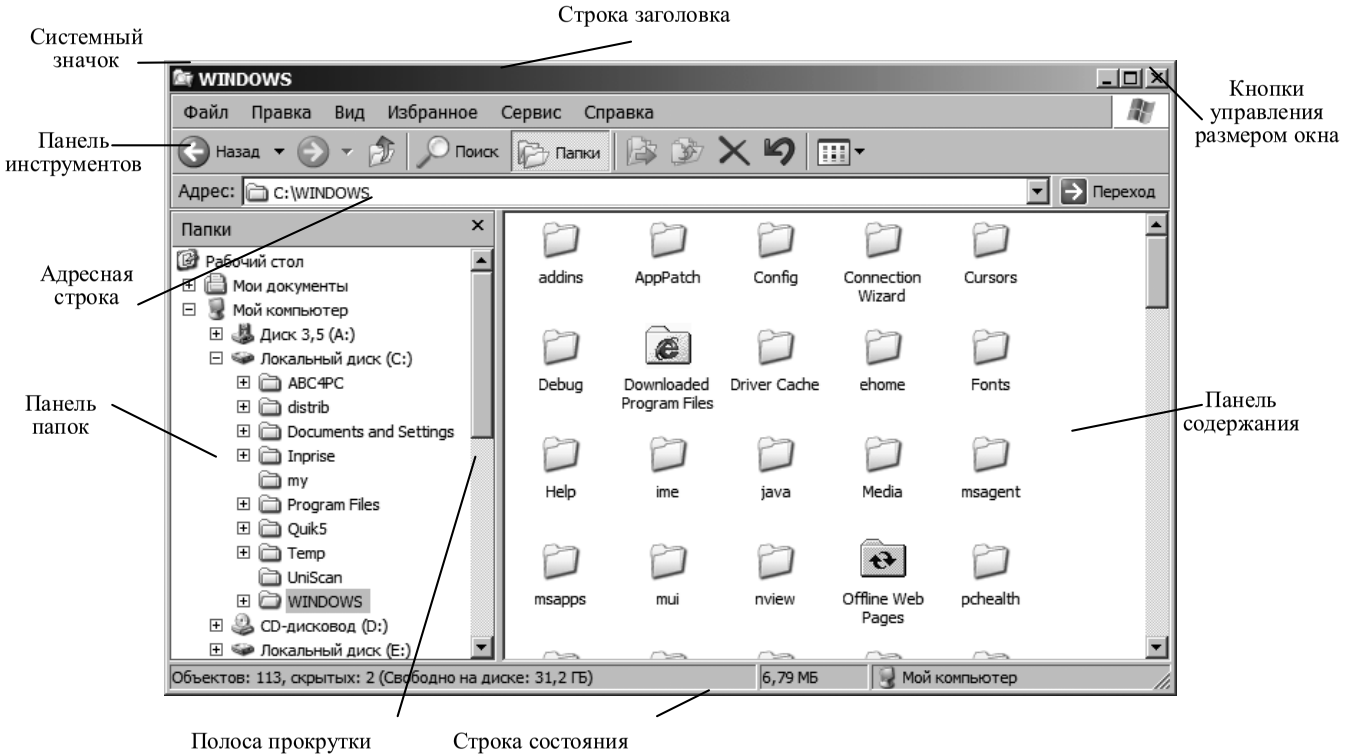


Рис. 3.8. Окно программы *Проводник*

Копирование и перемещение файлов и папок. Папка, из которой копируют, называется **источником**, папка, в которую копируют, — **приемником**. Копирование выполняют методом перетаскивания значка объекта с правой панели окна *Проводник* на левую.

Удаление файлов и папок. Работа начинается с навигации. На левой панели открывают папку, содержащую удаляемый объект, а на правой панели выделяют для удаления нужный объект (или группу объектов).

Создание ярлыков объектов. **Ярлыком объекта** в Windows называется указатель на объект. В отличие от ярлыка, значок — это лишь графическое изображение объекта. Ярлыки объектов можно создавать двумя способами: методом перетаскивания (вручную) или с помощью специальной *программы-мастера* (автоматически). ***Мастерами*** в системе Windows называются специальные программы, работающие в режиме диалога с пользователем. Диалог строится по принципу «запрос — ответ».

Буфер обмена. В системной среде Windows обмен данными можно произвести через *систему окон* или посредством программы *Проводник,* используя основные приемы управления мышью (перетаскивание, специальное перетаскивание), а также через общесистемные приемы, которые используют буфер обмена для работы с объектами. Эти приемы работают во всех приложениях Windows. Через буфер обмена можно переносить из одного документа в другой фрагменты текстов, иллюстрации, звукозаписи, видеофрагменты, файлы, папки.

Буфер обмена — это специальная область памяти, которая предназначена для временного хранения переносимого, копируемого или удаляемого объекта.

Групповое выделение объектов. Часто возникает ситуация, когда надо работать не с одним, а с группой объектов. Поэтому необходимо выделить группу объектов. Для выделения группы объектов используют следующие приемы:

— чтобы выделить произвольную группу объектов, надо, удерживая нажатой клавишу *<Ctrl>,* последовательно щелкать левой кнопкой мыши на каждом из нужных объектов. Выделение объектов при нажатой клавише *<Ctrl>* действует как переключатель, т. е. повторный щелчок;

— если выделяемые объекты расположены подряд, то следует, удерживая нажатой клавишу *<Shift>,* щелкнуть на последнем объекте выделяемой группы; все промежуточные объекты выделятся автоматически.

Другие графические пользовательские интерфейсы. Технологии общения с ПК зависят от *интерфейса*. Современные ОС поддерживают командный, WIMP- и SILK-интерфейсы. Ставится вопрос о создании так называемого общественного интерфейса (social interface).

Командный интерфейс означает выдачу на экран системного приглашения для ввода команды. Например, в MS-DOS это приглашение выглядит как C:\>, в Unix — как $.

WIMP-интерфейс (от англ. Windows — окна, Image — образ, Menu — меню, Pointer — указатель) является графическим, т. е. на экране высвечивается окно, содержащее образы программ и меню действий. Для выбора одного из них используется указатель.

В случае *SILK-интерфейса* (от англ. Speech — речь, Image — образ, Language — язык и Knowledge — знание) на экране по речевой команде происходит перемещение от одних поисковых образов к другим.

Предполагается, что при использовании *общественного интерфейса* не нужно будет разбираться в меню. Экранные образы однозначно укажут дальнейший путь перемещения от одних поисковых образов к другим по смысловым семантическим связям.

Пользовательским интерфейсом называется набор приемов взаимодействия пользователя с приложением. Пользовательский интерфейс включает общение пользователя с приложением и язык общения.

Рассмотрим кратко основные положения стандарта пользовательского интерфейса. Пользовательский интерфейс зависит от интерфейса, обеспечиваемого ОС. Свойствами интерфейса являются конкретность и наглядность. Графическая система Windows очень удобна, а богатство ее возможностей сделало ее оптимальной системой для повседневной работы. Приложения, написанные по Windows, используют тот же интерфейс, поэтому его единообразие сводит к минимуму процесс обучения работе с любым приложением Windows.

Разработка пользовательского интерфейса состоит из проектирования панелей и диалога.

Панель приложения разделена на три части: меню действий, тело панели и область функциональных клавиш.

Меню действий обеспечивает наглядность действий, которые могут быть запрошены пользователем установкой курсора, функциональной клавишей или каким-то другим простым способом. На цветном экране меню действий обычно имеет другой цвет по отношению к цвету панели. На монохромном экране используется сплошная линия для его отделения. Меню действий содержит объекты, состоящие из одного или нескольких слов. Два последних из них резервируются для действий *Выход* и *Справка*. Размещаются объекты слева направо по мере убывания частоты их использования.

Тело панели содержит следующие элементы: разделители областей, идентификатор панели, заголовок панели, инструкцию, заголовок столбца и группы, заголовок поля, указатель протяжки, область сообщений, область команд, поле ввода, поле выбора.

Область функциональных клавиш — необязательная часть, показывающая соответствие клавиш и действий, которые выполняются при их нажатии.

Для указания текущей позиции на области функциональных клавиш принят принцип «объект — действие». Этот принцип разрешает пользователю сначала выбрать объект, а затем произвести действие с ним, что минимизирует число режимов, упрощает и ускоряет обучение работе с приложениями. Если панель располагается в отдельной ограниченной части экрана, то она называется **окном**, которое может быть первичным или вторичным. В *первичном окне* диалог начинается, и если в приложении не нужно создавать другие окна, то первичным окном считается весь экран. Оно может содержать столько панелей, сколько нужно для ведения диалога. *Вторичные окна* вызываются из первичных. В них пользователь ведет диалог параллельно с первичным окном. Часто вторичные окна используются для подсказки. Первичные и вторичные окна имеют заголовок в верхней части окна. Пользователь может переключаться из первичного окна во вторичное и наоборот. Существует также понятие *всплывающих окон,* которые позволяют расширить диа-

лог пользователя с приложением, ведущийся из первичного или вторичного окна. В основном всплывающие окна используются для передачи сообщений или подсказки.

Когда пользователь и ЭВМ обмениваются сообщениями, диалог движется по одному из путей приложения, т. е. пользователь движется по приложению, выполняя конкретные действия. При этом действие не обязательно требует от приложения обработки информации. Диалоговые действия также контролируют информацию, которую набирает пользователь. Если он перешел к другой панели и его действия могут привести к потере информации, рекомендуется требовать подтверждение о том, следует ли ее сохранить. При этом пользователю предоставляется шанс сохранить информацию, отменить последний запрос, вернуться на один шаг назад.

Путь, по которому движется диалог, называется **навигацией**. Он может быть изображен в виде сети или графа, где узлы — действия, а дуги — переходы. Диалог состоит из двух частей: *запросов на обработку информации* и *навигации по приложению.* Часть запросов на обработку и навигацию является унифицированной. Унифицированные действия диалога — это действия, имеющие одинаковый смысл во всех приложениях. Некоторые унифицированные действия могут быть запрошены из ниспадающего меню, посредством действия *Команда,* функциональной клавишей. К унифицированным действиям диалога относятся: *Отказ, Команда, Ввод, Вывод, Подсказка, Регенерация, Извлечение, Идентификатор, Клавиши, Справка.*

Тема 3.4

ИНСТАЛЛЯЦИЯ ПРОГРАММ. КОМПЬЮТЕРНЫЕ ВИРУСЫ И АНТИВИРУСНЫЕ ПРОГРАММЫ

3.4.1. Инсталляция программ

Понятие «инсталляция» (от англ. installation — установка) означает процесс установки программного или аппаратного обеспечения на конкретную ЭВМ для конкретного пользователя. Инсталляция необходима, так как программные продукты

разрабатываются для работы на ЭВМ различных конфигураций для удовлетворения различных потребностей пользователя и поставляются в виде набора дискет, компакт-дисков и других устройств памяти. Инсталляция выполняется с помощью специальной установочной программы, поставляемой разработчиком. *Установочная программа* (от англ. installation program) проверяет наличие свободной памяти, прежних версий программы, устанавливает дополнительные программы, которые необходимы для запуска устанавливаемой программы. При подключении устройств с помощью установочной программы осуществляются настройка системных параметров, установка драйверов и тестирование. Установочная программа выполняется один раз (если, конечно, она завершается успешно), после чего можно работать с новой программой или устройством.

Например, при инсталляции ОС определяется, какие драйверы необходимы для конкретного дисплея, принтера; какое количество файлов может работать в системе; количество буферов и пр. **Драйвер** (от англ. driver — управляющая программа) — это программа, обеспечивающая связь между ОС и периферийным устройством и регулирующая поток данных, проходящих через устройство. Драйвер можно считать частью ОС. При подключении к ЭВМ нового устройства необходимо иметь драйвер, обеспечивающий его работу.

Буфер (от англ. buffer) — это:

1) дополнительная память для временного хранения данных. Буфер предназначен для компенсации более низкой скорости работы выходного устройства по сравнению со скоростями работы процессора и оперативной памяти;

2) часть оперативной памяти, используемая системой для временного хранения данных при выполнении операций копирования и переноса. Например, буфер диска хранит данные, считываемые с диска, и посылает их затем в ОП, и наоборот.

3.4.2. Компьютерные вирусы

Большую опасность для ЭВМ представляют компьютерные вирусы. **Компьютерный вирус** — это специально написанная небольшая по размерам программа, которая может приписывать

себя к другим программам (т. е. заражать их), а также выполнять различные нежелательные действия. Программа, внутри которой находится компьютерный вирус, называется **зараженной**. Когда такая программа начинает работу, то сначала управление получает вирус, который находит и заражает другие программы, а также выполняет ряд вредных действий, в частности засоряет активную память, портит файлы и т. д.

Для маскировки вируса его действия по заражению других программ и нанесению вреда могут выполняться не всегда, а при каких-либо условиях. После того как вирус выполнит нужные ему действия, он передает управление той программе, в которой он находится, и она работает как обычно, т. е. внешне работа зараженной программы какое-то время не отличается от работы незараженной.

Вирус может действовать достаточно быстро и без выдачи сообщений, поэтому пользователь часто не замечает, что компьютер работает несколько странно. Однако по прошествии какого-то времени на компьютере может происходить следующее:

— некоторые программы перестают работать или работают неправильно;

— на экран выводятся посторонние сообщения, символы, рисунки и т. д.;

— работа на компьютере существенно замедляется;

— некоторые файлы оказываются испорченными и т. д.

Многие вирусы устроены так, что при запуске зараженной программы они остаются постоянно (точнее, до перезагрузки ОС) в памяти компьютера и время от времени заражают другие программы. Кроме того, зараженные программы с данного компьютера могут быть перенесены с помощью дискет или по сети на другие компьютеры.

Среди всех вирусов можно выделить следующие группы:

— *загрузочные* (от англ. boot) *вирусы* заражают программу начальной загрузки компьютера, хранящуюся в загрузочном секторе диска и запускающуюся при загрузке компьютера;

— *файловые вирусы* в простейшем случае заражают исполняемые файлы, но могут распространяться и через файлы документов (системы Word for Windows) и даже вообще не модифицировать файлы, а лишь иметь к ним какое-либо отношение;

— *загрузочно-файловые вирусы* имеют признаки как загрузочных, так и файловых вирусов;

— *драйверные вирусы* заражают драйверы устройств компьютера или запускают себя путем включения в файл конфигурации дополнительных строк;

— *сетевые вирусы* распространяются в сетях, объединяющих сотни и тысячи компьютеров.

По способу заражения все эти вирусы подразделяют на резидентные и нерезидентные. И каждый вирус имеет, конечно, особенности в своем алгоритме.

Создание компьютерных вирусов с юридической точки зрения можно квалифицировать как преступление.

Но есть и хорошие новости: предприняв определенные меры предосторожности, описанные в специальной литературе, можно избежать последствий вирусной атаки, защитить свои программы и данные. Если заражение все же произошло, следует прибегнуть к помощи антивирусных программ.

3.4.3. Антивирусные программы и защита информации

Перед инсталляцией программ обязательно осуществляется их проверка на антивирус.

Антивирусная программа (от англ. antivirus program) — это программа, которая производит поиск и уничтожение вирусов. Антивирусные программы (или программные комплексы) также помогают восстановить испорченные файлы.

Известные в настоящее время антивирусные программы можно подразделить на несколько типов.

«Детекторы». Их назначение — обнаружить вирусы. Программы — «детекторы» вирусов могут сравнивать загрузочные сектора дисков с известными загрузочными секторами, формируемыми операционными системами различных версий, и таким образом обнаруживать загрузочные вирусы или выполнять сканирование файлов на магнитных дисках с целью обнаружения сигнатур известных вирусов. Такие программы в настоящее время в чистом виде редки.

«Фаги», или ***программы-«доктора»***. «Фаг» (от греч. phágos — пожиратель) — это программа, которая способна не только об-

наружить вирус, но и уничтожить его, т. е. она удалит его код из зараженных программ и восстановит их работоспособность. Известным в России «фагом» является Aidstest, разработанный Д. Лозинским (http://antivir.ru). Программа создана для детектирования и удаления самых разнообразных вирусов, в том числе и неизвестных.

«Ревизоры». Программы-«ревизоры» относятся к самым надежным средствам защиты от вирусов и должны входить в арсенал каждого пользователя. «Ревизоры» запоминают исходное состояние программ, каталогов и системных областей диска тогда, когда компьютер не заражен вирусом, а затем периодически сравнивают текущее состояние с исходным. Наиболее известна в России программа-«ревизор» ADinf, разработанная Д. Мостовым.

«Сторожа». «Сторож» — резидентная программа, постоянно находящаяся в памяти компьютера, контролирующая операции, связанные с попыткой коррекции файлов с расширением .COM и .EXE, с изменением атрибутов файлов, их записи в загрузочные сектора диска, прямой записи на диск по абсолютному адресу и т. д. «Сторож» предупреждает пользователя об этих операциях, но не лечит зараженные программы.

«Вакцины». «Вакцины», или «иммунизаторы», — это резидентные программы, предотвращающие заражение файлов. «Вакцины» применяются, если отсутствуют программы-«доктора», «лечащие» этот вирус. «Вакцинация» возможна только для известных вирусов. В настоящее время программы-«вакцины» имеют ограниченное применение.

Dr.Web для Windows. Эта программа представляет собой классический «полифаг» и предназначена для использования в 32-разрядных операционных системах семейства Windows. Она производит сканирование файлов и системных областей дисков компьютера на наличие в них компьютерных вирусов и при нахождении последних производит их лечение.

Doctor Web реализует эвристический метод поиска, который заключается в обнаружении фрагментов программ, типичных для компьютерных вирусов. Эвристический анализатор используется «полифагами» для обнаружения вирусов, не входящих в базу данных «полифага». Эффективность эвристического анализатора определяется двумя параметрами: процен-

том обнаруженных вирусов и процентом ложных срабатываний (подозрение на вирусы в файлах, в которых их нет). Doctor Web может находить и обезвреживать полиморфные вирусы (не имеющие определенной сигнатуры), проверять файлы, находящие в архивах.

Программа ***AntiViral Tookit Pro (AVP)***. AVP — новый шаг в борьбе с компьютерными вирусами. Это 32-разрядное приложение, оптимизированное для работы в среде Microsoft Windows, использует все ее возможности и предоставляет пользователю максимум сервиса — возможности обновления антивирусных баз через Internet (в том числе и автоматически), задания параметров автоматического сканирования и «лечения» зараженных файлов. Обновления на сайте AVP появляются практически еженедельно, а база данных включает описание более 40 тыс. вирусов. AVP состоит из нескольких важных модулей:

— *AVP-Сканер* — проверяет и «лечит» оперативную память (DOS, XMS, EMS); файлы, включая архивные и упакованные, и документы, созданные в формате Microsoft Office; системные сектора, содержащие Master Boot Record; загрузочный сектор (Boot-сектор) и таблицу разбиения диска (Partition Table);

— *AVP-Monitor* — резидентный модуль, постоянно находящийся в памяти компьютера и отслеживающий все файловые операции в системе. Обнаруживает и удаляет вирус до момента реального заражения системы в целом;

— *AVP-Inspector* — модуль, осуществляющий контроль изменений размеров файлов, позволяет отлавливать известные вирусы. Внедряясь в файл, вирус неизбежно увеличивает его объем, вызывает изменение его размера и тем самым выдает себя.

Практика функционирования систем обработки информации и вычислительных систем доказала, что существует достаточно много возможных направлений утечки информации и путей несанкционированного доступа в системах и сетях. В их числе:

— чтение остаточной информации в памяти системы после выполнения санкционированных запросов;

— копирование носителей и файлов информации с преодолением мер зашиты;

— маскировка под зарегистрированного пользователя;

— использование программных «ловушек»;

— использование недостатков операционной системы;

— незаконное подключение к аппаратуре и линиям связи;

— злоумышленный вывод из строя механизмов защиты;

— внедрение и использование компьютерных вирусов.

Обеспечение безопасности информации в вычислительных сетях и в автономно работающих ПЭВМ достигается комплексом организационных и организационно-технических мер, технических и программных средств.

Организационные меры защиты информации:

— ограничение доступа в помещения, в которых происходят подготовка и обработка информации;

— обеспечение доступа к обработке и передаче конфиденциальной информации только проверенных должностных лиц;

— хранение магнитных носителей и регистрационных журналов в закрытых для доступа посторонних лиц сейфах;

— исключение просмотра посторонними лицами содержания обрабатываемых материалов через дисплей, принтер и т. д.;

— использование криптографических кодов при передаче по каналам связи ценной информации;

— уничтожение красящих лент, бумаги и иных материалов, содержащих фрагменты ценной информации.

Организационно-технические меры защиты информации:

— подключение оборудования, обрабатывающего ценную информацию, к питанию от независимого источника или через специальные фильтры;

— установка на дверях помещений кодовых замков;

— использование для отображения информации при вводе-выводе жидкокристаллических или плазменных дисплеев, а для получения твердых копий — струйных принтеров и термопринтеров, поскольку дисплей дает такое высокочастотное электромагнитное излучение, что изображение с его экрана можно принимать на расстоянии нескольких сотен километров;

— уничтожение информации, хранящейся в ОЗУ и на «винчестере», при списании или отправке ПЭВМ в ремонт;

— установка клавиатуры и принтеров на мягкие прокладки с целью снизить возможность снятия информации акустическим способом;

— ограничение электромагнитного излучения путем экранирования помещений, где происходит обработка информации, листами из металла или специальной пластмассы.

Технические средства защиты информации — это системы охраны территорий и помещений с помощью экранирования машинных залов и контрольно-пропускные системы. Защита информации с помощью технических средств предусматривает:

— контроль доступа к различным уровням памяти компьютера;

— блокировку данных и ввод ключей;

— выделение контрольных бит для записей с целью идентификации и др.

Архитектура программных средств защиты информации включает:

— контроль безопасности, в том числе контроль регистрации вхождения в систему, фиксацию в системном журнале, контроль действий пользователя;

— реакцию, в том числе звуковую, на нарушение системы защиты контроля доступа к ресурсам сети;

— контроль мандатов доступа;

— формальный контроль защищенности операционных систем (базовой операционной и сетевой);

— контроль алгоритмов защиты;

— проверку и подтверждение правильности функционирования технического и программного обеспечения.

К *механизмам* обеспечения безопасности относятся:

— идентификация пользователей;

— шифрование данных;

— электронная подпись;

— управление маршрутизацией и др.

Идентификация пользователей позволяет устанавливать конкретного пользователя, работающего за терминалом и принимающего или отправляющего сообщения.

Право доступа к определенным вычислительным и информационным ресурсам, программам и наборам данных, а также к вычислительным сетям в целом предоставляется ограниченному контингенту лиц, и система должна распознавать пользователей, работающих за терминалами. Идентификация пользователей чаще всего производится с помощью паролей.

Пароль — совокупность символов, известных подключенному к сети абоненту, — вводится в начале сеанса взаимодейст-

вия с сетью, а иногда и в конце сеанса (в особо ответственных случаях пароль выхода из сети может отличаться от входного). Система может предусматривать ввод пароля для подтверждения правомочия пользователя через определенные интервалы времени. Вычислительная система определяет подлинность пароля и тем самым — пользователя.

Для защиты средств идентификации пользователей от неправомочного применения пароли передаются и сравниваются в зашифрованном виде, таблицы паролей также хранятся в зашифрованном виде, что исключает возможность их прочтения без знания ключа.

Для идентификации пользователей могут применяться и карточки с магнитным покрытием, на которых записывается персональный идентификатор пользователя, или карточки со встроенным чипом. Чтобы уменьшить риск злоупотреблений, при вводе карточек также предусматривается какой-либо способ идентификации пользователя, например короткий пароль.

Наиболее надежным, хотя и наиболее сложным является способ идентификации пользователя на основе анализа его индивидуальных параметров: отпечатков пальцев, рисунков линий ладони, радужной оболочки глаз и др.

Шифрование данных — это обеспечение секретности методами криптографии, т. е. методами преобразования данных из общепринятой формы в кодированную (шифрование) и обратного преобразования (дешифрования) на основе правил, известных только взаимодействующим абонентам сети. Криптография применяется для защиты передаваемых данных, а также информации, хранимой в базах данных, на магнитных и оптических дисках и т. д.

К криптографическим средствам предъявляется требование сохранения секретности, даже когда известна сущность алгоритмов шифрования-дешифрования. Секретность обеспечивается введением в алгоритмы специальных ключей (кодов). Зашифрованный текст превращается в исходный только в том случае, когда в процессе шифрования и дешифрования используется один и тот же ключ. Область значений ключа выбирается столь большой, что практически невозможно его определить путем простого перебора.

Тема 3.5

ИСТОРИЯ И ПЕРСПЕКТИВЫ РАЗВИТИЯ ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ И ПЕРСОНАЛЬНЫХ КОМПЬЮТЕРОВ

3.5.1. История развития вычислительной техники

Первые проекты электронных вычислительных машин появились в конце 1930-х — начале 1940-х гг. Технические предпосылки для этого уже были созданы. В 1904 г. был изобретен первый ламповый диод, а в 1906 г. — первый триод (соответственно двух- и трехэлектродные электронные лампы); в 1914 г. было изобретено электронное реле (ламповый триггер). Триггерные схемы стали широко применяться в электронике для переключения и релейной коммутации.

Благодаря развитию вычислительной техники в середине 1930-х гг. стало возможным создание программно-управляемых вычислительных машин, а построение ВМ на электронных схемах открывало широкие перспективы, связанные с увеличением их надежности и быстродействия.

ЭВМ появились, когда возникла острая необходимость в проведении трудоемких и точных расчетов. Уровень прогресса в таких областях науки и техники, как, например, атомная энергетика и аэрокосмические исследования, во многом зависел от возможности выполнения сложных расчетов, которые нельзя было осуществить на электромеханических счетных машинах. Требовался переход к вычислительным машинам, работающим с большей производительностью.

Во все времена людям необходимо было считать. Сначала для счета использовали пальцы рук или камешки. Но даже простые арифметические операции с большими числами трудны для мозга человека. Поэтому уже более пятнадцати веков назад в странах Средиземноморья был придуман простейший инструмент для счета — абак, представляющий собой набор костяшек, нанизанных на стержни. Этот прообраз современных счетов широко использовали купцы. Стержни абака в арифметическом смысле представляют собой десятичные разряды. Каждая костяшка на первом стержне имеет достоинство 1, на втором — 10, на третьем — 100 и т. д.

В России так называемые русские счеты — в их основе лежит десятичная система счисления — появились в XVI в. До XVII в. счеты оставались практически единственным счетным инструментом.

В 1614 г математик Дж. Непер изобрел логарифмы. Открытие Непера состояло в том, что таким способом можно выразить любое число и что сумма логарифмов двух любых чисел равна логарифму произведения этих чисел. Это дало возможность свести действие умножения к более простому действию сложения. Непер создал таблицы логарифмов. Чтобы перемножить два числа, нужно найти в этой таблице их логарифмы, сложить их и отыскать число, соответствующее этой сумме, в обратной таблице — таблице антилогарифмов. На основе этих таблиц в 1654 г. Р. Биссакар и в 1657 г. — независимо от него — С. Партридж разработали прямоугольную логарифмическую линейку — основной счетный прибор инженера до середины XX в.

В 1642 г. Блез Паскаль изобрел механическую суммирующую машину, использующую десятичную систему счисления. Каждый десятичный разряд представляло колесико с десятью зубцами, обозначавшими цифры от 0 до 9. Всего колесиков было восемь, т. е. машина Паскаля была восьмиразрядной.

В 1673 г. Готфрид Лейбниц построил механический арифмометр, выполнявший все четыре арифметических действия. В нем использовалась двоичная система счисления.

В 1804 г. Жозеф Жаккар создал ткацкую машину для выработки тканей с крупным узором. Этот узор программировался с помощью целой колоды перфокарт — прямоугольных карточек из картона. На них информация об узоре записывалась пробивкой отверстий (перфораций), расположенных в определенном порядке. При работе машины эти перфокарты «ощупывались» специальными штырями. Именно таким механическим способом с них считывалась информация для плетения запрограммированного узора ткани. Машина Жаккара явилась прообразом машин с программным управлением, созданных в XX в.

В 1820 г. Тома де Кольмар разработал первый коммерческий арифмометр, способный умножать и делить. Арифмометры получили широкое распространение при выполнении сложных расчетов.

В 1830 г. Чарльз Бэббидж попытался создать универсальную аналитическую машину, которая должна была выполнять вычисления без участия человека. Для этого в нее вводились программы, заранее записанные на перфокарты из плотной бумаги посредством отверстий, сделанных на них в определенном порядке. Принципы программирования для аналитической машины Бэббиджа разработала в 1843 г. Ада Лавлейс — дочь английского поэта Байрона.

Аналитическая машина должна была запоминать данные и промежуточные результаты вычислений, т. е. иметь память. Эта машина содержала три основные части: устройство для хранения чисел, набиравшихся с помощью зубчатых колес (память), устройство для операций над числами (арифметическое устройство) и устройство для операций над числами с помощью перфокарт (устройство программного управления). Работа по созданию аналитической машины не была завершена, но заложенные идеи помогли в XX в. построить первые компьютеры (напомним: в переводе с английского «компьютер» означает «вычислитель»).

В 1880 г. в России Вильгодт Однер изобрел механический арифмометр с зубчатыми колесами, а в 1890 г. начался его массовый выпуск. Под названием «Феликс» этот арифмометр выпускался вплоть до 1950 г.

В 1888 г. Герман Холлерит создал первую электромеханическую счетную машину — табулятор, в котором нанесенная на перфокарты информация расшифровывалась электрическим током. В 1890 г. изобретение Холлерита было впервые использовано во второй американской переписи населения. Работа, которую 500 сотрудников раньше выполняли целых семь лет, Холлерит с 43 помощниками на 43 табуляторах выполнили всего за один месяц.

В 1937 г. Джордж Стибиц из обыкновенных электромеханических реле создал двоичный сумматор — устройство, способное выполнять операцию сложения чисел в двоичном коде. И сегодня двоичный сумматор является одним из главных компонентов любого компьютера, основой его арифметического устройства.

В 1937—1942 гг. Джордж Атанасофф работал над моделью вычислительной машины на вакуумных электронных лампах.

В ней использовалась двоичная система счисления. Для ввода данных и вывода результатов вычислений применялись перфокарты. В 1942 г. работа над этой машиной была практически завершена, но из-за войны финансирование исследований прекратилось.

В 1937 г. Конрад Цузе сконструировал на основе электромеханических реле свою первую вычислительную машину — Z1. Исходные данные вводились в нее с помощью клавиатуры, а результат вычислений высвечивался на панели с множеством электрических лампочек. В 1938 г. К. Цузе создал усовершенствованную модель — Z2. Программы в нее вводились с помощью перфоленты. Перфоленту изготавливали, пробивая отверстия в использованной 35-миллиметровой фотопленке. В 1941 г. К. Цузе построил действующий компьютер Z3, а позднее — и Z4, основанный на двоичной системе счисления. Они использовались для расчетов при создании самолетов и ракет. В 1942 г. Конрад Цузе и Хельмут Шрайер задумали перевести Z3 с электромеханических реле на вакуумные электронные лампы. Такая машина должна была работать в тысячу раз быстрее, но создать ее не удалось — помешала война.

В 1943—1944 гг. на одном из предприятий в сотрудничестве с учеными Гарвардского университета во главе с Говардом Эйкеном была создана вычислительная машина «Марк-1». Весила она около 35 т. В «Марк-1» использовались идеи, заложенные Ч. Бэббиджем в его аналитической машине. В отличие от Стибица и Цузе, Эйкен не осознал преимуществ двоичной системы счисления и в своей машине использовал десятичную систему. Машина могла манипулировать числами длиной до 23 разрядов. Для перемножения двух таких чисел ей было необходимо затратить 4 с. В 1947 г. была создана машина «Марк-2», в которой использовалась уже двоичная система счисления. В этой машине операции сложения и вычитания занимали в среднем 0,125 с, а умножение — 0,25 с.

Электромеханические реле работали слишком медленно. Поэтому уже в 1943 г. американцы начали разработку вычислительной машины на основе электронных ламп. В 1946 г. Преспер Эккерт и Джон Мочли построили первую электронную цифровую вычислительную машину — ENIAC. Ее вес составлял 30 т, она занимала площадь 170 м^2. Вместо огромного числа элек-

тромеханических реле, в ENIAC работало 18 тыс. электронных ламп. Считала машина в двоичной системе и производила 5000 операций сложения или 300 операций умножения в секунду. На электронных лампах было построено не только арифметическое, но и запоминающее устройство. Числовые данные вводились с помощью перфокарт, а программы — с помощью штекеров и наборных полей, т. е. для каждой новой программы приходилось соединять тысячи контактов. Поэтому для подготовки к решению новой задачи требовалось несколько дней, хотя сама задача решалась за несколько минут. Это был один из основных недостатков машины.

3.5.2. Поколения ЭВМ

В истории развития ЭВМ выделяют пять этапов, соответствующих пяти поколениям ЭВМ.

Период машин ***первого поколения*** начинается с переходом к серийному производству ЭВМ в начале 1950-х гг. В них были реализованы основные принципы, предложенные Джоном фон Нейманом (см. параграф 3.1.3):

1. Принцип хранимой программы. Машина имеет память, в которой хранятся программа, данные и результаты промежуточных вычислений. Программа вводится в машину, так же как и данные, в виде двоичных кодов (а не штекерным методом, т. е. коммутацией проводов в определенной последовательности).

2. Адресный принцип. В команде указываются не сами числа, над которыми нужно выполнять арифметические действия, а адреса ячеек памяти, где эти числа находятся.

3. Автоматизм. После ввода программы и данных машина работает автоматически, выполняя предписания программы без вмешательства человека. Для этого машина запоминает адрес выполняемой команды, а каждая команда содержит указание об адресе следующей команды. Указание может быть одним из трех типов: неявным (перейти к команде, следующей по адресу за выполняемой), безусловным (перейти к команде по заданному адресу), условным (проверить заданное условие и в зависимости от его выполнения перейти к команде по тому или иному адресу).

4. Переадресация. Адреса ячеек памяти, указанные в команде, можно вычислять и преобразовывать как числа.

Структура ЭВМ, в которой реализованы принципы фон Неймана, впоследствии получила название структуры его имени (или классической). Все дальнейшее развитие ЭВМ шло двумя путями: совершенствование структуры фон Неймана и поиск новых структур.

Технической основой элементной базы процессоров первого поколения ЭВМ были электронные вакуумные лампы (ЭВЛ), а в качестве оперативных запоминающих устройств использовались электронно-лучевые трубки (ЭЛТ). Это были громоздкие машины, занимающие много места и потребляющие много электроэнергии. Они делали несколько тысяч операций в секунду и обладали памятью в несколько тысяч машинных слов. Такие машины предполагали монопольный режим использования, т. е. в распоряжении пользователя были все ресурсы машины и ее управление. Программист писал свою программу в машинных кодах и отлаживал ее за пультом машины, которая на время отладки была полностью в его распоряжении. При этом 90 % времени машина простаивала в ожидании команд, т. е. использование машинных ресурсов было малоэффективным из-за отсутствия развитой операционной системы. Использовались ЭВМ первого поколения в основном для научных расчетов.

Первой отечественной ЭВМ была МЭСМ (малая электронная счетная машина), разработанная в 1947—1951 гг. под руководством академика С.А. Лебедева. В 1952 г. была введена в эксплуатацию БЭСМ (большая электронная счетная машина), созданная также под его руководством. В 1955 г. начался выпуск малой ЭВМ «Урал-1» (руководитель проекта Б.И. Рамеев). Примером зарубежной серийной модели ЭВМ является IBM-701 (США).

Второе поколение ЭВМ (конец 1950-х — середина 1960-х гг.) называют транзисторно-ферритовым, так как транзисторы (твердые диоды и триоды) заменили электронные лампы в процессорах, а ферритовые (намагничиваемые) сердечники — электронно-лучевые трубки в оперативных запоминающих устройствах.

Применение транзисторов существенно повлияло на характеристики и структуру машин. Транзисторные схемы позволи-

ли на порядок повысить плотность монтажа электронной аппаратуры и существенно (на несколько порядков) снизить потребление электроэнергии. Срок службы транзисторов на два-три порядка превосходил срок службы электронных ламп. Скорость ЭВМ возросла до сотен тысяч операций в секунду, а память — до десятков тысяч машинных слов.

Создание долговременной памяти на магнитных дисках и лентах, а также возможность подключения к ЭВМ изменяемого состава внешних устройств существенно расширили функциональные возможности вычислительных машин.

В организации вычислительного процесса также были крупные достижения — это совмещение во времени вычислений и ввода-вывода информации, переход от монопольного режима использования ресурсов машины к пакетной обработке. Задания для ЭВМ (на перфокартах, магнитных лентах или дисках) собирались в пакет, который обрабатывался без перерыва между заданиями. Это позволило более экономно использовать ресурсы машины.

Были разработаны методы программирования в символических обозначениях, созданы первые алгоритмические языки и трансляторы с этих языков, библиотеки стандартных программ.

Наиболее широкое применение нашли отечественные ЭВМ БЭСМ-4, М-220, «Минск-32». Типичным представителем зарубежной ЭВМ второго поколения является IBM-7090.

Третье поколение ЭВМ (конец 1960-х — начало 1970-х гг.) характеризуется появлением в качестве элементной базы процессора интегральных полупроводниковых схем (вместо отдельных транзисторов), что привело к дальнейшему увеличению скорости до миллиона операций в секунду и памяти — до сотен тысяч слов.

ЭВМ третьего поколения также характеризуются крупнейшими сдвигами в архитектуре ЭВМ, их программном обеспечении, организации взаимодействия человека с машиной. Это прежде всего наличие развитой конфигурации внешних устройств (алфавитно-цифровые терминалы, графопостроители и т. п.) с использованием стандартных средств сопряжения и развитая операционная система, обеспечивающая работу в мультипрограммном режиме (несколько одновременно размещаемых

в оперативной памяти программ совместно используют ресурсы процессора). Метод использования ресурсов ЭВМ — режим разделения времени совместно с пакетной обработкой. Высокое быстродействие позволяет разбить на кванты время обслуживания пользователей, обрабатывать в течение кванта задание каждого и возвращаться к пользователю за такое малое время, что у него создается иллюзия, будто он один пользуется ресурсами машины.

Решающее значение в развитии вычислительной техники во всем мире сыграло создание семейства вычислительных машин на интегральных схемах с широким диапазоном вычислительной мощности и совместимых снизу вверх на уровне машинных языков, внешних устройств, модулей конструкции и системы элементов. Программная совместимость снизу вверх машин одного и того же семейства предполагает, что любая программа, выполнявшаяся на «младшей» машине, должна без всяких переделок выполняться на «старшей».

Широкое распространение получили также семейства мини-ЭВМ. Сущность их конструкторского решения состояла в такой минимизации аппаратуры центрального процессора, которая позволила на уровне технологии того времени создать универсальные ЭВМ, способные осуществлять управление в реальном масштабе времени, при котором темп выдачи управляющих воздействий на объект управления согласован со скоростью протекания процессов в этом объекте.

В нашей стране в период машин третьего поколения была создана Единая Система ЭВМ (ЕС ЭВМ), в основных чертах копирующая IBM-360 и IBM-370, а также серия мини-ЭВМ (СМ ЭВМ), ориентированная на зарубежные модели. Вклад отечественной науки в мировое развитие электронной вычислительной техники в этот период связан с промышленным внедрением многопроцессорной ЭВМ М-10.

В период машин третьего поколения произошел крупный сдвиг в области применения ЭВМ. Если раньше они использовались в основном для научно-технических расчетов, то в 1960—1970-е гг. первое место стала занимать обработка символьной информации, в основном экономической.

Машины серии ЕС ЭВМ имели универсальное назначение, а основной областью применения СМ ЭВМ была автоматиза-

ция технологических процессов, научных экспериментов, испытательных установок и проектно-конструкторских работ.

Переход к машинам ***четвертого поколения*** — ЭВМ на больших интегральных схемах (БИС) — происходил во второй половине 1970-х гг. и завершился приблизительно к 1980 г. Теперь на одном кристалле размером 1 $см^2$ стали размещаться сотни тысяч электронных элементов. Скорость и объем памяти возросли в десятки тысяч раз, по сравнению с машинами первого поколения, и составили примерно 10^9 оп/с и 10^7 слов соответственно.

Характерные особенности машин четвертого поколения — тесная связь аппаратной и программной реализаций в структуре машины, отход от принципа минимизации аппаратуры и поручение ей функций программы, что стало возможным благодаря относительно низкой стоимости БИС.

Развитие архитектуры ЭВМ в период машин четвертого поколения привело к появлению структур, в которых вычислительный процесс может протекать по нескольким ветвям параллельно, что приводит к увеличению производительности вычислительных машин. Идея параллелизма была технически реализована в многопроцессорных системах, состоящих из двух или более взаимосвязанных процессоров, работающих с общей памятью и управляемых общей операционной системой.

В результате возросшего быстродействия ЭВМ стало возможным расширить оперативную память за счет введения виртуальной памяти, основанной на страничном обмене информацией между внешней и основной памятью.

Наиболее крупным достижением, связанным с применением БИС, стало создание микропроцессоров, а затем на их основе — микроЭВМ. Если прежние поколения ЭВМ требовали для своего размещения специальных помещений, системы вентиляции, специального оборудования для электропитания, то требования, предъявляемые к эксплуатации микроЭВМ, ничем не отличаются от условий эксплуатации бытовых электроприборов. При этом они имеют достаточно высокую производительность, экономичны в эксплуатации и дешевы. МикроЭВМ используются в измерительных комплексах, системах числового программного управления, в управляющих системах различного назначения.

Дальнейшее развитие микроЭВМ привело к созданию персональных компьютеров, широкое распространение которых качалось с 1975 г., когда фирма IBM выпустила свой первый персональный компьютер IBM PC. Сейчас такие компьютеры (совместимые с IBM PC) составляют около 90 % всех производимых в мире ПК. В ПК реализован принцип открытой архитектуры, который означает, что по мере улучшения характеристик основных блоков ПК можно легко заменить устаревшие части, а модернизированный блок будет совместим с ранее используемым оборудованием. В числе других преимуществ ПК — развитые средства диалога, высокая надежность, удобство эксплуатации, наличие программного обеспечения, охватывающего практически все сферы человеческой деятельности.

Начали серийно производиться и суперЭВМ. Рост степени интеграции БИС стал технологической основой производительности ЭВМ. В нескольких серийных моделях была достигнута производительность свыше 1 млрд операций в секунду. К числу наиболее значительных разработок машин четвертого поколения относится ЭВМ «Крей-3» (США), спроектированная на основе принципиально новой технологии — замены кремниевого кристалла арсенидом галлия и имеющая производительность до 16 млрд операций в секунду. Примером отечественной суперЭВМ является многопроцессорный вычислительный комплекс «Эльбрус» с быстродействием до $1{,}2 \times 10^8$ оп/с.

С конца 1980-х гг. в истории развития вычислительной техники наступила пора ***пятого поколения*** ЭВМ. Технологические, конструкторские, структурные и архитектурные идеи машин пятого поколения принципиально отличаются от машин предшествующих поколений. Прежде всего, их структура и архитектура отличаются от фоннеймановской (классической). Высокая скорость выполнения арифметических вычислений дополняется высокими скоростями логического вывода. Даже скорость предполагается выражать в единицах логического вывода. Машина состоит из нескольких блоков. Блок общения обеспечивает интерфейс между пользователем и ЭВМ на естественном языке, и в будущем дисциплина «Программирование» как наука для пользователя перестанет быть актуальной. Важное место в структуре ЭВМ занимает блок *базы знаний*, в котором хранятся знания, накопленные человечеством в раз-

личных предметных областях; знания постоянно расширяются и пополняются. Следующий блок, называемый *решателем*, организует подготовку программы решения задачи на основании знаний, получаемых из базы знаний, и исходных данных, полученных из блока общения. Ядро вычислительной системы составляет ЭВМ высокой производительности.

В связи с появлением новой базовой структуры ЭВМ в машинах пятого поколения широко используются модели и средства, разработанные в области искусственного интеллекта.

3.5.3. Перспективы развития ПЭВМ

Перспективы развития персональных компьютеров во многом определяются функциональными возможностями, технико-эксплуатационными характеристиками и архитектурным построением микропроцессоров (МП).

В настоящее время крупнейшим мировым производителем МП является фирма Intel. Последние модели этой фирмы — МП Pentium применяются в мощных настольных ПЭВМ, рабочих станциях и многопроцессорных серверах.

В настоящее время нашли широкое применение *транспьютеры*. Они, как правило, используются в качестве сопроцессоров и рассчитаны на работу в параллельных системах с однотипными процессорными элементами и аппаратной поддержкой вычислительных процессоров. В состав системы команд транспьютеров входят команды управления процессами, поддержки инструкций, языков высокого уровня. Транспьютеры используют быстрые коммуникационные каналы, которые позволяют по одной магистрали передавать данные в процессор, а по другой (одновременно) — данные из него. Высокая производительность обеспечивается прежде всего за счет высокой скорости работы АЛУ и передачи операндов.

В современных МП широко применяются кэш- и виртуальная память, что по функциональным возможностям приближает ПЭВМ к большим ЭВМ. В ПЭВМ стали использовать многозадачный режим работы, динамическое распределение памяти, системы ее защиты.

На отечественном компьютерном рынке появилось немало разнообразных моделей ПЭВМ. Однако большинство из них

построены на МП фирмы Intel. Менее распространены модели ПЭВМ, созданные на базе МП фирмы Motorola.

В настоящее время тенденции в развитии производства ПЭВМ сводятся к насыщению рынка машинами следующих трех классов: профессиональными многопроцессорными ПЭВМ, приблизившимися по своим параметрам к большим ЭВМ; сравнительно недорогими ПЭВМ для массового потребителя; микроминиатюрными ПЭВМ (типа NOTEBOOK и HANDHELD). При выборе ПЭВМ следует обращать внимание прежде всего на скорость работы МП (на его тактовую частоту), которая во многом определяет диапазон применения компьютера. Важным фактором эффективности использовании ПЭВМ является емкость ОП, обеспечивающая возможность хранения набора программ, которые планируется выполнять на выбранной ПЭВМ. Емкость ОП для работы в среде Windows, например, должна быть 16 Мбайт и более.

Периферийные устройства необходимо выбирать по их технико-эксплуатационным параметрам, ориентируясь на классы задач, которые должны решаться на ПЭВМ, условия их эксплуатации, а также на удобства, предоставляемые пользователю. При этом следует учитывать возможность развития потребностей пользователя. Это особенно важно при выборе НЖМД, поскольку многие современные программные средства требуют для своего хранения сотни мегабайт памяти на НЖМД типа «винчестер».

В настоящее время появились мощные переносные ПК и серверы, TV PC — компьютерные телеприемники, Wallet PC — компьютерные бумажники, помещающиеся в кармане, Kiosk PC — устройства, заменяющие телефонный аппарат и предоставляющие широкий спектр услуг — от видеоконференции до пересылки денежных сумм. Ключом такого рода технологических достижений остается микропроцессор.

МП Pentium имеет суперскалярную архитектуру, два конвейера с отдельными исполнительными устройствами, встроенный сопроцессор с плавающей точкой. Локальная шина PCI (Peripheral Component Interconnect) предназначена для передачи данных между процессором и высокоскоростной периферийной ПЭВМ. Пропускная способность PCI достигает сотен мегабайт в секунду, причем возможно дальнейшее ее наращи-

вание — до гигабайт в секунду. Шина PCI позволяет использовать одни и те же высокоскоростные периферийные устройства в ПЭВМ с процессорами Intel, ALPHA или RISC. Использование шины PCI с процессором Pentium обеспечивает максимальную пропускную способность при работе с сетями, графикой, дисками и т. п. Фирма Intel рекомендует применять МП Pentium при решении задач моделирования, трехмерного проектирования, для создания серверов и многопроцессорных систем.

Прогресс в развитии МП обеспечивается использованием новых архитектурных решений, в частности транспьютерной и RISC-архитектуры конвейерного выполнения команд, применения сопроцессоров, параллельной обработки данных и т. п.

Рассмотрим особенности МП с архитектурой RISC (Reducted Instruction Set Computer — компьютер с сокращенной системой команд). В этих МП применяется сравнительно небольшой (сокращенный) набор наиболее часто употребляемых команд, определенный в результате статистического анализа большого числа программ. Для RISC-архитектуры характерны следующие особенности: все команды имеют одинаковый формат; большинство команд — трехадресные; большое количество внутренних регистров МП, что позволяет резко сократить число обращений к ОП, а следовательно, уменьшить время машинного цикла; конвейеризация выполнения команд; наличие кэш-памяти. Ограниченный набор команд сравнительно простой структуры дает возможность уменьшить количество аппаратуры.

При одной и той же тактовой частоте ПЭВМ RISC-архитектуры имеют производительность в два — четыре раза выше, чем ПЭВМ обычной архитектуры. В настоящее время ПЭВМ с RISC-архитектурой применяются в качестве графических рабочих станций, серверов локальных сетей, служат основой для создания современных управляющих, телекоммуникационных и банковских сетей. На отечественном рынке предлагаются модели новых систем RISC/6000 — семь моделей серверов и четыре модели рабочих станций на базе МП Power PC (фирма IBM). МП Power PC 601 работает с тактовой частотой от 50 до 200 МГц. Фирма App1e несколько лет назад выпустила модель Power Macintosh 8100/80 на базе процессора Power PC 601 RISC 80 МГц со встроенным сопроцессором и кэш-памятью 32 Мбайт, ОП

емкостью 16 Мбайт (с возможностью расширения до 264 Мбайт) и «винчестером» емкостью 1000 Мбайт.

В 2001 г. персональным компьютерам исполнилось 20 лет. Посмотрим, как они изменились за эти годы. Первые из них, оборудованные микропроцессором Intel, работали с тактовой частотой всего 4,77 МГц и имели оперативную память емкостью 16 Кбайт. Современные ПК, оборудованные микропроцессором Pentium 4, созданным в 2001 г., имеют тактовую частоту 2 ГГц, оперативную память емкостью 128—256 Мбайт и долговременную память («винчестер») объемом 40—80 Гбайт. Подобного прогресса не наблюдается ни в одной отрасли техники, кроме цифровой вычислительной. Если бы такими же темпами развивалось, скажем, самолетостроение, то современные самолеты весили бы не более килограмма и летали со скоростью света.

Миллионы компьютеров используются практически во всех отраслях экономики, промышленности, науки, техники, педагогики, медицины... Основные причины такого гигантского прогресса — в необычайно высоких темпах микроминиатюризации устройств цифровой электроники и успехах программирования, сделавших общение рядовых пользователей с персональным компьютером простым и удобным.

ПРАКТИЧЕСКИЕ РАБОТЫ

В практических работах рассматриваются:

— стандартные приемы управления объектами Windows;

— общепринятые способы выполнения операций по обслуживанию файловой структуры (навигация по файловой структуре; создание, копирование, перемещение, удаление объектов) в операционной среде Windows;

— стандартные способы работы с приложениями Windows, а также некоторые общепринятые элементы настройки пользовательского интерфейса.

Поэтому предложенные практические задания можно выполнить в любой операционной системе семейства Windows независимо от версии, установленной на компьютере. Однако надо учитывать, что могут не совпадать адреса объектов, так как файловая структура создается конкретным пользователем.

Перед началом работы рассмотрим алгоритмы выполнения наиболее часто используемых стандартных операций.

1. *Выделить объект* — навести указатель мыши на значок объекта и щелкнуть один раз левой кнопкой мыши.

2. *Выполнить команду из списка меню* — навести указатель мыши на команду и щелкнуть один раз левой кнопкой мыши.

3. *Открыть объект* — навести указатель мыши на значок объекта и щелкнуть два раза (двойной щелчок) левой кнопкой мыши.

4. *Зацепить* — навести указатель мыши на значок объекта, нажать левую (правую) кнопку мыши и не отпускать, пока не выполните необходимую операцию.

5. *Вызвать контекстное меню объекта* — навести указатель мыши на значок объекта и щелкнуть один раз правой кнопкой мыши.

6. *Вызвать контекстное меню Рабочего стола (Открытого окна)* — навести указатель мыши на свободную от значков поверхность *Рабочего стола* (*Открытого окна*) и щелкнуть один раз правой кнопкой мыши.

7. *Перетащить* — предварительно зацепив значок объекта, переместить его в нужном направлении.

8. *Активизировать экранные кнопки-вкладки* (пассивный элемент управления) — навести указатель мыши на экранную вкладку и щелкнуть один раз левой кнопкой мыши.

9. *Перемещение внутри окна папки от объекта к объекту* (поиск объекта внутри окна папки) — с помощью клавиши управления курсором или полосы горизонтальной (вертикальной) прокрутки перетаскивать в нужном направлении экранные кнопки полос.

Работа № 1. Объекты Windows

Цель работы:

— освоить работу с ОС Windows NT;

— научиться работать с окнами;

— освоить быстрый поиск объектов;

— научиться создавать папки, файлы, ярлыки;

— научиться удалять и восстанавливать удаленные объекты.

Задание 1: загрузка Windows; элементы *Рабочего стола* и работа с ними.

Методика выполнения задания:

1. Включите компьютер, дождитесь окончания загрузки операционной системы.

2. Рассмотрите значки, расположенные на рабочем столе.

3. Разместите значки на рабочем столе, расположив их по своему усмотрению, предварительно зацепив левой кнопкой мыши за значок и перетащив его в нужном направлении.

4. Наведите указатель мыши на значок объекта и, удерживая левую кнопку мыши, перетащите его в нужном направлении и отпустите кнопку.

5. Выполните обратное действие, выстроив значки автоматически по левому краю *Рабочего стола*. Выполните в контекстном меню (наведите указатель мыши на свободную от значков поверхность *Рабочего стола* и щелкните один раз правой кнопкой мыши) команду *Упорядочить значки → Автоматически*.

6. Рассмотрите *Панель задач*. Закройте и откройте *Главное меню* Windows, используя кнопку *Пуск*. Перетащите *Панель задач,* разместив ее по вертикали в правой части *Рабочего стола*. Верните *Панель задач* на место.

Задание 2: работа с окнами.

Методика выполнения задания:

1. Откройте системную папку *Мой компьютер*. Для этого наведите указатель мыши на значок папки *Мой компьютер* и выполните двойной щелчок левой кнопкой мыши или вызовите контекстное меню, щелкнув один раз правой кнопкой мыши, и выполните команду *Открыть*.

2. Рассмотрите окно и найдите следующие элементы окна: *Строка заголовка, Меню, полосы прокрутки*.

3. Включите и отключите панель инструментов *(Меню → Вид → Панель инструментов)*. Подводя курсор к каждой кнопке, прочитайте, для чего она используется.

4. Переместите окно в другое место *Рабочего стола*. Наведите указатель мыши на строку заголовка и, удерживая левую кнопку, переместите строку в нужном направлении.

5. Измените размеры окна. Наведите указатель мыши на любой угол окна или на любую его сторону, при этом указатель

мыши примет вид *двунаправленной стрелки*; зацепив за угол или сторону окна, потащите его в любом направлении.

6. Распахните окно на весь экран и верните ему прежний размер, используя кнопку *Развернуть* из набора кнопок управления окном. Наведите указатель мыши на элемент управления (кнопку *Развернуть*) и щелкните левой кнопкой мыши.

7. Сверните окно на *Панель задач* и разверните его. Выполните описанные в предыдущем пункте действия с кнопкой *Развернуть* из набора кнопок управления окном.

8. Откройте диск [C:] и ознакомьтесь с его содержимым. Наведите указатель мыши на значок диска [C:] и щелкните два раза (двойной щелчок) левой кнопкой мыши.

9. Запустите текстовый процессор Word.

10. Откройте последовательно папки (щелкните два раза левой кнопкой мыши на значке папки) Program Files\Microsoft Office\Office и щелкните левой кнопкой мыши на значке файла приложения *Word* (для поиска папок и файла используйте полосы горизонтальной и вертикальной прокрутки).

11. Выйдите из программы, закрыв окно. Для этого наведите указатель мыши на кнопку *Закрыть* из набора кнопок, управляющих окном, и щелкните левой кнопкой мыши.

12. Аналогично выполните запуск табличного процессора Excel, найдя значок файла приложения *Excel*, и откажитесь от работы с ним.

13. Закройте все окна на *Рабочем столе*. Наведите указатель мыши на кнопку *Закрыть* из набора кнопок, управляющих окном, и щелкните левой кнопкой мыши.

Задание 3: создание папок и файлов на *Рабочем столе*.

Методика выполнения задания:

1. На свободной поверхности *Рабочего стола* вызовите контекстное меню (щелкните правой кнопкой мыши) и выполните (наведите указатель мыши и щелкните левой кнопкой) команду *Создать → Папку*.

2. Наберите на клавиатуре имя папки: № КУРСА_ГРУППА № (например: 1КУРС_ГРУППА№401) и нажмите клавишу <*Enter*>.

3. Откройте свою папку. Вы ее только что создали.

4. В текущей папке создайте еще две папки, дайте им названия РАБОЧИЙ_СТОЛ и РАЗНОЕ.

5. В папке РАБОЧИЙ_СТОЛ создайте документ Microsoft Word под именем ОТВЕТ. Для этого на свободной поверхности рабочего стола вызовите контекстное меню и выполните команду *Создать → Документ Microsoft Word,* затем наберите на клавиатуре имя документа (файла), например ОТВЕТ, и обязательно нажмите клавишу <*Enter*>.

6. Откройте документ с именем ОТВЕТ (наведите указатель мыши и выполните двойной щелчок). Обратите внимание: документ пуст. Напечатайте названия всех объектов, находящихся на *Рабочем столе* вашего компьютера.

7. Сохраните изменения в документе, для этого выполните (наведите указатель мыши и щелкните левой кнопкой) команду меню *Файл → Сохранить.*

8. В папке РАЗНОЕ создайте точечный рисунок под именем КАРТИНКА. Для этого на свободной поверхности рабочего стола вызовите контекстное меню и выполните команду *Создать → Точечный рисунок.*

9. Наберите на клавиатуре имя документа (файла), например КАРТИНКА, и обязательно нажмите клавишу <*Enter*>.

10. Откройте документ с именем КАРТИНКА. Обратите внимание: документ пуст. Нарисуйте квадрат, в нем — треугольник, очертите квадрат кругом.

11. Сохраните изменения в документе. Для этого выполните (наведите указатель мыши и щелкните левой кнопкой) команду меню *Файл → Сохранить.*

Работа № 2. Программа *Проводник*

Цель работы: научиться работать с программой *Проводник*.

Указание: для выполнения практической работы надо знать объекты Windows и основные приемы управления в операционной среде Windows, уметь создавать папки и документы (файлы).

Задание 1: раскрытие (смена рабочей — активной — папки) и разворачивание папок.

Методика выполнения задания:

1. Включите персональный компьютер, дождитесь окончания загрузки операционной системы.

2. Запустите программу *Проводник* с помощью *Главного меню (Пуск → Программы → Стандартные → Проводник)*. Обратите внимание на то, какая папка открыта на правой панели *Проводника* в момент запуска.

3. Разыщите на левой панели папку *Мои документы* и откройте ее, щелкнув левой кнопкой мыши на значке папки. Обратите внимание на то, какая папка открыта на левой панели *Проводника*. На правой панели должно отобразиться содержимое папки *Мои документы,* т. е. папка *Мои документы* в данный момент активная (рабочая).

4. На левой панели *Проводника* найдите папку WINDOWS и разверните ее одним щелчком на значке узла «+». Обратите внимание на то, что раскрытие и разворачивание папок на левой панели — это разные операции. На левой панели отобразилось содержимое папки (подмножество) WINDOWS. Содержимое правой панели не изменилось, т. е. активной осталась папка МОИ ДОКУМЕНТЫ.

5. На левой панели *Проводника* внутри папки WINDOWS разыщите папку для временного хранения объектов под названием TEMP, раскройте ее. Обратите внимание, как изменилось содержимое панелей. Какая папка в настоящий момент активная?

Задание 2: создание и копирование файлов и каталогов (папок) с помощью программы *Проводник*.

Методика выполнения задания:

1. Запустите программу *Проводник* с помощью *Главного меню (Пуск → Программы → Стандартные → Проводник)*.

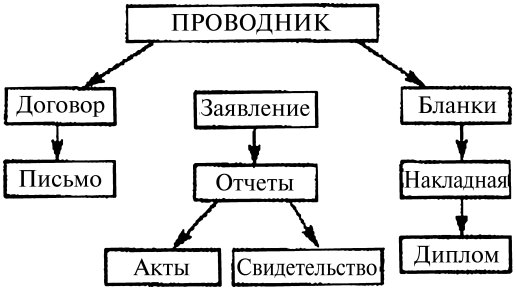


Рис. 3.9. Файловая структура

2. Откройте свой каталог архива практических работ и создайте в нем предложенную на рис. 3.9 файловую структуру.

3. Создайте папку ПРОВОДНИК. Для этого в правом окне программы *Проводник* откройте контекстное меню и вы-

полните команду *Создать* → *Папку,* а далее используйте алгоритм создания папок, описанный ранее.

4. В папке ПРОВОДНИК создайте подкаталоги согласно схеме на рис. 3.9.

5. На левой панели *Проводника* разверните все папки созданной файловой структуры. Для этого последовательно щелкайте левой кнопкой мыши на соответствующих значках узла «+».

6. Скопируйте папку АКТЫ в папку ДИПЛОМ. Для этого в левом окне *Проводника* правой кнопкой мыши перетащите значок папки АКТЫ, поместите его точно на значок ДИПЛОМ и выполните команду *Копировать*.

7. Переместите папку СВИДЕТЕЛЬСТВА в папку БЛАНКИ (перемещайте правой кнопкой мыши, выполнив команду *Переместить*).

8. Переименуйте папку ОТЧЕТЫ в папку ЗАЯВКА. Для этого вызовите контекстное меню объекта папки ОТЧЕТЫ и выполните команду *Переименовать*. Затем введите с клавиатуры новое имя и обязательно нажмите клавишу <*Enter*>.

9. В папке БЛАНКИ создайте ярлык объекта calc.exe стандартного приложения *Калькулятор,* расположенного по адресу: *C:\WINDOWS\SYSTEM32\calc.exe*. Для этого сделайте (раскройте) папку БЛАНКИ текущей, выполните команду контекстного меню *Создать* → *Ярлык,* затем используйте алгоритм создания ярлыка, описанный ранее.

10. В папке ПИСЬМА создайте текстовый файл, дав ему имя Налоговая_инспекция.txt. Для этого сделайте папку ПИСЬМА текущей, затем используйте алгоритм создания документа, описанный ранее.

11. В папке НАКЛАДНАЯ создайте документ Microsoft Word, дав ему имя Канцелярские_товары.

12. С помощью программы *Проводник* скопируйте файл Налоговая_инспекция.txt в папку ПРОВОДНИК. Для этого сделайте папку ПИСЬМА текущей, затем на правой панели *Проводника* наведите указатель мыши на значок файла Налоговая_инспекция.txt и, зацепив его правой кнопкой мыши, перетащите и поместите этот значок точно на значок папки ВАША_ФАМИЛИЯ на левой панели *Проводника*. Отпустите кнопку мыши и выполните команду *Копировать*.

13. Файл Канцелярские_товары переместите в папку БЛАНКИ. Для этого сделайте папку НАКЛАДНАЯ текущей, затем на правой панели *Проводника* наведите указатель мыши на значок файла Канцелярские_Товары и, зацепив его правой кнопкой мыши, перетащите и поместите этот значок точно на значок папки БЛАНКИ на левой панели *Проводника*. Отпустите кнопку мыши и выполните команду *Переместить*.

Работа № 3. Работа в окнах папки *Мой компьютер*

Цель работы:

— освоить работу посредством системы окон *Мой компьютер;*

— познакомиться с понятием «спецификация объекта» (файла, папки);

— освоить различные способы копирования, перемещения и удаления файлов и папок.

Задание 1: создание файлов и каталогов (папок) с помощью системы окон *Мой компьютер*.

Методика выполнения задания:

1. Откройте папку *Мой компьютер*. В строке меню выполните команду *Сервис → Свойство папки*. Выберите вкладку *Общие,* в появившемся диалоговом окне поставьте флажок *Открывать каждую папку в своем окне*.

2. Создайте папку *Мой компьютер\Ваш каталог архива Практических работ*\ПРОВОДНИК\ДОГОВОРА\ПИСЬМА\ПОЛЬЗОВАТЕЛЬ. Для этого необходимо папку ПИСЬМА сделать текущей (открыть последовательно окна перечисленных в спецификации объектов) и выполнить алгоритм создания папки. Сколько окон открыто на экране? Закройте окна.

3. Создайте папку *Мой компьютер\Ваш каталог архива Практических работ*\ПРОВОДНИК\ЗАЯВЛЕНИЯ\ЗАЯВКА\СВИДЕТЕЛЬСТВА\СТУДЕНТ. Для этого необходимо папку СВИДЕТЕЛЬСТВА сделать текущей (открыть последовательно окна перечисленных в спецификации объектов) и выполнить алгоритм создания папки. Сколько окон открыто на экране? Закройте окна.

4. Создайте файлы *Мой компьютер\Ваш каталог архива Практических работ*\ПРОВОДНИК\ДОГОВОРА\ПИСЬМА\ПОЛЬЗОВАТЕЛЬ\ответ.doc и *Мой компьютер\Ваш Каталог_архива_Практических_работ*\ПРОВОДНИК\ДОГОВОРА\ПИСЬМА\ПОЛЬЗОВАТЕЛЬ\проба_сил.txt. Для этого необходимо сделать папку ПОЛЬЗОВАТЕЛЬ текущей и далее использовать алгоритм создания документа, описанный в практической работе № 1.

5. Закройте окна.

Задание 2: выделение группы объектов, копирование, перемещение.

Методика выполнения задания:

1. Переместите файл *Мой компьютер\Ваш каталог архива Практических работ*\ПРОВОДНИК\ДОГОВОРА\ПИСЬМА\Налоговая_инспекция.txt в папку *Мой компьютер\Ваш каталог архива Практических работ*\ПРОВОДНИК\ДОГОВОРА\ПИСЬМА\ПОЛЬЗОВАТЕЛЬ. Для этого:

— последовательно откройте окна папок, указанных в спецификации файла;

— откройте окно папки ПОЛЬЗОВАТЕЛЬ;

— расположите окна папок ПИСЬМА и ПОЛЬЗОВАТЕЛЬ на экране так, чтобы они не перекрывали друг друга;

— зацепите правой кнопкой мыши значок файла Налоговая_инспекция.txt и перетащите в окно папки ПОЛЬЗОВАТЕЛЬ;

— отпустите кнопку мыши и выполните команду *Переместить*.

2. Скопируйте файл *Мой компьютер\Ваш каталог архива Практических работ*\ВАША ФАМИЛИЯ\БЛАНКИ\НАКЛАДНАЯ\ Канцелярские_товары.doc в папку *Мой компьютер\Ваш каталог архива Практических работ*\ВАША ФАМИЛИЯ\ДОГОВОРА\ПИСЬМА\ПОЛЬЗОВАТЕЛЬ. Для этого:

— последовательно откройте окна папок, указанных в спецификации файла;

— откройте окно папки ПОЛЬЗОВАТЕЛЬ;

— расположите окна папок НАКЛАДНАЯ и ПОЛЬЗОВАТЕЛЬ на экране так, чтобы они не перекрывали друг друга;

— зацепите правой кнопкой мыши значок файла Канцелярские_товары.doc и перетащите в окно папки ПОЛЬЗОВАТЕЛЬ;

— отпустите кнопку мыши и выполните команду *Копировать*.

3. Скопируйте папку *Мой компьютер\Ваш каталог архива Практических работ*\ПРОВОДНИК\ЗАЯВЛЕНИЯ\ЗАЯВКА\СВИДЕТЕЛЬСТВА\ СТУДЕНТ в папку *Мой компьютер\Ваш каталог архива Практических работ*\ВАША ФАМИЛИЯ\ДОГОВОРА\ПИСЬМА\ПОЛЬЗОВАТЕЛЬ. Для этого:

— последовательно откройте окна папок, указанных в спецификации папки;

— откройте окно папки ПОЛЬЗОВАТЕЛЬ;

— расположите окна папок СВИДЕТЕЛЬСТВА и ПОЛЬЗОВАТЕЛЬ на экране так, чтобы они не перекрывали друг друга;

— зацепите правой кнопкой мыши значок папки СТУДЕНТ и перетащите в окно папки ПОЛЬЗОВАТЕЛЬ;

— отпустите кнопку мыши и выполните команду *Копировать*.

4. Закройте окна папок, кроме окна папки ПОЛЬЗОВАТЕЛЬ.

5. Выделите:

— любой файл; для этого наведите указатель мыши на значок объекта и щелкните левой кнопкой;

— группу смежных файлов; для этого, удерживая нажатой клавишу *<Shift>,* щелкните на первом и последнем объектах выделяемой группы. Все промежуточные объекты выделятся автоматически;

— группу несмежных файлов; для этого, удерживая нажатой клавишу *<Ctrl>,* последовательно щелкайте левой кнопкой мыши на каждом из нужных объектов.

6. Выполните сортировку файлов по времени, размеру, объему; для этого в строке меню выполните соответствующую команду, например: *Вид → Упорядочить значки/по имени* и т. д.

7. Скопируйте все объекты папки ПОЛЬЗОВАТЕЛЬ в папку *Мой компьютер\Ваш каталог архива Практических работ*\ВАША ФАМИЛИЯ посредством буфера обмена. Для этого:

— откройте папку ВАША ФАМИЛИЯ, папка ПОЛЬЗОВАТЕЛЬ уже открыта;

— расположите окна папок на экране так, чтобы они не перекрывали друг друга;

— выделите группу необходимых объектов в папке-источнике (откуда копируете);

— вызовите контекстное меню и выполните команду *Копировать;*

— в папке-приемнике (куда копируете) вызовите контекстное меню и выполните команду *Вставить.*

8. Скопируйте все объекты папки ПОЛЬЗОВАТЕЛЬ в папку *Мой компьютер\Ваш каталог архива Практических работ*\ПРОВОДНИК\БЛАНКИ посредством буфера обмена. Для этого:

— откройте папку БЛАНКИ, папка ПОЛЬЗОВАТЕЛЬ уже открыта;

— расположите окна папок на экране так, чтобы они не перекрывали друг друга;

— выделите группу необходимых объектов в папке-источнике (откуда копируете) и нажмите одновременно клавиши <*Ctrl*> и <*C*> (внимание: при обозначении команд, выполняемых одновременным нажатием двух клавиш, принято использовать выражение: «нажать комбинацию клавиш...», а между наименованиями клавиш ставить знак «+», т. е. в данном случае: «нажать комбинацию клавиш <*Ctrl*> + <*C*>»);

— в папке-приемнике (куда копируете) нажмите комбинацию клавиш <*Ctrl*> + <*V*>.

9. Закройте окна.

10. Переместите все объекты папки БЛАНКИ в папку *Мой компьютер\Ваш каталог архива Практических работ*\ПРОВОДНИК\БЛАНКИ\НАКЛАДНАЯ\ДИПЛОМ посредством буфера обмена. Для этого:

— откройте папки БЛАНКИ и ДИПЛОМ;

— расположите окна папок на экране так, чтобы они не перекрывали друг друга;

— выделите группу необходимых объектов в папке-источнике (откуда копируете) и нажмите комбинацию клавиш <*Ctrl*> + <*V*>;

— в папке-приемнике (куда копируете) нажмите комбинацию клавиш <*Ctrl*> + <*V*>.

11. Переместите объекты обратно в папку БЛАНКИ, используя соответственно команды контекстного меню *Вырезать* и *Вставить.*

12. Удалите объект БЛАНКИ.

13. Отчет представьте в виде файловой структуры, развернутой в *Проводнике*.

Работа № 4. Антивирусная проверка с помощью программы *Dr.Web*

Цель работы:

— освоить работу с антивирусной программой Dr. Web;

— познакомиться с понятием «защита информации»;

— освоить различные способы обнаружения и удаления вирусов.

Задание 1: с помощью антивирусной программы Dr.Web провести проверку файлов с расширением .doc локального диска [C:], присвоив программе Dr.Web самый высокий приоритет. Проверить также память и загрузочные секторы диска.

Методика выполнения задания:

1. Запустите программу Dr.Web *(Пуск → Программы → Dr.Web)*.

2. Щелкните на кнопке *Дерево дисков,* выберите диск [C:].

3. Щелкните на кнопке *Установка* и во вкладке *Проверка* установите флажки: *Проверять память, Проверять загрузочные секторы,* во вкладке *Типы* укажите файлы с расширением *.doc.

4. Во вкладке *Общие* установите самый высокий приоритет программе Dr.Web.

5. Щелкните на кнопке *Статистика,* просмотрите результаты работы программы.

6. Отчет оформите в виде фотографии экрана.

Задание 2: с помощью антивирусной программы Dr.Web провести проверку всех файлов диска [C:] в режиме обнаружения неизвестных вирусов, при этом проверять архивные файлы.

Методика выполнения задания:

1. Запустите программу Dr.Web *(Пуск → Программы → Dr.Web)*.

2. Щелкните на кнопке *Дерево дисков*, выберите диск [C:].

3. Щелкните на кнопке *Установка,* во вкладке *Проверка* установите флажок *Эвристический анализ.*

4. Щелкните на кнопке *Установка,* во вкладке *Типы* установите флажки: *Все файлы, Файлы в архивах.*

5. Щелкните на кнопке *Список отчета*, просмотрите результаты работы.

6. Отчет о работе оформите в виде фотографии экрана.

Работа № 5. Антивирусная проверка с помощью программы *AVP*

Цель работы:

— освоить работу с антивирусной программой AVP;

— освоить различные способы обнаружения вирусов и их удаления.

Задание: с помощью антивирусной программы AVP-Сканер произвести проверку локального диска [C:], включая сканирование системной памяти, системных секторов и файлов в режиме обнаружения неизвестных вирусов и предупреждения о поврежденных файлах или подозрительных последовательностях машинных инструкций в памяти компьютера.

Методика выполнения работы:

1. Запустите программу AVP *(Пуск → Программы → AntiViral Tookit Pro → AVP-Сканер).*

2. В *Главном окне* установите флажок *Локальные диски*. Если имеется несколько локальных дисков, снимите выделение со всех, за исключением диска [C:].

3. Во вкладке *Объекты* установите флажки: *Память, Секторы, Файлы.*

4. Во вкладке *Параметры* установите флажки: *Предупреждения* и *Анализатор кода*.

5. Щелкните на кнопке *Пуск* в *Главном окне.*

6. Во вкладке *Статистика* просмотрите результаты работы программы.

7. Оформите отчет в виде фотографии экрана.

КОНТРОЛЬНЫЕ ВОПРОСЫ К РАЗДЕЛУ 3

1. Дайте определение ЭВМ.

2. Сформулируйте принципы построения ЭВМ фон Неймана.

3. Что такое архитектура и конфигурация ЭВМ?

4. **Опишите шинную организацию ЭВМ.**
5. **Назовите состав и функции блоков центрального процессора ЭВМ.**
6. **Для чего предназначены регистры?**
7. **Какие функции выполняет устройство управления ЭВМ?**
8. **Расскажите о составе устройств памяти ЭВМ.**
9. **Какова система шин ЭВМ?**
10. **Какова последовательность работы элементов ЭВМ при обработке информации?**
11. **Что такое ПЭВМ?**
12. **В чем отличие ПЭВМ от ЭВМ других классов?**
13. **Из каких устройств состоит системный блок? Каково их назначение?**
14. **Какие периферийные устройства входят в состав ПЭВМ?**
15. **Назовите назначение и основные характеристики периферийных устройств.**
16. **Расскажите об известных вам внешних запоминающих устройствах ПК.**
17. **Какие виды печатающих устройств ПК вы знаете?**
18. **Что такое операционная система ПК?**
19. **Дайте определение понятия «пользовательский интерфейс».**
20. **Из каких этапов состоит технология информационных процессов в ПК?**
21. **Назовите функции наиболее распространенных текстовых процессоров.**
22. **Расскажите о работе на ПК с графическими объектами.**
23. **Дайте краткую характеристику табличного процессора.**
24. **Как вы понимаете диалоговый режим работы ПК?**
25. **Перечислите основные направления развития ПЭВМ.**
26. **По каким характеристикам осуществляется классификация ЭВМ по поколениям?**
27. **Дайте характеристику ЭВМ четвертого поколения.**
28. **Расскажите об особенностях ЭВМ пятого поколения.**

Раздел 4

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

В результате изучения материала четвертого раздела студент должен:

знать:

— назначение и возможности электронных таблиц;

— назначение и основные возможности баз данных;

— основные объекты баз данных и допустимые операции над ними;

уметь:

— применять текстовый редактор для редактирования и форматирования текстов;

— применять графический редактор для создания и редактирования изображений;

— строить диаграммы;

— применять электронные таблицы для решения задач;

— создавать простейшие базы данных;

— осуществлять сортировку и поиск информации в базе данных;

иметь представление:

— о возможности соединения разнотипной информации в одном электронном документе с помощью технологии мультимедиа;

— о работе электронной почты;

— об информационных ресурсах и технологиях поиска информации в сети Internet.

Методические указания

Преподаватель должен донести до студентов две очень важные мысли: первая — информационная технология (ИТ) — это умение правильно работать с информацией, которое обеспечи-

вается совокупностью программно-технических средств и приемов; вторая — в настоящее время разработано довольно большое число ИТ, и в будущем их возможности будут очень быстро расширяться.

Студенты должны уяснить, что:

— пакеты прикладных программ делают применение ЭВМ массовым, освобождая пользователя от самостоятельного составления сложных программ;

— программным средством можно пользоваться, даже не зная его структуры и методов, положенных в его основу;

— с помощью одного и того же ППП можно решать различные классы задач (иногда не очень похожих друг на друга);

— ИТ существенно облегчают хранение и поиск нужной информации;

— для решения различных жизненных задач нужны разные ИТ;

— ресурсы ЭВМ (в первую очередь объем памяти и быстродействие) и их эксплуатационные характеристики влияют на возможности, предоставляемые ИТ.

Необходимо подчеркнуть сложность технологического процесса сбора, передачи, обработки и хранения информации и важность информационных систем как составляющей инфраструктуры общества, в котором информация выступает одним из главных ресурсов его жизнедеятельности. Отсюда становится очевидным, что автоматизированная информационная технология представляет собой совокупность методов и способов сбора, передачи, накопления, хранения, поиска и обработки информации на основе применения средств вычислительной техники и связи.

Любая вычислительная сеть — это совокупность технических, коммуникационных и программных средств, обеспечивающих эффективное распределение вычислительных ресурсов. Вычислительные сети, являясь одновременно и продуктом, и мощным стимулом развития интеллекта человека, позволяют:

— построить мощные хранилища информации (БД);

— расширить перечень решаемых задач по обработке информации;

— повысить надежность информационной системы за счет дублирования работы ПК;

— создать новые виды сервисного обслуживания, например электронную почту;

— снизить стоимость обработки информации.

Появление новейших ИТ в среде глобальных сетей открыло дорогу к централизации обработки и использования информации, т. е. осуществляется возврат к централизации ИТ для коллективного использования общих информационных ресурсов.

Тема 4.1

КЛАССИФИКАЦИЯ И ХАРАКТЕРИСТИКА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

4.1.1. Введение

Современные информационные системы организационного управления предназначены оказывать помощь специалистам и руководителям, принимающим решения, в получении в необходимом количестве своевременной и достоверной информации; создании условий для организации автоматизированных офисов, проведения с применением компьютеров и средств связи оперативных совещаний, сопровождаемых звуковым и видеорядом. Достигается это переходом на новую информационную технологию.

Новая (современная) информационная технология — это технология, основанная на:

— повсеместном применении ЭВМ и оргтехники;

— активном участии пользователей (непрофессионалов в области вычислительной техники и программирования) в информационном процессе;

— высоком уровне дружественного пользовательского интерфейса;

— широком применении ППП общего и проблемного назначения;

— возможности доступа пользователя к базам данных и программ, в том числе и удаленным, благодаря локальным и глобальным сетям ЭВМ;

— анализе ситуаций при выработке и принятии управленческих решений с помощью автоматизированных рабочих мест специалистов;

— применении систем искусственного интеллекта;
— внедрении экспертных систем;
— использовании телекоммуникаций;
— создании геоинформационных систем и других технологий.

4.1.2. Состав и содержание информационных технологий

Стремительное развитие и совершенствование эксплуатационных возможностей ЭВМ создало предпосылки для автоматизации умственного труда и формирования рынка информационных продуктов и услуг. Развитие ИТ шло параллельно с появлением новых видов технических средств обработки и передачи информации, новых средств коммуникаций.

В настоящее время ИТ классифицируют по следующим признакам: способу реализации в автоматизированных информационных системах (АИС), степени охвата задач управления, классам реализуемых технологических операций, типу пользовательского интерфейса, вариантам использования сети ЭВМ, обслуживаемым предметным областям и др.

По способу реализации в АИС информационные технологии подразделяют на традиционные и современные. *Традиционные* ИТ существовали в условиях централизованной обработки данных, до периода массового использования ПЭВМ. Они были ориентированы главным образом на снижение трудоемкости пользователя (например, инженерные и научные расчеты, формирование регулярной отчетности на предприятиях и др.). *Новые (современные)* ИТ связаны в первую очередь с информационным обеспечением процесса управления в режиме реального времени.

По степени охвата задач управления выделяют ИТ электронной обработки данных, автоматизации функций управления, поддержки принятия решений, электронного офиса, экспертной поддержки. *Электронная обработка данных* выполняется с использованием ЭВМ без пересмотра методологии и организации процессов управления при решении локальных математических и экономических задач. При *автоматизации управленческой деятельности* вычислительные средства, включая суперЭВМ и ПЭВМ, используются для комплексного

решения функциональных задач, формирования регулярной отчетности и работы в информационно-справочном режиме для подготовки управленческих решений. К этой же группе относятся ИТ *поддержки принятия решений,* которые предусматривают широкое использование экономико-математических методов и моделей, ППП для аналитической работы и формирования прогнозов, составления бизнес-планов, обоснованных оценок и выводов относительно процессов и явлений производственно-хозяйственной деятельности.

К названной группе относятся и широко внедряемые в настоящее время ИТ *электронного офиса* и *экспертной поддержки решений*. Эти два варианта ИТ ориентированы на использование достижений в области новейших методов автоматизации работы специалистов и руководителей, создание наиболее благоприятных условий для выполнения профессиональных функций, качественного и своевременного информационного обслуживания за счет автоматизированного набора управленческих процедур, реализуемых в условиях конкретного рабочего места и офиса в целом.

Электронный офис предусматривает наличие интегрированных ППП, которые обеспечивают комплексную реализацию задач предметной области. В настоящее время все большее распространение приобретают электронные офисы, сотрудники и оборудование которых могут находиться в разных помещениях. Необходимость работы с документами, материалами и БД конкретного предприятия или учреждения в гостинице, транспорте, дома привела к появлению электронных офисов, включенных в соответствующие сети ЭВМ.

ИТ *экспертной поддержки* составляют основу автоматизации труда специалистов-аналитиков. Эти работники, кроме аналитических методов и моделей для исследования складывающихся ситуаций, вынуждены использовать сведения, составляющие базу знаний в конкретной предметной области.

По классам реализуемых технологических операций информационные технологии подразделяют на работающие с текстовым и табличным процессорами; графическими объектами; системами управления БД; гипертекстовыми и мультимедийными системами.

Подчеркнем, что создание программных средств для вывода высококачественного звука и видеоизображения — перспек-

тивное направление развития компьютерной технологии. Технология формирования видеоизображения получила название компьютерной графики. **Компьютерная графика —** это создание, хранение и обработка моделей объектов и их изображений с помощью ЭВМ. Эта технология проникла в область моделирования различных конструкций (машиностроение, авиационная техника, автомобилестроение, строительная техника и др.), экономического анализа, проникает в рекламную деятельность, делает занимательным досуг. Формируемые и обрабатываемые с помощью цифрового процессора изображения могут быть демонстрационными и анимационными. К *демонстрационным изображениям* относят, как правило, коммерческую (деловую) и иллюстративную графику. К *анимационной графике* принадлежит инженерная и научная графика, а также графика, связанная с рекламой, искусством, играми, когда на экран выводятся не только одиночные изображения, но и последовательность кадров в виде фильма (интерактивный вариант). *Интерактивная графика* — одно из наиболее прогрессивных направлений среди современных ИТ. Бурное развитие этого направления связано с появлением новых графических станций и специализированных программных средств, позволяющих создавать реалистические объемные движущиеся изображения, сравнимые по качеству с кадрами видеофильма.

В классическом понимании *система управления БД* (СУБД) представляет собой набор программ, позволяющих создавать и поддерживать БД в актуальном состоянии. Обычно любой текст представляется как одна длинная страница символов, которая читается в одном направлении. *Гипертекстовая технология* заключается в том, что текст представляется как многомерный, т. е. как иерархическая структура. Материал текста делится на фрагменты. Каждый видимый на экране ЭВМ фрагмент, дополненный многочисленными связями с другими фрагментами, позволяет уточнить информацию об изучаемом объекте и двигаться в одном или нескольких направлениях по выбранной связи.

Программно-техническая организация обмена с компьютером текстовой, графической, аудио- и видеоинформацией получила название **мультимедиа-технологии**.

По типу пользовательского интерфейса можно рассматривать ИТ с точки зрения возможностей доступа

пользователя к информационным и вычислительным ресурсам. Так, *пакетная* ИТ исключает возможность пользователя влиять на обработку информации, пока она проводится в автоматическом режиме. Это объясняется организацией обработки информации, которая основана на выполнении программно-заданной последовательности операций над заранее накопленными в системе и объединенными в пакет данными. В отличие от пакетной, *диалоговая* ИТ предоставляет пользователю неограниченную возможность взаимодействовать с хранящимися в системе информационными ресурсами в реальном масштабе времени, получая при этом всю необходимую информацию для решения функциональных задач и принятия решений. Интерфейс *сетевой* ИТ предоставляет пользователю средства теледоступа к территориально распределенным информационным и вычислительным ресурсам благодаря развитым средствам связи.

В настоящее время наблюдается тенденция к объединению различных типов ИТ в единый компьютерно-технологический комплекс, который носит название **интегрированного**. Особое место в нем принадлежит средствам коммуникации, не только обеспечивающим чрезвычайно широкие технологические возможности автоматизации управленческой деятельности, но и являющимся основой создания самых разнообразных сетевых вариантов ИТ: *локальных, многоуровневых, распределенных* и *глобальных информационно-вычислительных сетей*.

Чрезвычайно разнообразна классификация ИТ по обслуживаемым предметным областям. Например, только в экономике таковыми являются бухгалтерский учет, банковская, налоговая и страховая деятельность и др.

Тема 4.2

ТЕХНОЛОГИЯ ОБРАБОТКИ ТЕКСТОВОЙ ИНФОРМАЦИИ

4.2.1. Текстовый редактор: назначение и основные функции

Пользователь ПЭВМ часто встречается с необходимостью подготовки тех или иных документов — писем, статей, служебных записок, отчетов, рекламные материалов и т. д. Для их под-

готовки текст редактируемого документа выводится на экран, и пользователь может в диалоговом режиме вносить в него свои изменения. Все внесенные изменения фиксируются. При распечатке выводится отформатированный текст, в котором учтены все исправления. Пользователь может переносить части текста из одного места документа в другое, использовать несколько видов шрифтов для выделения отдельных участков текста, печатать подготовленный документ на принтере в нужном количестве экземпляров.

Удобство и эффективность применения компьютера привели к созданию множества программ для обработки документов. Такие программы называются **текстовыми процессорами** или **редакторами текстов программ**. Другими словами, редактор текста — это программа или ее часть, обеспечивающая создание текстовых документов и их корректирование. При этом текстовый документ — это не только статья, письмо, служебная записка, но и тексты программ, команды ОС. Любой пользовательский интерфейс должен обеспечить работу с текстом, а значит, должен иметь в своем составе текстовый редактор. Около 80 % времени работы всех компьютеров в мире уходит на работу с текстами.

Следует подчеркнуть, что в информатике под документом (от лат. documentum — свидетельство) понимается совокупность данных в электронном виде, подтверждающих, объясняющих, свидетельствующих о чем-либо. Это не обязательно текст, это может быть письмо со встроенным музыкальным фрагментом или видеоклипом, доклад со встроенной электронной таблицей, файл презентации.

Возможности текстовых редакторов различны — от программ, предназначенных для подготовки небольших документов простой структуры, до программ для набора, оформления и полной подготовки к типографскому изданию книг и журналов (издательские системы).

Редакторы текстов программ редактируют программы на том или ином языке программирования. Часто они встроены в систему программирования. Это редакторы Turbo (Borland), C++, Turbo Pascal, Multi-Edit, Brief и т. д. Они выполняют следующие функции: диалоговый просмотр текста; редактирование строк программы; копирование и перенос блоков текста; копирование одной программы или ее части в указанное место другой программы; контекстный поиск и замену текста; авто-

матический поиск строки, содержащей ошибку; распечатку программы или ее части.

Часто редакторы текстов программ позволяют автоматически проверять синтаксическую правильность программ; иногда они объединены с отладчиками программ на уровне исходного текста. Редакторы текстов программ можно использовать для создания и корректирования небольших документов. Однако для серьезной работы с документами предпочтительнее использовать специальные редакторы, ориентированные на работу с текстами, имеющими структуру документа, т. е. состоящими из разделов, страниц, абзацев, приложений, слов и т. д. Такие редакторы обеспечивают следующие функции: возможность использования различных шрифтов символов, работу с пропорциональными шрифтами, задание произвольных междустрочных промежутков, автоматический перенос слов на новую строку, автоматическую нумерацию страниц, обработку и нумерацию сносок, печать верхних и нижних заголовков страниц, выравнивание краев абзацев, набор текста в несколько столбцов, проверку правописания и подбор синонимов, построение оглавления индексов, сортировку текстов и данных и т. д.

Существует несколько сотен редакторов текстов — от самых простых до весьма мощных и сложных. Самые распространенные из них — Microsoft Word, WordPerfect, WordStar. В США наиболее популярны Microsoft Word для Windows и WordPerfect, в Европе и России — Microsoft Word.

Среди простых редакторов текста в России распространение получил «Лексикон». Он имеет интерфейс на русском языке и позволяет готовить несложные документы с текстом на русском и английском языках. «Лексикон» вполне подходит тем, кому нужен простой инструмент для подготовки небольших и несложных документов, не требующих высокого полиграфического качества. Но возможностей «Лексикона» недостаточно, если требуется обеспечить высокое качество печати или подготовить сложные документы большого объема, рекламные буклеты или книги. В таком случае лучше воспользоваться Microsoft Word.

В Microsoft Word для Windows реализована фоновая проверка орфографии. По мере введения текста текстовый редактор проверяет его и подчеркивает слова, содержащие ошибки, красной волнистой линией. Ошибки проверяются не только

в словах, но и в предложениях. Сомнительные словосочетания и предложения подчеркиваются волнистой зеленой линией.

Microsoft Word — мощный интеллектуальный текстовый редактор для создания профессионально оформленных документов. Он содержит инструмент рисования таблиц путем обычного рисования линий в тех местах, где они должны быть в таблице. Эти линии автоматически превращаются в элементы таблицы. Выравнивание введенных линий по краям таблицы также происходит автоматически. Внешний вид текстового редактора Microsoft Word показан на рис. 4.1.

Кроме того, Microsoft Word работает с *Мастером писем*, который позволяет установить параметры письма, его оформление, вставить общий текст (например, обратный адрес и адрес получателя), а также отправить письмо. ПЭВМ всегда оснащена одним или несколькими текстовыми процессорами.

Если назначение всех текстовых редакторов в целом одно и то же, то предоставляемые ими возможности и средства их реализации — разные. То же относится к графическим процессорам и электронным таблицам. Среди текстовых процессоров Windows (как наиболее распространенной среды) можно выделить Write и Word. Технология их использования основана на интерфейсе WIMP, но возможности Word значительно расширены, и в какой-то мере его можно рассматривать как настольную типографию.

Текстовые процессоры (редакторы) обеспечивают следующие функции: набор текста, хранение его на магнитных носителях, просмотр и печать. В большинстве текстовых процессоров реализованы функции проверки орфографии, выбора шрифтов и кеглей, центровки заголовков, разбиения на страницы, печати в одну или несколько колонок, вставки в текст таблиц и рисунков, использования шаблонов, постраничных ссылок, перемещения фрагментов текста, изменения структуры документа.

С помощью средств форматирования можно создать внешний вид документа, изменить стиль, подчеркнуть, выделить курсивом, изменить размеры символов, выделить абзацы, выровнять их влево, вправо, к центру, выделить рамкой. Перед печатью документ можно просмотреть, проверить текст, выбрать размер бумаги, задать число копий для вывода. Повторяющиеся фрагменты текста можно обозначить как *Автотекст*

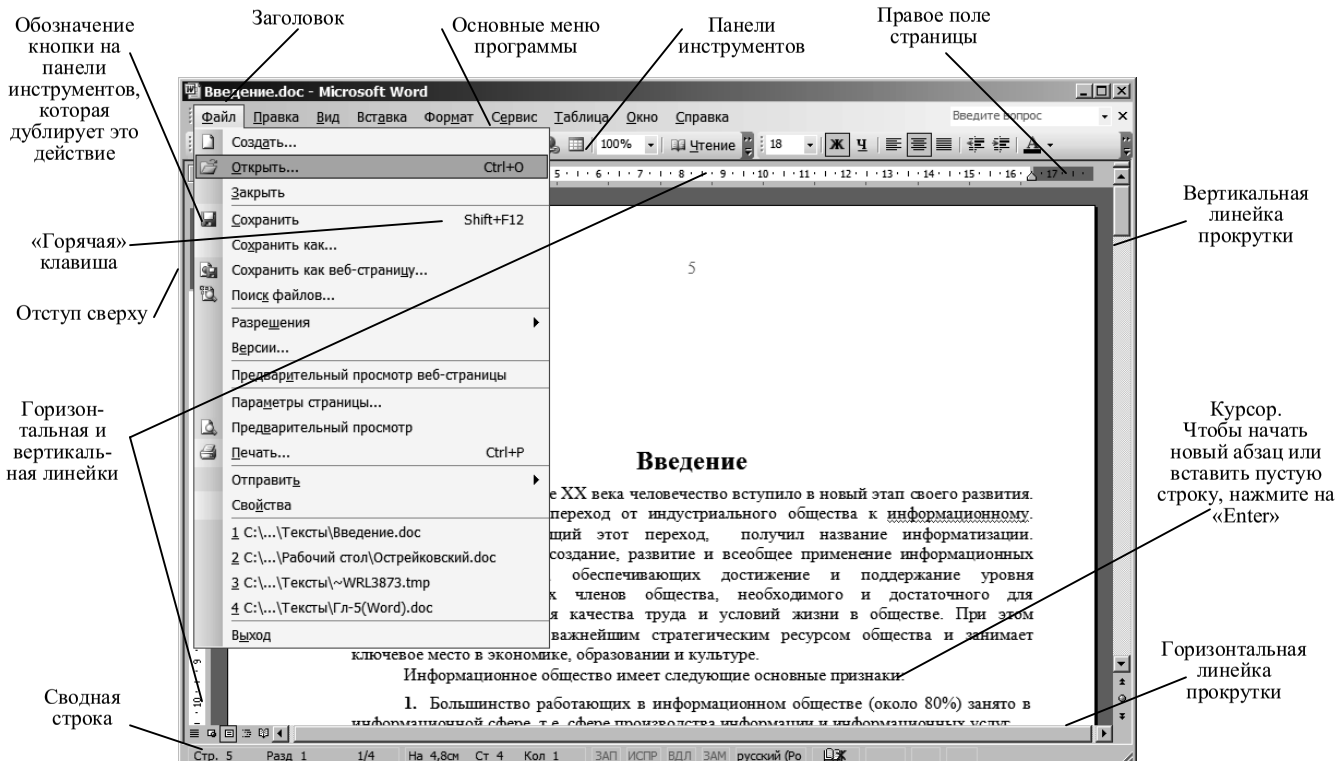


Рис. 4.1. Внешний вид текстового процессора Microsoft Word

и присвоить ему имя; в дальнейшем при указании в нужном месте имени *Автотекста* текстовый процессор автоматически вставит туда повторяющийся фрагмент.

4.2.2. Издательские системы

Для подготовки рекламных буклетов, оформления журналов и книг служат специальные издательские системы. Они позволяют готовить их и печатать на лазерных принтерах или выводить на фотонаборные автоматы сложные документы высокого качества.

Настольные издательские системы (**НИС**) — это программы, предназначенные для профессиональной издательской деятельности, позволяющие осуществлять электронную верстку широкого спектра основных типов документов (информационного бюллетеня, краткой цветной брошюры, объемного каталога, торговой заявки, справочника). Предусмотренные в пакетах данного типа средства позволяют:

— компоновать (верстать) текст;

— использовать всевозможные шрифты и полиграфические изображения;

— осуществлять редактирование текста на уровне лучших текстовых процессоров;

— обрабатывать графические изображения;

— обеспечивать вывод документов высокого качества;

— работать в сетях и на разных платформах.

Наилучшими пакетами в этой области для ПЭВМ являются Corel Ventura, PageMaker, QuarkXPress, FrameMaker, Microsoft Publisher, PagePlus, Compu Work Publisher.

Имеются два основных вида издательских систем. Издательские системы первого вида очень удобны для подготовки небольших материалов с иллюстрациями, графиками, диаграммами, различными шрифтами в тексте (например, газет, небольших журналов). Типичный пример такой системы — Aldus PageMaker.

Издательские системы второго вида более подходят для подготовки объемных документов, например книг. Одна из самых распространенных таких систем — Ventura Publisher (Corel Ventura) — управляется меню и может считывать тексты, под-

готовленные с помощью других текстовых редакторов (например, Microsoft Word), сохраняя при этом параметры форматирования, заданные этими редакторами.

Основная операция издательских систем — верстка (размещение текста по страницам документа, вставка рисунков, оформление текста различными шрифтами и т. д.). В режиме ввода редактирования текста Ventura и Aldus PageMaker значительно уступают в скорости и удобстве текстовым редакторам. Поэтому чаще всего документы подготавливают в два этапа: набирают текст в редакторе типа Microsoft Word, а затем считывают его системой Aldus PageMaker или Ventura и осуществляют окончательную подготовку документа.

Основные функции настольных издательских систем: использование сотен видов шрифтов (начертаний и размеров символов текста), которые отображаются на экране так же, как при печати; размещение фрагментов в документе; изменение и корректировка рисунков и диаграмм; изменение интервала между буквами (обычный, разреженный или уплотненный); формирование таблиц; выравнивание нижнего края текста на странице на заданную границу (чтобы страницы документа имели единообразный вид); набор формул и т. д.

Многим пользователям дли выполнения издательских работ вполне достаточно возможностей Microsoft Word для Windows. В последнее время производители издательских систем стали встраивать в них элементы профессионального цветоделения, обеспечивающие подготовку высококачественных цветных изданий, а также средства графических редакторов.

Тема 4.3

ТЕХНОЛОГИЯ ОБРАБОТКИ ГРАФИЧЕСКОЙ ИНФОРМАЦИИ

4.3.1. Теоретические основы представления графической информации

Графика — наиболее общий способ визуального представления данных в компьютере, в котором объединяются текстовые данные и графические образы.

Графический редактор — это программный комплекс, обеспечивающий пользователя средствами для создания графических образов, картин, рисунков и даже мультипликации. Он позволяет рисовать на экране дисплея разными цветами с помощью *Пера*, мыши и пр. Кроме того, графический редактор заботится о сохранности созданного изображения и удобстве работы с принтером, памятью и т. п.

Минимальный элемент изображения (или картинки) на экране дисплея называется **пикселем** (от англ. picture element — элемент изображения, сокращенно — pixel). Пиксель может храниться, адресоваться и показываться.

Как минимальный элемент изображения пиксель — это фактически точка на экране дисплея. Количество цветов, которые может отображать пиксель, называется **глубиной пикселя**. Она определяется длиной значения цвета. Например, при длине 4 бит пиксель может определять 16 цветов.

Элементарные объекты (треугольники, окружности и пр.), из которых создается изображение, получили название **примитивов** или **графических примитивов** (от англ. graphics primitive — графический примитив как графический элемент, из которого состоят графические объекты). Например, из треугольников или ломаной можно создать изображение горы. Чем больше графических примитивов будет просчитано, тем ближе к естественному восприятию будет изображение.

Имеется два способа, или формата, представления графического изображения на машинных носителях — растровая и векторная графика.

Растровая графика — способ представления изображения в виде набора точек. Каждая точка является *элементом* растра, ее описание хранится в специальных растровых файлах. Существует несколько форматов растровых файлов, например DIB (Device Independent Bitmap) — аппаратно-независимый растровый формат, используемый в Windows.

Векторная графика — это способ представления изображения как совокупности графических элементов (графических примитивов: отрезков, дуг и пр.), описанных любым способом, в том числе графическими *командами*. Графические команды хранятся в метафайлах, которые чаще всего представляются как файлы в двоичном коде, но могут иметь вид ASCII-текста.

4.3.2. Графические редакторы

Потребность ввода графиков, схем, диаграмм, рисунков в текст или документ вызвала необходимость создания графических процессоров. Графические процессоры представляют собой инструментальные средства, позволяющие создавать и модифицировать графические образы с использованием иллюстративной, коммерческой, научной и когнитивной графики.

Информационные технологии (ИТ) иллюстративной графики позволяют создавать иллюстрации для различных текстовых документов в виде регулярных структур — различных геометрических фигур (векторная графика). Процессоры, реализующие ИТ иллюстративной растровой графики, дают возможность пользователю выбрать толщину и цвет линий, палитру заливки, шрифт для записи и наложения текста, создавать разные графические образы, а также стирать или разрезать рисунок, перемещать его части. Эти средства реализованы в пакете Paint Brush. Существуют также ИТ, позволяющие просматривать изображения в режиме слайдов, спецэффектов и вживлять их (Corel DRAW, Storyboard, Animator, 3DStudio).

ИТ коммерческой, или деловой, графики обеспечивают отображение информации, хранящейся в табличных процессорах, БД и отдельных локальных файлах, в виде двух- или трехмерных графиков, круговой диаграммы, столбиковой гистограммы, линейных графиков и др. (Excel, FoxPro и т. д.).

ИТ научной графики предназначены для обслуживания задач оформления научных отчетов, содержащих математические, химические и прочие формулы, задач картографии и др.

Когнитивные компьютерные средства — это комплекс виртуальных устройств, программ и систем, реализующих комплексную обработку зрительной информации в виде образов, процессов, структур. Когнитивная графика позволяет представить в виде зрительных образов различные математические формулы и закономерности для доказательства сложных теорем, открывает новые возможности для познания законов функционирования сознания — этой наиболее сложной и сокровенной тайны мироздания. Средства когнитивной графики связаны со многими новейшими ИТ, включая гипертексты и мультимедиа.

Большинство графических процессоров удовлетворяет стандарту пользовательского интерфейса WIMP. Панель содержит меню действий, линейки инструментов и цветов. Линейка инструментов состоит из набора графических символов, требующихся для построения практически любого рисунка. Линейка цветов содержит цветовую гамму монитора ПК.

Графические редакторы-пакеты, предназначенные для обработки графической информации, подразделяют на ППП обработки для растровой и векторной графики.

ППП для обработки растровой графики предназначены для работы с фотографиями и включают в себя набор средств по кодированию фотоизображений в цифровую форму. Признанный лидер среди пакетов данного класса — Adobe Photoshop. Известны также пакеты Picture Publisher, Photo Works Plus. Все программы ориентированы на работу в среде Windows.

ППП для обработки векторной графики предназначены для профессиональной работы, связанной с художественной и технический иллюстрацией, с последующей цветной печатью (на рабочем месте дизайнеров, например). Они занимают промежуточное положение между пакетами для систем автоматизированного проектирования (САПР) и настольными издательскими системами (НИС). Пакеты данного класса обладают достаточно широким набором функциональных возможностей для осуществления сложной и точной обработки графических изображений и включают в себя (помимо инструментария для создания графических изображений) средства:

— выравнивания (по базовой линии и странице, по сетке, пересечению, ближайшей точке и т. п.);

— манипулирования объектами;

— обработки текста в части оформления и модификации параграфов, работы с различными шрифтами;

— импорта (экспорта) графических объектов (файлов) различных форматов;

— вывода на печать с соответствующей настройкой экранного образа на полиграфическое исполнение;

— настройки цвета.

Своеобразным стандартом среди пакетов этого класса является CorelDRAW. Можно также отметить Aldus Free Hand, Freelance Graphics.

ППП демонстрационной графики являются конструкторами графических образов деловой информации, призванными в наглядной и динамичной форме представлять результаты некоторого аналитического исследования. Работа с пакетами этого типа строится по следующей схеме: разработка общего плана представления, выбор шаблона для оформления элементов, формирование и импорт элементов, таких, как текст, графики, таблицы, диаграммы, звуковые эффекты и видеоклипы. Программы просты в работе и снабжены интерфейсом, почти не требующим дополнительного изучения. К наиболее популярным пакетам данного типа относятся PowerPoint, Harvard Graphics, WordPerfect Presentations, Freelance Graphics.

Тема 4.4

ТЕХНОЛОГИЯ ОБРАБОТКИ ЧИСЛОВОЙ ИНФОРМАЦИИ

4.4.1. Электронные таблицы: назначение и основные функции

Множество задач, которые предстоит решить фирмам и предприятиям, носят учетно-аналитический характер и требуют табличной компоновки данных с подведением итогов по различным группам и разделам данных, например при составлении баланса, справок для налоговых органов, всевозможных финансовых отчетов и т. п. Для хранения и обработки информации, представленной в табличной форме, используют электронные таблицы (ЭТ).

Программные средства для проектирования таблиц называют также **табличными процессорами**. Они позволяют не только создавать таблицы, но и автоматизировать обработку табличных данных. Кроме того, с помощью ЭТ можно выполнять различные экономические, бухгалтерские и инженерные расчеты, а также строить разного рода диаграммы, проводить сложный экономический анализ, моделировать и оптимизировать решение различных хозяйственных операций и многое другое.

Функции табличных процессоров весьма разнообразны и включают:

— создание и редактирование ЭТ;

— оформление и печать ЭТ;

— создание многотабличных документов, объединенных формулами;

— построение диаграмм, их модификацию и решение экономических задач графическими методами;

— работу с электронными таблицами как с базами данных (сортировка таблиц, выборка данных по запросам);

— создание итоговых и сводных таблиц;

— использование при построении таблиц информации из внешних баз данных;

— решение экономических задач типа «что будет, если...» путем подбора параметров;

— решение оптимизационных задач;

— статистическую обработку данных;

— создание слайд-шоу;

— разработку макрокоманд, настройку среды под потребности пользователя и т. д.

Табличные процессоры различаются в основном набором выполняемых функций и удобством интерфейса, поэтому целесообразно проанализировать лишь широко используемые программные продукты.

Перспективные направления разработки ЭТ основными фирмами-разработчиками определяются по-разному. Так, фирма Microsoft уделяет первостепенное внимание совершенствованию набора функциональных средств табличного процессора Excel. В Excel многие функции разработаны более тщательно, чем в других редакторах ЭТ, а возможность использования массивов обеспечивает большую гибкость при работе с таблицами.

Фирма Lotus основные усилия сконцентрировала на разработке инструментов групповой работы. Версия 4.0 пакета Lotus 1-2-3 дополнена Version Manager для моделирования по принципу «что будет, если...», а версия 5.0 дополнена средствами маршрутизации и связи с Notes, что позволяет создать приложения в других пакетах. К сильным сторонам Lotus 1-2-3 можно отнести простоту создания и редактирования графиков, а также наиболее логичную структуру трехмерных таблиц. Предусмотрено также совершенствование групповой работы с таблицами: использование Team Consolidate предоставляет возмож-

ность группе пользователей редактировать копии ЭТ, а затем их объединять. В версию пакета для Windows включен язык программирования Lotus Script.

Пакет Quattro Pro в результате тестирования получил достаточно высокие оценки, но ни одна из особенностей этого пакета не вызвала к себе повышенного внимания. Наиболее привлекательными оказались возможности сортировки данных (которые хорошо реализованы в Excel), а также удобство эксплуатации. В то же время отмечались сложности при освоении графических возможностей Quattro Pro и недостаточный объем справочной информации.

Ситуация, сложившаяся к настоящему времени на рынке ЭТ, характеризуется явным лидирующим положением фирмы Microsoft: 80 % всех пользователей ЭТ предпочитают Excel, на втором месте по объему продаж — Lotus 1-2-3, далее следует Quattro Pro.

4.4.2. Содержание электронных таблиц

Итак, комплекс программных средств, реализующий создание, регистрацию, редактирование, хранение и обработку электронных таблиц и выдачу их на печать, называется **табличным процессором**. *Электронная таблица* — это двухмерный массив строк и столбцов, размещенный в памяти ЭВМ. Для Windows был создан процессор Excel, технология работы с которым аналогична работе с любым приложением Windows интерфейса WIMP.

Для обозначения строк в электронных таблицах используется цифровая нумерация, для обозначения столбцов — буквенные индексы. Количество строк и столбцов в разных ЭТ различно. Например, в табличном процессоре Excel 256 столбцов и до 65 536 строк. Место пересечения строки и столбца называется **ячейкой**. Каждая ячейка имеет свой уникальный идентификатор (адрес), состоящий из имени столбца и номера строки, например A28, B97 и т. п. ЭТ могут содержать несколько рабочих бланков, которые объединяются в один файл и носят название **рабочей книги**. В рабочую книгу можно поместить несколько различных типов документов, например рабочий лист с ЭТ, лист диаграмм, лист макросов и т. п.

Адресом (идентификатором) ячейки служат буква, указывающая столбец, и цифра, указывающая номер строки. И то и другое отображено на рабочем листе. Существует абсолютная и относительная адресация ячеек. Абсолютная адресация применяется более широко. При относительной адресации в верхней строке состояния указывается приращение со знаком от начала искомой клетки.

Ширина столбца и высота строки даются по умолчанию. Однако имеется возможность форматирования ячейки, столбца, строки, листа. При этом можно изменить стиль текста, что позволяет улучшить внешний вид документа без применения текстового редактора.

Данные в виде чисел, текста или формул вводятся в ту ячейку, которая отмечена текстовым курсором. Для указания блока ячеек достаточно указать адрес левой верхней ячейки диагонали блока, поставить точку, двоеточие или адрес нижней правой ячейки диагонали. Блок можно задать выделением.

Редактирование таблиц позволяет копировать, удалять, очищать ячейку, блок, лист и выполнять многие другие функции, перечисленные в меню действий *Правка* и *Вставка*. Посредством технологии OLE (Object Linker and Embedding) в таблицу можно вставить рисунок, график, диаграмму, любой объект, представленный другой программой.

Большинство электронных таблиц имеют средства создания, редактирования и включения в нужное место листа графиков и диаграмм, а также большое число встроенных функций: математических, статистических и др. Это существенно облегчает процесс вычислений и расширяет диапазон применения ЭТ. Пользователю предоставляется возможность переопределить панель инструментов, вид рабочего листа, изменить масштабирование, включить полосы прокрутки, переключатели, меню. Сервисные функции табличного процессора Excel позволяют проверить орфографию текста, защитить данные от чтения или записи, создать диалоговые окна или обратиться к динамическим библиотекам Windows.

В табличном процессоре Excel есть средство создании макросов — Visual Basic — объектно-ориентированный язык программирования. Отличие его, например, от Pascal или C++ в том, что в Visual Basic нет возможности создавать новые типы

объектов или порождать потомки уже существующих. Однако пользователь получает большой набор готовых объектов: рабочие книги, листы, ячейки, диаграммы и т. д.

Все табличные процессоры позволяют создавать БД и предоставляют учебные средства работы с ними.

В Microsoft Excel имеется один тип файла — *рабочая книга*, состоящая из рабочих листов, листов диаграмм и макросов, но при этом все листы подшиты в рабочую книгу. Такой подход упрощает работу с несколькими документами за счет быстрого доступа к каждому листу через ярлычки в нижней части листа; позволяет работать с листами, объединенными в группу, например группу учетных карточек на товар. Более того, если производится группа действий на одном листе, эти действия автоматически повторяются на всех листах группы, что упрощает оформление однотипных по структуре листов. Объемные ссылки позволяют создавать сводные документы на основе данных из нескольких листов без ввода громоздких формул с внешними ссылками. Микротехнология *Мастер сводных таблиц* позволяет выбрать нужные данные из документа, представить их сводной таблицей, изменяя структуру, внешний вид, добавляя итоговые строки, группируя и сортируя. В рабочую книгу можно включать информацию о теме, авторе, ключевых словах. Ее можно использовать при поиске файла на диске или при выяснении его назначения.

В процессоре Excel можно использовать многооконную систему, позволяющую выполнять параллельные действия. Все объекты, созданные пользователем (сформированные таблицы, макросы, выборки из БД, диаграммы и графики), можно сохранить в виде файла или распечатать.

4.4.3. Работа с электронными таблицами

В электронной таблице можно работать как с отдельными ячейками, так и с группой ячеек, которые образуют блок. Имена ячеек в блоках разделяют двоеточием («:»), например блок A1:B4 включает в себя ячейки A1, A2, A3, A4, B1, B2, B3, B4. С блоками ячеек в основном выполняются операции копирования, удаления, перемещения, вставки и т. д. Адреса исполь-

зуются в формулах как ссылки на определенные ячейки. Таким образом, введенные один раз значения можно многократно и в любом месте таблицы использовать без повторного набора. Соответственно при изменении значения ячейки автоматически произойдут изменения в тех формулах, в которых содержатся ссылки на данную ячейку.

Технология работы с табличным документом аналогична процедурам подготовки текстовых документов: редактируемый отчет в виде таблицы выводится на экран, и пользователь может в диалоговом режиме вносить в него свои изменения (т. е. редактировать содержимое клеток ЭТ). Все внесенные изменения сразу же отображаются на экране компьютера. Вид окна программы Excel показан на рис. 4.2.

В клетки ЭТ могут быть введены текст, цифры и формулы. Во всех табличных процессорах существуют синтаксические соглашения, позволяющие отличить формально-цифровую информацию от текстовой, которых должен придерживаться пользователь, если хочет добиться правильных результатов. Обычно синтаксические правила интуитивно понятны и легко запоминаются (например, задание формулы начинается со знака «=» и т. п.). **Формула** — это выражение, состоящее из числовых величин и арифметических операций. Кроме числовых величин, в формулу могут входить в качестве документов адреса ячеек, функции и другие формулы. Пример формулы: =A5/H8*12. В ячейке, в которой находится формула, виден только результат вычислений. Саму формулу можно увидеть в строке ввода, когда данная ячейка станет активной.

Функции представляют собой запрограммированные формулы, позволяющие производить часто встречающиеся последовательности вычислений. Например, функция автосуммирования может быть представлена следующим образом: =СУММ (A1:A4).

В Microsoft Excel можно работать с четырьмя основными типами документов: электронной таблицей (в Excel она называется *рабочим листом*), рабочей книгой, диаграммой, макротаблицей.

Рабочий лист служит для организации и анализа данных. Одновременно на нескольких листах данные можно вводить, править, производить с ними вычисления.

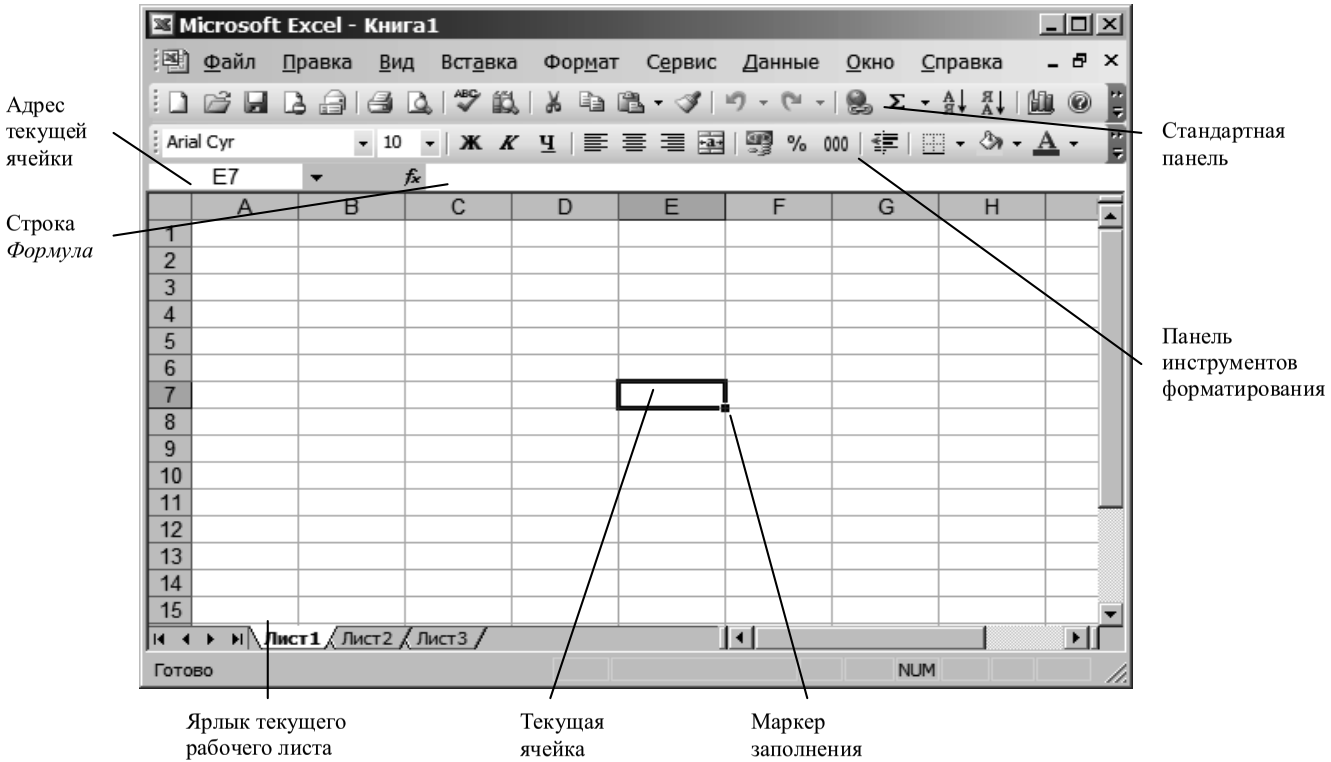


Рис. 4.2. Окно программы Excel

Рабочая книга представляет собой электронный эквивалент папки-скоросшивателя. Она состоит из листов, имена которых выводятся на ярлычках в нижней части экрана. По умолчанию книга, как правило, открывается тремя рабочими листами (Лист 1, Лист 2, Лист 3), однако их число можно увеличить или уменьшить. В книгу можно поместить несколько различных типов документов: листы с ЭТ, листы диаграмм для графического представления данных и модули для создания и хранения макросов, используемых при хранении специальных задач, и т. п.

Диаграмма представляет собой графическое изображение связей между числами ЭТ. Она позволяет показать количественное соотношение между сопоставляемыми величинами.

Макротаблица (макрос) — это последовательность команд, которую приходится постоянно выполнять пользователю в повседневной работе. Макросы позволяют автоматизировать часто встречающиеся операции.

4.4.4. Процесс проектирования электронных таблиц

Так как любая ЭТ состоит из заголовка таблицы, заголовков столбцов (шапки таблицы), информационной части (исходных и выходных данных, расположенных в соответствующих ячейках), процесс проектировании электронных таблиц состоит из следующих этапов:

— формирования заголовка ЭТ;

— ввода названий граф документа;

— ввода исходных данных;

— ввода расчетных формул;

— форматирования ЭТ с целью придания ей профессионального вида;

— подготовки к печати и печать ЭТ.

При необходимости ЭТ могут сопровождаться различными пояснительными комментариями и диаграммами. Excel предоставляет большой набор возможностей по графическому представлению данных. Так, возможно выбирать из различных типов диаграмм, причем каждый тип имеет несколько разновидностей (подтипов).

Диаграммы можно строить либо на рабочем бланке таблицы, либо на новом рабочем бланке. В Excel с помощью *Масте-*

ра диаграмм, вызов которого осуществляется с панели инструментов диаграмм, можно создать диаграмму по шагам, просмотреть любой тип диаграмм и выбрать оптимальный для данной таблицы. Включенная в рабочий бланк диаграмма может находиться в одном из трех режимов:

— просмотра, когда диаграмма выделена по периметру прямоугольником;

— перемещения, изменения размера и удаления, когда диаграмма по периметру выделена прямоугольником с маленьким квадратиком;

— редактирования, когда диаграмма выделена по периметру синим цветом или синим цветом выделен ее заголовок.

Представление данных в виде диаграмм позволяет наглядно представить числовые данные и осуществить их анализ по нескольким направлениям.

Тема 4.5

ТЕХНОЛОГИЯ ХРАНЕНИЯ, ПОИСКА И СОРТИРОВКИ ИНФОРМАЦИИ

4.5.1. Способы организации баз данных

В настоящее время в нашей стране накоплен большой опыт разработки автоматизированных систем управления (АСУ). Этот опыт говорит о том, что центральным техническим вопросом разработки АСУ и любых других информационных систем является вопрос организации, хранения и комплексного использования данных. В конечном счете это привело к созданию развитых средств управления данными, которые служат основой любой информационной системы, построенной на базе использования средств вычислительной техники.

АСУ, спроектированные на основе концепции банков данных, обладают целым рядом характерных свойств, которые выгодно отличают их от предшествующих разработок, основой которых была система массивов данных, ориентированная на решение комплекса установившихся задач. Использование автоматизированных банков данных позволяет обеспечить многоаспектный доступ к совокупности взаимосвязанных данных,

достаточно высокую степень независимости прикладных программ от изменений логической и физической организации данных, интеграцию и централизацию управления данными, устранение излишней избыточности данных, возможность совмещения пакетов и телепроцессорной обработки данных. Поэтому разработка АСУ для любой сферы применения связана прежде всего с созданием автоматизированных банков данных.

Так как основой любого управлении является информация о состоянии объекта, то именно поэтому центром АСУ являются данные, их организация, тщательное ведение, хранение и использование. Меняются техника, программное хозяйство, но данные остаются. Работа с ними — дело достаточно дорогое, поэтому ученые задумались над системными принципами организации, положенными в основу создания банков данных.

Под **автоматизированным банком данных** понимается организационно-техническая система, представляющая собой совокупность баз данных пользователей, технических и программных средств формирования и ведения этих баз и коллектива специалистов, обеспечивающих функционирование системы.

Рис. 4.3. Составные части банка данных

Составными частями любого банка данных являются база данных, система управления базой данных (СУБД), администратор базы данных, прикладное программное обеспечение (рис. 4.3).

Функционирование СУБД основано на введении двух уровней организации базы данных — логического и физического, соответствующих двум аспектам организации данных — логическому (с точки зрения использования данных в приложениях) и физическому (с точки зрения хранения данных в памяти ЭВМ).

Описание логической организации базы данных определяет взгляд пользователей на организацию данных в системе, которые отображают состояние некоторой предметной области. Необходимо отметить, что в общем случае структуры физиче-

ской и логической организации данных могут не совпадать. Формальное описание логической организации данных иногда называют **моделью данных** или **схемой**.

Говоря о физической организации, необходимо отметить, что существует много различных способов организации данных в запоминающей среде, с помощью которых можно обеспечить соответствие некоторой модели.

Наиболее общее представление о базе данных можно сформулировать так: **база данных** (data base) представляет собой совокупность хранимых во внешней памяти ЭВМ больших объемов данных и является интегрированной, т. е. представляет собой комплекс взаимосвязанных данных, предназначенных для обеспечения информационных нужд различных пользователей, каждый из которых имеет отношение к отдельным (возможно, совместно используемым) частям данных; работа с БД может осуществляться либо в пакетном режиме, либо с удаленных терминалов в режиме реального времени.

Таким образом, БД — это совокупность хранимых в памяти ЭВМ и специальным образом организованных взаимосвязанных данных, отображающих состояние предметной области. БД также предназначена для обеспечения информационных нужд определенных пользователей.

На одной ЭВМ, как правило, создается несколько различных БД. Со временем те БД, которые предназначены для выполнения родственных функций, могут объединиться, если это будет способствовать повышению производительности всего вычислительного комплекса.

БД хранит информацию об объектах реального мира и отношениях между этими объектами через данные и связи между данными. Прежде чем говорить о размещении данных и связей между ними на устройствах памяти, необходимо представить взаимосвязь данных на логическом уровне, создав своеобразную модель данных.

Эта модель данных должна быть четко определена, для чего в системе управления базами данных необходимо представить средства (язык) для ее описания. Основное назначение *модели данных* состоит в том, чтобы представить информационную картину в целом, без отвлекающих деталей, связанных с особенностями хранения. Модель данных — это инструмент, с по-

мощью которого разрабатывается стратегия получения любых данных, хранящихся в базе.

Рассмотрим иерархическую, сетевую и реляционную модели данных. Можно считать, что в некотором смысле эта последовательность отражает прогресс в понимании проблем обработки данных. Однако начнем рассмотрение все-таки с сетевой модели, так она наиболее наглядно отражает отношения объектов проблемной области.

Сетевая модель данных, или **сетевая БД** (от англ. network data base). Отношения объектов реального мира всегда могут быть представлены в виде некоторой сети. Это представление рисует довольно наглядную картину реальной действительности и, кроме того, претендует на то, что может быть естественным образом отражено в долговременной памяти вычислительной (информационной) системы. Пример сетевой модели данных приведен на рис. 4.4.

Каждый узел сети соответствует элементу данных, отображающему группу однородных объектов реального мира. Тогда в реальной сети или, как говорят, экземпляре сети, в каждом узле будет находиться идентификатор соответствующего объекта, например шифр детали.

На самом деле представлять и рисовать сеть на таком уровне детализации достаточно трудно. Потому в вершинах сети обычно стоят целые записи, состоящие из совокупности идентификаторов. Например, в вершине сети, имеющей имя *деталь*,

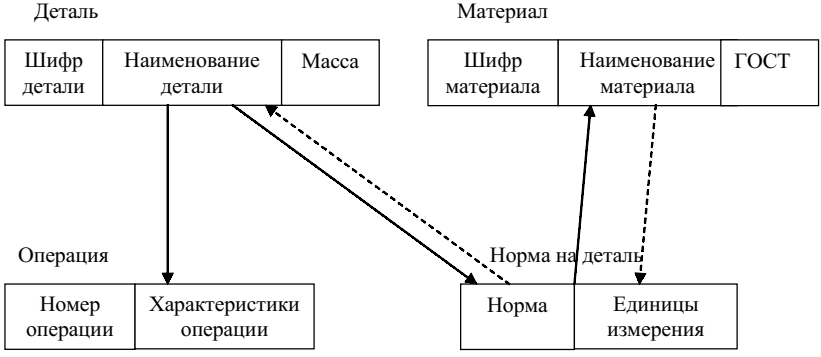


Рис. 4.4. Пример сетевой модели данных

фактически стоит запись (сегмент), состоящая из полей *шифр детали*, *наименование детали* и *масса детали*. На каком основании объединять различные элементы данных в записи (сегменты) — это предмет особого разговора.

Например, 11591 — табельный номер служащего — является числовым идентификатором. Этот идентификатор не описывает свойства, их приходится задавать дополнительно.

Более полно объект описывается записью об объекте, которая обычно состоит из идентификатора объекта — знака, позволяющего отличить один объект от другого среди однородных объектов, и идентификаторов (значений) свойств (атрибутов). Например, запись о служащем некоторой организации имеет табельный номер служащего в качестве идентификатора и такие элементы данных, как должность, заработная плата, льготы и т. д., рассматриваемые как идентификаторы (значения) свойств служащего.

Следует подчеркнуть, что понятия «объект» и «свойства» относительны. Если речь идет о служащем, то естественно понимать должность как свойство служащего. Но если речь идет о должности, например в смысле должностных инструкций, то уже сама должность выступает в качестве объекта, который может иметь свойства. В частности, в определенном контексте табельный номер служащего может рассматриваться как свойство должности.

Таким образом, сетевая БД — это такая база данных, которая организована в виде сети.

Иерархическая модель данных. Более удобного представления данных можно достигнуть за счет увеличения информационной избыточности на уровне модели данных. Шагом на пути к увеличению информационной избыточности является переход от сетевого к иерархическому представлению данных, тем более что сеть можно представить в виде совокупности деревьев. Для этого в иерархических структурах требуется повторить и несколько преобразовать некоторые вершины сети.

Иерархия (от греч. hierós — священный, arché — власть) — это структура объектов, при которой строго выражены их уровни. Объекты нижнего уровня подчиняются объектам верхнего уровня, если между ними есть связи. Иерархия может быть представлена деревом. *Дерево* — это *структура*, напоминаю-

щая дерево. В ее основании имеется один элемент (корень), связанный с несколькими элементами (стволами), которые, в свою очередь, связаны с еще несколькими (ветками), и т. д. до последних элементов (листьев). Получается многоуровневое дерево. **Дерево** — это связанный неориентированный граф без циклов.

Иерархическая база данных — это БД, элементы которой организованы на основе иерархического принципа.

Под **избыточностью** понимаются включения в устройство, языки, данные дополнительных элементов, без которых его работа в принципе возможна, но вместе с тем их наличие повышает надежность обработки, упрощает, а иногда и усложняет взаимодействие этих элементов.

Если на уровне сети, не вдаваясь в подробности физического хранения данных, БД представляется в виде сложной объемной паутины, то на уровне иерархической модели БД представляется в виде совокупности отдельных древовидных структур, в корнях которых стоят идентификаторы объектов, а на последующих ярусах раскрываются свойства этих объектов.

Таким образом, по крайней мере на уровне логического представления, некоторые отношения и связи реального мира имитируются с помощью информационной избыточности. Это еще не значит, что информационная избыточность будет реальной, т. е. будет иметь место и на уровне физического отображения на долговременную память. Такое видение мира более удобно с точки зрения обработки данных.

Следовательно, сетевые модели данных могут быть разложены на иерархические структуры. Однако следует отметить, что модель данных, представленная как совокупность нескольких деревьев, не обладает наглядностью, так как не создается впечатления некоторой взаимосвязанной системы данных. Вообще говоря, одновременно с использованием деревьев для описания модели данных неплохо представлять себе модель данных в виде сети, с тем чтобы ясно понимать, как с ней взаимодействуют иерархические структуры.

Сетевая модель, если она не очень громоздка, позволяет графически представить взаимодействие объектов, отображенных в БД. Это всегда полезно, даже если затем для описания используются иные структуры.

Реляционная модель данных. Наиболее абстрактной моделью является реляционная модель данных. Абстрактна она в том смысле, что в значительной степени ориентирована на интересы пользователя (программиста) и совершенно не несет в себе черт реального отображения на физическую память. Эта модель исторически возникла позже других, и ее появление оправдывается тем, что по мере усложнения информационных систем и прогресса в устройствах долговременной памяти поддержание математического обеспечения, несущего в себе черты «суеты представления», обходится очень дорого.

Реляционная модель получается путем дальнейшей формализации иерархической модели. В этой модели все связи между объектами задаются путем явной фиксации идентификаторов объектов в записях.

На первый взгляд реляционная модель может быть представлена в виде однородных таблиц (отношений), которые напоминают стандартные последовательные файлы. Однако это очень упрощенная точка зрения. Дело в том, что последовательный файл предполагает определенную упорядоченность и обработку в соответствии с этой упорядоченностью, или, иными словами, предполагается вход со стороны одного поля (ключа сортировки).

Существенное отличие реляционной модели от обыкновенного последовательного файла заключается в том, что все столбцы в таблице с точки зрения входа предполагаются эквивалентными.

Информационная избыточность, по крайней мере на логическом уровне, максимальная. Однако повторим еще раз: это не предполагает реальной информационной избыточности при отображении данных на физическую память.

На физическом уровне, например, может быть абсолютно не избыточная сеть или сеть с перекрестными связями. Задача системы управления базой данных заключается в том, чтобы перевести запрос, сформулированный в терминах реляционной модели, в последовательность команд, осуществляющих поиск по сети.

Таким образом, реляционная модель предполагает очень высокую степень «интеллектуальности» систем управления базами данных.

4.5.2. Системы управления базами данных

СУБД — составная часть автоматизированного банка данных, обеспечивающая работу прикладных программ с базой данных.

Одним из важнейших назначений СУБД является обеспечение *независимости данных*. Под этим термином понимается независимость данных и использующих их прикладных программ друг от друга в том смысле, что изменение одних не приводит к изменению других. Необходимо также отметить такие возможности СУБД, как обеспечение защиты и секретности данных, восстановление БД после сбоев, ведение учета работы с БД. Однако это далеко не полный перечень того, что должна осуществлять СУБД для обеспечения интерфейса пользователей с БД и жизнеспособности всего автоматизированного банка данных.

Система управления данными имеет набор средств, которые обеспечивают определенные способы доступа к данным. Наиболее общими операциями, которые выполняются средствами СУБД, являются операции поиска, исправления, добавления и удаления данных. Необходимо отметить, что операция поиска — главная среди перечисленных.

Степень реализации принципа независимости данных определяет гибкость СУБД. Учет особенностей обработки данных в какой-либо предметной области позволяет спроектировать специализированные СУБД, ориентированные на применение в АСУ предприятий с дискретным характером производства.

Существуют и универсальные СУБД, используемые для различных приложений. При настройке универсальных СУБД для конкретных приложений они должны обладать соответствующими средствами. Процесс настройки СУБД на конкретную область применения называется **генерацией системы**. К универсальным СУБД относятся, например, системы Base, Paradox, Microsoft Access, MS DOS, UNIX, Windows, Oracle.

4.5.3. Организация поиска данных

Записи логического файла идентифицируются с помощью уникальной группы символов — **ключа**. Обычно ключом является поле или совокупность полей фиксированной длины. В общем случае в качестве ключа может выступать любое поле за-

писи. Каждому значению ключа может соответствовать одна или несколько записей файла. Ключ, каждому значению которого соответствует одна и только одна запись файла, называется **первичным** или **основным**. Логические записи файла могут иметь несколько первичных ключей.

Каждой записи при ее хранении в памяти соответствует вполне определенное место, задаваемое *адресом*. Способ, задающий соответствие между основным ключом записи и ее адресом в памяти, называется **способом адресации**. Основную проблему при адресации файла можно сформулировать следующим образом: как по основному ключу определить местоположение записи с данным ключом? Существует много способов адресации файлов, их можно отнести к трем основным группам: это способы последовательной, индексной и прямой адресации. Используются также различные комбинации перечисленных способов.

Способы *последовательной адресации* характеризуются тем, что логические записи занимают непрерывный участок памяти и порядок их расположения определяется не значением ключа, а номером следования этого ключа в заданной последовательности ключей логической записи. То есть не существует однозначного соответствия между значением одного ключа и адресом памяти.

При *индексной адресации* соответствие между основным ключом записи и ее адресом в памяти задается с помощью таблицы или иерархии таблиц, т. е. индекса.

Способы *прямой адресации* характеризуются наличием некоторого алгоритма преобразования значения ключа записи непосредственно в адрес ее расположения во внешней памяти.

В зависимости от способов адресации файла могут использоваться различные способы локализации (поиска) записи с заданным значением ключа. Но при последовательной адресации поиск записи может осуществляться только путем просмотра (сканирования) файла с проверкой ключей записи.

Существует также много различных способов поиска записей, например блочный и дихотомический. Наиболее простым способом является последовательное сканирование с проверкой ключа каждой записи.

При использовании способов индексной адресации поиск записи осуществляется путем предварительного просмотра ин-

декса. В результате поиска в индексе сокращается объем просматриваемой памяти, занимаемой непосредственно записями файла. Сам индекс и его просмотр также могут быть организованы различными способами. Поиск записей файлов, использующих прямую адресацию, осуществляется в соответствии с выбранным алгоритмом преобразования значения ключа записи в ее адрес.

4.5.4. Технология использования СУБД

Рассмотрим технологию использования современных СУБД на примере СУБД Microsoft Access.

Технология создания БД с помощью типовых инструментальных средств, рассчитанных на массового пользователя-непрограммиста, предоставляется Microsoft Access. Несмотря на ориентированность на конечного пользователя, в Access присутствует язык программирования и имеется возможность интеграции с другими программными средствами Microsoft Office. Access (от англ. access — доступ) — популярная настольная система управления базой данных, рассчитанная на конечного пользователя. В то же время на небольшом предприятии (при объеме данных до 1 Гбайт) с количеством компьютеров не более 10 ресурсов Access вполне достаточно для обслуживания всего делопроизводства вместе со средствами Microsoft Office. Все пользователи могут обращаться к одной БД, установленной на одном компьютере, который может не быть сервером.

Проблемы сохранности и доступа к данным решаются с помощью использования средств зашиты, которые предоставляет Access. Характерными чертами Access являются направленность на конечного пользователя (непрограммиста); сохранение общего подхода, принятого в построении продуктов Microsoft для WINDOWS; массовость использования.

В Access для работы с данными служат процессор баз данных, средства быстрого построения интерфейса (*Конструктор форм и отчетов*), объекты доступа и манипулирования данными (таблицы, формы, запросы, макрокоманды, макросы, модули). Автоматизация рутинных типовых операций выполняется с помощью готовых визуальных средств или макрокоманд, объединяемых в макросы. Таким образом, пользователи Access

могут обратиться к созданию процедур и функций для работы с данными. При этом, если недостает возможностей визуальных готовых средств, обращаются в макрокомандам, а если и их возможностей недостаточно, можно использовать язык программирования. Он позволяет создавать свои массивы, типы данных, функции, приложения. Имеется возможность целиком создать БД с помощью программирования, когда в этом появится необходимость.

Создание новой БД начинается с запуска Access и появления диалогового окна. Выбор опции *Запуск мастера* приводит в окно *Создание*. Далее для создания базы можно использовать шаблоны. Чтобы обратиться к списку шаблонов, необходимо перейти на вкладку *Базы данных*. Создаются БД выбором из определенного списка. При этом возможен выбор таблиц, а в таблицах — выбор нужных полей. После этого пользователь получает БД с таблицами, формами ввода и вывода. В табл. 4.1 приведен список *Мастеров* (программных модулей), имеющихся в Access. Дополнительно к перечисленным возможностям все созданные формы можно редактировать с помощью вспомогательных диалоговых окон. При первом знакомстве с Access такой способ создания БД весьма эффективен.

Таблица 4.1

***Мастера* СУБД Access**

Наименование	Назначение
Мастер баз данных	Создает базы данных из определенного списка (возможен выбор необходимых таблиц и полей), формы и отчеты
Мастер таблиц	Создает таблицы из списка уже готовых, которые можно изменять. Интересен только на начальном этапе использования таблиц, хотя определенный тип задач можно решить, применяя только таблицы, предоставляемые этим мастером
Мастер простых форм	Создает простую форму, в которую выводятся выбранные пользователем поля из таблицы или запроса
Мастер форм с диаграммой	Создает форму с диаграммой, отражающей данные для полей из таблиц и запросов, которые служат источником данных для форм

Окончание табл. 4.1

Наименование	Назначение
Мастер форм со сводной таблицей Microsoft Excel	Создает форму, в которую включен объект «страница Excel» со сводной таблицей
Мастер построения кнопок	Создает кнопки в форме или отчете с выбранными вами свойствами или функциональностью
Мастер построения групп	Создает группу переключателей, которая может содержать множество кнопок, флажков, выключателей
Мастер построения списков	Создает списки на основе полей из таблиц и запросов, SQL-выражений или предопределенного набора значений
Мастер построения комбинированных списков	Создает комбинированные списки на основе полей из таблиц и запросов, SQL-выражений или предопределенного набора значений
Мастер построения подчиненных форм	Создает подчиненную форму, которая может служить аналогом объектов Grid или Browse в СУБД
Мастер создания отчета	Создает отчет, в который выводятся выбранные пользователем поля из таблицы или запрос с возможностями установить сортировку и группировку
Мастер создания наклеек	Создает наклейки, как стандартные, так и иных размеров
Мастер создания отчетов с диаграммой	Позволяет выводить на печать диаграммы, внешний вид которых зависит от данных в таблице или запросе, являющихся источником данных для отчета

Технология *ввода данных* в базу допускает использование таблицы и формы, через которые обеспечивается работа только с одной строкой таблицы. Ввод с помощью формы позволяет располагать поля в нужном порядке, удобном для пользователя. Создание форм пользователь может выполнять сам или с помощью *Мастера*. Этапы создания формы включают выбор полей, внешнего вида, стиля и названия формы.

Работа с БД начинается с создания таблиц. Обращение к режиму *Создать* предоставляет возможность выбора одного из пяти вариантов технологии создания таблицы (табл. 4.2).

Таблица 4.2

Способы создания таблиц в СУБД Access 7.0

Способ создания	Описание
Режим создания	Пользователю предоставляется таблица с 30 полями, куда необходимо ввести данные. После их сохранения Access решает, какой тип данных присвоить каждому полю. Недостаток этого способа — невозможность создать таблицу с полями примечаний
Конструктор таблиц	После выбора этой операции открывается *Конструктор таблиц*, в котором пользователь должен самостоятельно создать поля, выбрать для них типы данных, размеры и, если необходимо, установить свойства полей
Мастер таблиц	Из определенного набора таблиц пользователь может создать таблицу по своему вкусу. Возможно, что некоторые таблицы целиком подойдут для данного приложения, тогда следует их использовать, чтобы побыстрее завершить процесс
Импорт таблиц	Позволяет импортировать данные из таблиц других приложений в БД. Новые таблицы теряют непосредственную связь с другими приложениями. В появившемся диалоговом окне необходимо выбрать тип файла и имя импортируемого файла. Тип файла ODBC позволяет импортировать данные практически любого формата
Связь с таблицами	Очень похоже на предыдущий пункт, но при этом таблица остается в своем формате, т. е. может использоваться несколькими приложениями

Технология запросов к данным базы в большинстве СУБД строится программно, а в Access она выполняется визуально, за исключением сквозных запросов. Пользователь, благодаря Access, реализует разнообразные запросы выборки, при этом они могут модифицировать исходные данные. В этом заложены резервы ускорения работы с данными. Недостатком технологии Access является замедление скорости работы с данными при увеличении таблиц.

Пользователь Access непосредственно участвует в формировании запросов, не прибегая к услугам программистов. Он может направлять запросы в базу для добавления, удаления, обновления и создания таблиц. Запросы можно составить и про-

граммным путем. Одна из сильных сторон технологии Access — фильтры, которые позволяют выбирать информацию с помощью запросов или установки критериев. Создание параметрических запросов дает возможность пользователю вводить значения для отбора данных.

Наряду с формами для каждой таблицы могут быть созданы *отчеты* с помощью средств СУБД или программным путем, что более трудоемко. Для каждой таблицы можно создать *автоотчет* с выводом данных в столбец.

При создании отчета с выбором полей, но без вывода всех имеющихся в таблице или запросе данных Access позволяет обратиться к *Мастеру отчетов,* который, помимо выбора полей, группирует данные по какому-либо полю, устанавливает интервал группировки, порядок сортировки, диаграммы, макет отчета и его стиль. Для построения еще более сложных отчетов используется *Конструктор отчетов*.

Программное создание отчетов используется для построения собственных *мастеров*.

Технология выполнения разнообразных действий и функций с данными базы в среде Access осуществляется макрокомандами, которые объединяются в макросы. Задаваемые параметры придают этим действиям гибкость, которой иначе можно добиться только путем кропотливого программирования. Хотя сами макросы упрощают работу, их создание требует от пользователя затрат труда и времени. В Access имеется около 50 макрокоманд.

Технологии создания баз данных для ПК ориентированы на решение несложных задач с ограниченным объемом информации.

Тема 4.6

МУЛЬТИМЕДИЙНЫЕ ТЕХНОЛОГИИ

4.6.1. Принципы и способы использования мультимедийных технологий

Пакеты программ мультимедиа предназначены для использования в ПЭВМ для отображения и обработки аудио- и видеоинформации. При этом компьютер, помимо программных

средств, должен быть оборудован дополнительными платами, позволяющими осуществлять ввод-вывод аналоговой информации, ее преобразование в цифровую форму.

Программы мультимедиа для ПЭВМ появились сравнительно недавно благодаря значительному росту вычислительных возможностей ПК и большим достижениям в области производства оптических дисков. Дело в том, что при представлении аналоговой информации в цифровом виде требуются огромные объемы памяти: несколько минут видеофильма занимают десятки мегабайт памяти. Естественно, что работа с таким большим файлом возможна лишь при наличии быстродействующего процессора (желательно использовать ПК с RISC-процессором и быстродействующей шиной данных). Кроме того, распространение таких мультимедиа-приложений невозможно на традиционных магнитных дискетах, для этого необходимо использовать оптические компакт-диски (CD-ROM).

Среди мультимедиа-программ можно выделить две небольшие группы. Первая включает пакеты для обучения и досуга. Поставляемые на CD-ROMах емкостью от 200 до 700 Мбайт каждый, они содержат аудиовизуальную информацию по определенной тематике. Разнообразие этих программ огромно, и рынок их постоянно расширяется при одновременном улучшении качества видеоматериалов. Так, созданы и продаются электронные энциклопедии по отраслям знания; электронные «учителя» иностранных языков, бизнеса, политики; деловые и авантюрные игры.

Вторая группа включает программы для подготовки видеоматериалов по созданию мультимедиа-представлений, демонстрационных дисков и стендовых материалов. К пакетам данного вида относятся Director for Windows, Multimedia Viewer Kit, NEC MultiSpin.

HyperCard — первый продуманный и удобный авторский инструмент для работы с Multimedia, поскольку он имеет аппарат ссылок на видео- и аудиоматериалы, цветную графику и текст с его озвучиванием.

Мультимедиа — это интерактивная технология, обеспечивающая работу с неподвижными изображениями, видеоизображением, анимацией, текстом и звуковым рядом. Одним из первых инструментальных средств создания технологии

мультимедиа явилась гипертекстовая технология, которая обеспечивает работу с текстовой информацией, изображением, звуком, речью.

Анимация (от англ. animate — оживлять) — процесс создания движущихся графических изображений на экране дисплея; используется при проектировании различных объектов, моделировании физических явлений, в обучающих системах и игровых программах. В настоящее время разработаны специальные анимационные программные комплексы (пакеты), позволяющие любой объемный объект двигать и вращать в разных направлениях с разными скоростями. С помощью таких пакетов можно создавать мультипликационные фильмы.

Появлению систем мультимедиа способствовал технический прогресс: возросли оперативная и внешняя память, графические возможности ЭВМ, увеличилось качество видеотехники, появились лазерные компакт-диски и др.

Теле-, видео- и большинство аудиоаппаратуры, в отличие от компьютеров, имеют дело с аналоговым сигналом. Поэтому возникла проблема стыковки разнородной аппаратуры с компьютером и управления ими. Для хранения изображения неподвижной картинки на экране с изображением 512×482 точек (пикселей) требуется 250 Кбайт, при низком качестве изображения. Потребовалась разработка программных и аппаратных методов сжатия и развертки данных. Такие устройства и методы были разработаны с коэффициентом сжатия 100 : 1 и 160 : 1. Это позволило на одном компакт-диске разместить около часа полноценного озвученного видео. Наиболее прогрессивными методами сжатия и развертки данных считаются JPEG (Joint Photographic Expert Group — объединенная группа по фотографии; стандарт графических изображений может хранить до 16 млн цветов) и MPEG (Motion Picture Expert Group — экспертная группа по движущимся изображениям; формат графических файлов, в котором видеоизображение сжимается и хранится в определенном файле). Были разработаны звуковые платы (Sound Blaster) и платы мультимедиа, которые аппаратно реализуют алгоритм перевода аналогового сигнала в дискретный. К компакт-дискам было подсоединено постоянное запоминающее устройство CD-ROM.

В 1988 г. Стив Джобс создал принципиально новый тип персонального компьютера — NeXT (сокр. от Extended Technology — расширенная технология), у которого базовые средства систем мультимедиа заложены в архитектуру, аппаратные и программные средства. Были применены новые мощные центральные процессоры 68030 и 68040, процессор обработки сигналов DSP, который обеспечивает обработку звуков и изображений, синтез и распознавание речи, сжатие изображения и работу с цветом. Объем оперативной памяти равнялся 32 Мбайтам; использовались стираемые оптические диски, стандартно встроенные сетевые контроллеры, которые позволяют подключаться к сети; обеспечены методы сжатия, развертки и т. д.; объем памяти «винчестера» — 10,5 Мбайт и 1,4 Гбайт.

Технологии работы с NeXT — новый шаг в общении с машиной. До него работа осуществлялась с интерфейсом SILK (Speech Image Language Knowledge — речь, образ, язык, знания). В состав NeXT входит система электронной мультимедиапочты, позволяющая обмениваться сообщениями типа речи, текста, графической информации и т. д.

Многие операционные системы поддерживают технологию мультимедиа: Windows 2000, DOS 7.0, OS/2. Операционная система Windows 2000 включает аппаратные средства поддержки мультимедиа, что позволяет пользователям воспроизводить оцифрованное видео, аудио, анимационную графику, подключать различные музыкальные синтезаторы и инструменты. В Windows 2000 разработана специальная версия файловой системы для поддержки высококачественного воспроизведения звука, видео и анимации. Файлы с мультимедийной информацией хранятся на CD-ROM, жестком диске или на сетевом сервере. Оцифрованное видео обычно хранится в файлах с расширением AVI (Audio Video Interleave — чередование аудио- и видеоданных; формат записи, при котором чередуются числовые данные с видеоданными и отдельно со звуком), аудиоинформация — в файлах с расширением WAV (WAVeform), аудио в форме интерфейса MIDI (Musical Instrument Digital Interface — цифровой интерфейс музыкальных инструментов) — в файлах с расширением MID. Для их поддержки разработана файловая подсистема, обеспечивающая передачу информации с CD-ROM

с оптимальной скоростью, что существенно при воспроизведении аудио- и видеоинформации.

Даже из такого краткого перечисления возможностей технологии мультимедиа видно, что идет сближение рынка компьютеров, программного обеспечения, потребительских товаров и средств производства того и другого. Наблюдается тенденция развития мультимедиа-акселераторов. *Мультимедиа-акселератор* — программно-аппаратные средства, которые объединяют базовые возможности графических акселераторов с одной или несколькими мультимедийными функциями, требующими обычно установки в компьютер дополнительных устройств. К *мультимедийным функциям* относятся цифровая фильтрация и масштабирование видео, аппаратная цифровая сжатие-развертка видео, ускорение графических операций, связанных с трехмерной графикой (3D), поддержка «живого» видео на мониторе, наличие композитного видеовыхода, вывод TV-сигнала на монитор. *Графический акселератор* также представляет собой программно-аппаратные средства ускорения графических операций: перенос блока данных, закраска объекта, поддержка аппаратного курсора. Происходит развитие микросхемотехники с целью увеличения производительности электронных устройств и минимизации их геометрических размеров. Микросхемы, выполняющие функции компонентов звуковой платы, объединяются на одной большой микросхеме размером со спичечный коробок. И предела этому нет.

К 1990 г. было разработано более 60 пакетов программ с технологией мультимедиа. При этом стандарта не существовало, и в этом же году фирмы IBM и Microsoft одновременно предложили два стандарта: IBM — стандарт Multimedia, а Microsoft — стандарт MPC (Multimedia Personal Computer — мультимедийный персональный компьютер, имеющий звуковую плату и накопитель для CD-ROM-дисков). Остальные фирмы-производители стали разрабатывать пакеты программ на основе этих стандартов. В настоящее время используется стандарт MPC-2. Кроме того, разработаны стандарты на приводы CD-ROM, Sound Bluster— звуковые карты, MIDI-интерфейс — стандарт для подключений различных музыкальных синтезаторов, DCI-интерфейс — интерфейс с дисплейными драйверами, позволяющими воспроизводить полноэкранную видеоинформацию,

MCI-интерфейс — интерфейс для управления различными мультимедийными устройствами, стандарты на графические адаптеры MCI (Media Control Interface — интерфейс управления мультимедиа). Фирма Apple совместно с Fuji Film разработала первый промышленный стандарт IEEEP1394 для создания набора микросхем Fire Ware, позволяющий оснастить цифровым интерфейсом многие потребительские товары, такие, как видеокамеры, для использования их в технологии мультимедиа.

Появление систем мультимедиа произвело революцию в таких областях, как образование, компьютерный тренинг, бизнес и др. Технология мультимедиа создала предпосылки для удовлетворения растущих потребностей общества. Она позволила заменить техноцентрический подход к развитию сфер экономики (планирование индустрии зависит от прогноза возможных технологий) на антропоцентрический (индустрия управляется рынком). Это дает возможность динамически отслеживать индивидуальные запросы мирового рынка, что отражается в тенденции перехода к мелкосерийному производству. Феномен мультимедиа демократизирует научное, художественное и промышленное творчество. Именно авторские технологии обеспечили процесс информатизации общества.

Самое широкое применение технология мультимедиа получила в сфере образования. Как уже говорилось, созданы видеоэнциклопедии по многим школьным предметам, по музеям, городам, маршрутам путешествий, а также игровые ситуационные тренажеры, что сокращает время обучения. Формируется база знаний, в которой сконструированы живые миры; создается диалоговое кино, где потребитель может управлять ходом зрелища с клавиатуры дисплея, посредством реплик, если к компьютеру подключена плата распознавания речи.

Вместе с тем видеоигры могут стать инструмент манипулирования общественным сознанием: негативом здесь является культ насилия.

Технология мультимедиа создает предпосылки для развития «домашней индустрии», что приводит к сокращению производственных площадей, увеличивает производительность труда. Особые перспективы она открывает для дистанционного обучения.

4.6.2. Основные требования к аппаратной части компьютера

Мультимедиа является интеграционной технологией, позволяющей объединить в компьютере практически все виды информационных сообщений: текст, графику, анимацию, аудио- и видеосообщения в полной мере и обеспечить активное воздействие человека на эти данные в реальном масштабе времени. Мультимедиа позволяет получить на компьютере виртуальную реальность.

Чтобы иметь возможность пользоваться мультимедийными системами, необходимо иметь компьютер следующей минимальной конфигурации: процессор с тактовой частотой от 100 МГц и оперативной памятью не менее 256 Мбайт, НЖМД не менее 20 Гбайт, устройство управления компакт-диском с двойной скоростью и буфером не менее 64 Кбайт; 16-разрядную звуковую плату с динамиками и микрофоном.

Тема 4.7

КОМПЬЮТЕРНЫЕ КОММУНИКАЦИИ

4.7.1. Передача информации. Линии связи, их основные компоненты и характеристики

Передача информации осуществляется различными способами. Особенно эффективен в смысле экономии времени способ дистанционной передачи данных по каналам связи. Для его осуществления необходимы специальные технические средства сбора и регистрации информации с датчиков, установленных на рабочих местах, и передачи ее в ЭВМ.

Дистанционно можно передавать как первичную информацию с мест ее возникновения, так и результатную — в обратном направлении. В этом случае результатная информация отражается на дисплеях, табло, печатающем устройстве. Поступление информации по каналам связи в центр ее обработки в основном осуществляется двумя способами: на машинном носителе или непосредственно в ЭВМ с помощью специальных программных и аппаратных средств.

Дистанционная передача информации постоянно развивается и совершенствуется. Особое значение этот способ имеет в многоуровневых межотраслевых системах, где его применение значительно ускоряет прохождение информации с одного уровня управления на другой и сокращает общее время обработки данных.

Появление сетей ЭВМ, по сути, произвело своего рода техническую революцию, так как состоялось объединение технологии сбора, хранения, обработки и передачи информации на ЭВМ с техникой связи. Первые вычислительные сети (ВС) ЭВМ были разработаны в 1960-х гг. в США и СССР и объединяли большие ЭВМ и вычислительные центры (ВЦ). В США это была сеть ARPA, объединявшая 50 университетов и фирм, а в СССР — сеть Академии наук в Ленинграде.

Передача информации между территориально удаленными компонентами подобных распределенных систем осуществляется в основном с помощью стандартных телефонных и телеграфных каналов, а также витых пар проводов и коаксиальных кабелей связи. Современный прогресс в области оптоволоконной техники (использование световодов) позволяет резко повысить пропускную способность линий связи. Так, система F6M обеспечивает передачу информации объемом до 6,3 Мбит/с, заменяя до 96 каналов, а система P400M — передачу информации объемом до 400 Мбит/с, заменяя 5760 телефонных каналов.

Классифицируют коммуникации (каналы передачи данных), обеспечивающие движение информации между элементами сети, по способу связи. В проводных технологиях в качестве физической среды в каналах используются:

— плоский двухжильный кабель;
— витая пара проводов;
— коаксиальный кабель;
— световод.

Беспроводные сетевые технологии, использующие частотные каналы передачи данных (средой является эфир), представляют разумную альтернативу обычным проводным сетям и становятся все более привлекательными. Самое большое преимущество беспроводных технологий — это возможности, предоставляемые пользователям портативных компьютеров. Однако скорость передачи данных, достигаемая в беспровод-

ных технологиях, не может пока сравниться с пропускной способностью кабеля, хотя она в последнее время и значительно выросла (табл. 4.3). Важно, что для перехода к беспроводной технологии не нужно менять уже имеющиеся сети. Аппаратное обеспечение локальных беспроводных сетей теперь может работать с NetWare — семейством сетевых ОС, разработанных компанией Novell Inc, и другими популярными сетевыми операционными системами, а беспроводные рабочие станции можно добавлять к обычной кабельной сети.

Таблица 4.3

Характеристика беспроводных и кабельных технологий

Технология	Протокол	Способ передачи данных	Пропускная способность, Мбит/с	Радиус надежной связи, м
Различные	Ethernet	Кабель	10	—
Различные	Token Ring	Кабель	4 или 16	—
Altair Plus	Ethernet	СВЧ-излучение	3,3	до 43
ARLAN	Ethernet	СВЧ-излучение	1	до 40
Free Port	Ethernet	Инфракрасные лучи	5,7	до 25
InfaLan	Token Ring	Инфракрасные лучи	4	до 25
Raglan	Собственный	Широкополосные радиосигналы	0,23	до 250
WareLan	Собственный	Широкополосные радиосигналы	2	до 250

В спутниковых технологиях физической средой передачи данных является эфир. Использование спутников оправданно в случае значительного удаления абонентов друг от друга при чрезмерном ослаблении посылаемых электромагнитных сигналов с большими посторонними шумами. Чтобы сигналы, направленные отправителем, не смешивались с сигналами к получателю, при работе со спутником прокладываются два частотных канала — для отправителя и получателя. Это позволяет избежать ошибок при передаче информации.

4.7.2. Компьютерные телекоммуникации: назначение, структура, ресурсы

Основным средством компьютерных телекоммуникаций являются вычислительные сети. **Вычислительная сеть (ВС)** — это комплекс ЭВМ, вспомогательного оборудования, каналов связи и специального программного обеспечения для передачи данных между элементами сети. Под сетью обычно понимается структура организации технических средств, данных и программ.

Каналы связи — это технические устройства и физическая среда, обеспечивающие передачу данных. Они бывают аналоговые и цифровые, телефонные и телеграфные, радиочастотные и телевизионные, инфракрасные и оптические, а также выделенные и коммутируемые. Основная характеристика канала связи — его пропускная способность.

Дуплексный канал связи — канал, по которому передача данных происходит в оба направления одновременно.

Симплексный канал — канал, по которому передача данных в каждый момент времени происходит только в одном направлении.

Вычислительные сети позволяют автоматизировать управление производством, транспортом, материально-техническим снабжением в масштабе отдельных регионов и страны в целом.

Возможность концентрации в ВС больших объемов данных, общедоступность этих данных, а также программных и аппаратных средств обработки и высокая надежность их функционирования — все это позволяет улучшить информационное обслуживание пользователей и резко повысить эффективность применения ВТ.

Назначение ВС:

— организовывать параллельную обработку данных многими ЭВМ;

— создавать распределенные БД, размещаемые в памяти личных ЭВМ;

— специализировать отдельные ЭВМ (группы ЭВМ) для эффективного решения определенных классов задач;

— автоматизировать обмен информацией и программами между отдельными ЭВМ и пользователями сети;

— резервировать вычислительные мощности и средства передачи данных с целью быстрого восстановления нормальной работы сети в случае выхода из строя некоторых из них;

— перераспределять вычислительные мощности между пользователями сети в зависимости от изменения их потребностей и сложности решаемых задач;

— стабилизировать и повышать уровень загрузки ЭВМ и дорогостоящего периферийного оборудования;

— сочетать работу в широком диапазоне режимов: диалоговом, пакетном, в режимах «запрос — ответ», сбора, передачи и обмена информацией.

Как показывает практика, за счет расширения возможностей обработки данных, лучшей загрузки ресурсов и повышения надежности функционирования системы в целом стоимость обработки данных в ВС не менее чем в полтора раза ниже, по сравнению с обработкой аналогичных данных на автономных ЭВМ.

Вычислительные и информационные сети классифицируют по различным признакам. Сети, состоящие из программносовместимых ЭВМ, являются *однородными*, или *гомогенными*. Если ЭВМ, входящие в сеть, программно несовместимы, то такая сеть является *неоднородной*, или *гетерогенной*.

По типу организации передачи данных различают сети *с коммутацией каналов, с коммутацией сообщений, с коммутацией пакетов*. Имеются сети, использующие смешанные системы передачи данных.

По характеру реализуемых функций сети подразделяют на:

— *вычислительные,* предназначенные для решения задач управления на основе вычислительной обработки исходной информации;

— *информационные,* предназначенные для получения справочных данных по запросу пользователей (чисто информационные сети используются редко);

— *смешанные,* в которых реализуются вычислительные и информационные функции (в настоящее время это основной вид ВС).

По способу управления вычислительные и информационные сети подразделяют на сети с *децентрализованным, централизованным* и *смешанным управлением*. В первом случае каждая ЭВМ, входящая в состав сети, включает полный

набор программных средств для координации выполняемых сетевых операций. Сети такого типа сложны и достаточно дороги, так как операционные системы отдельных ЭВМ разрабатываются с ориентацией на коллективный доступ к общему полю памяти сети. При этом в каждый конкретный момент времени доступ к общему полю памяти предоставляется только для одной ЭВМ. А координация работы ЭВМ осуществляется под управлением единой операционной системы сети.

В условиях смешанных сетей под централизованным управлением ведется решение задач, обладающих высшим приоритетом и, как правило, связанных с обработкой больших объемов информации.

По структуре построения (топологии) сети подразделяют *одноузловые* и *многоузловые, одноканальные* и *многоканальные*. Топология вычислительной сети во многом определяется структурой сети связи, т. е. способом соединения абонентов друг с другом и ЭВМ. Известны такие структуры сетей: радиальная (звездообразная), кольцевая, многосвязная («каждый с каждым»), иерархическая, «общая шина» и др. (рис. 4.5) (подробное описание этих структур см. в 4.7.3).

Основные функции систем передачи данных в условиях функционирования ВС заключаются в организации быстрой и надежной передачи информации произвольным абонентам сети, а также в сокращении затрат на передачу данных. Последнее особенно важно, так как за прошедшее десятилетие произошло увеличение доли затрат на передачу данных в общей структуре затрат на организацию сетевой обработки информации. Это объясняется главным образом тем, что затраты на техническое обеспечение ВС сократились за этот период примерно в десять раз, тогда как затраты на организацию и эксплуатацию каналов связи сократились только в два раза.

Важнейшая характеристика сетей передачи данных — *время доставки информации* — зависит от структуры сети передачи данных, пропускной способности линий связи, а также от способа соединения каналов связи между взаимодействующими абонентами сети и способа передачи данных по этим каналам. Различают системы передачи данных с постоянным включением каналов связи (некоммутируемые каналы связи) и коммутацией на время передачи информации по этим каналам.

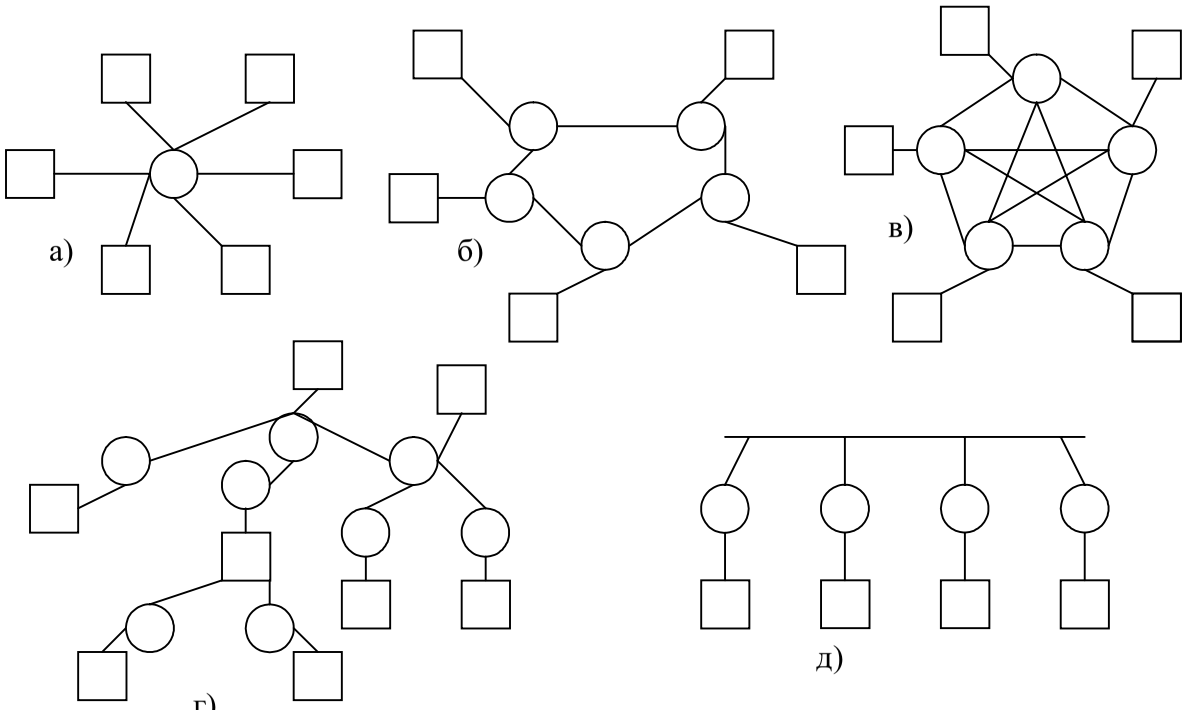


Рис. 4.5. Основные типы структур сетей ЭВМ: *а* — радиальная (звездообразная); *б* — кольцевая; *в* — многосвязная; *г* — иерархическая; *д* — «общая шина»; □ — ЭВМ; ○ — узел коммутации

При использовании *некоммутируемых каналов связи* средства приема-передачи абонентских пунктов и ЭВМ постоянно соединены между собой, т. е. находятся в режиме on-line. В этом случае отсутствуют потери времени на коммутацию, обеспечиваются высокая степень готовности системы к передаче информации, более высокая надежность каналов связи и как следствие — достоверность передачи информации. Недостатками такого способа организации связи являются низкий коэффициент использования аппаратуры передачи данных и линий связи, высокие расходы на эксплуатацию сети. Рентабельность подобных сетей достигается только при условии достаточно полной загрузки этих каналов.

При коммутации абонентских пунктов и ЭВМ только *на время передачи информации* (т. е. нормальным режимом для них является режим off-line) принцип построения узла коммутации определяется способами организации прохождения информации в сетях передачи данных. Существуют три основных способа подготовки и передачи информации в сетях, основанных на коммутации: коммутация каналов, сообщений и пакетов.

Коммутация каналов. Этот способ заключается в установлении физического канала связи для передачи данных непосредственно между абонентами сети. При использовании коммутируемых каналов тракт (путь) передачи данных образуется из самих каналов связи и устройств коммутации, расположенных в узлах связи.

Установление соединения заключается в том, что абонент посылает в канал связи заданный набор символов, прохождение которых по сети через соответствующие узлы коммутации вызывает установку нужного соединения с вызываемым абонентом. Этот транзитный канал образуется в начале сеанса связи, остается фиксированным на период передачи всей информации и разрывается только после завершения передачи информации.

Коммутация каналов как способ соединения используется в основном в сетях, где требуется обеспечить непрерывность передачи сообщений (например, при использовании телефонных каналов связи и абонентского телеграфа). В этом случае связь абонентов возможна только при условии использования ими однотипной аппаратуры, одинаковых каналов связи, а также единых кодов.

Коммутация сообщений. При коммутации сообщений поступающая на узел связи информация передается в память узла связи, после чего анализируется адрес получателя. В зависимости от занятости требуемого канала сообщение либо передается в память соседнего узла, либо становится в очередь для последующей передачи. Таким образом, способ коммутации сообщений обеспечивает поэтапный характер передачи информации.

Способ коммутации сообщений обеспечивает независимость работы отдельных участков сети, что значительно повышает эффективность использования каналов связи при передаче одного и того же объема информации (в этом случае он может достигать 80—90 % от максимального значения). В системе с коммутацией сообщений происходит сглаживание несогласованности в пропускной способности каналов и более эффективно реализуется передача многоадресных сообщений (так как не требуется одновременного освобождения всех каналов между узлом-передатчиком и узлом-приемником). Передача информации может производиться в любое время, так как прямая связь абонентов друг с другом необязательна.

Коммутация пакетов. В последние годы появился еще один способ коммутации абонентов сети — так называемая коммутация пакетов. Этот способ сочетает в себе ряд преимуществ способов коммутации каналов и коммутации сообщений. При коммутации пакетов перед началом передачи сообщение разбивается на короткие пакеты фиксированной длины, которые затем передаются по сети. В пункте назначения эти пакеты вновь объединяются в первоначальное сообщение, а так как их длительное хранение в запоминающем устройстве узла связи не предполагается, пакеты передаются от узла к узлу с минимальной задержкой во времени. В этом отношении указанный способ близок к способу коммутации каналов.

При коммутации пакетов их фиксированная длина обеспечивает эффективность обработки пакетов, предотвращает блокировку линий связей и значительно уменьшает емкость требуемой промежуточной памяти узлов связи. Кроме того, сокращается время задержки при передаче информации, т. е. скорость передачи информации превышает аналогичную скорость при способе коммутации сообщений.

К недостаткам способа коммутации пакетов следует отнести односторонний характер связи между абонентами сети.

Сопряжение ЭВМ и устройств в сетях. Существенное влияние на организацию систем обработки данных оказывают технические возможности средств, используемых для сопряжения (комплексирования) ЭВМ и других устройств. Основным элементом сопряжения является *интерфейс,* определяющий число линий, используемых для передачи сигналов и данных, а также способ (алгоритм) передачи информации по линиям связи.

Все интерфейсы, используемые в ВТ и сетях, подразделяют на три вида: параллельные, последовательные, связные.

Параллельный интерфейс состоит из большого числа линий, по которым передача данных осуществляется в параллельном коде (обычно в виде 8—128-разрядных слов), и обладает большой пропускной способностью: порядка 10^4—10^5 бит/с. Столь большие скорости передачи данных обеспечиваются за счет ограниченной длины интерфейса: обычно — от нескольких метров до десятков (очень редко — до ста) метров.

Последовательный интерфейс состоит, как правило, из одной линии, данные по которой передаются в последовательном коде. Пропускная способность последовательного интерфейса составляет 10^3—10^4 бит/с при длине линии интерфейса от десятков метров до километра.

Связные интерфейсы содержат каналы связи, работа которых обеспечивается аппаратурой, повышающей (в основном с помощью специальных физических методов) достоверность передачи данных. Связные интерфейсы обеспечивают передачу данных на любые расстояния, однако с небольшой скоростью (в пределах 10^2—10^3 бит/с). Их применение экономически целесообразно на расстоянии не менее километра.

В многопроцессорных и многомашинных вычислительных системах используются в основном параллельные интерфейсы для сопряжения отдельных устройств в ЭВМ и только в отдельных случаях применяются последовательные интерфейсы для подключения периферийных устройств. Параллельные интерфейсы обеспечивают в первую очередь передачу сигналов прерывания, а также отдельных слов (команд) и блоков данных между сопрягаемыми ЭВМ и устройствами.

В распределенных системах из-за значительных расстояний между устройствами применяются последовательные и связные интерфейсы, которые исключают возможность передачи отдельных сигналов прерывания между сопрягаемыми устройствами и требуют представления информации в виде операций ввода-вывода.

От организации интерфейсов между устройствами во многом зависит организация программного обеспечения. ПО ВС обеспечивает организацию коллективного доступа к вычислительным и информационным ресурсам сети, динамическое распределение и перераспределение ресурсов сети с целью повышения оперативности обработки информации и максимальной загрузки аппаратных средств, а также в случае отказа и выхода из строя отдельных технических средств и т. д.

Программное обеспечение ВС включает три компонента:

— *общее программное обеспечение,* образуемое базовым ПО отдельных ЭВМ, входящих в состав сети;

— *специальное программное обеспечение,* образуемое прикладными программными средствами, отражающими специфику предметной области пользователей при реализации задач управления;

— *системное сетевое программное обеспечение,* представляющее собой комплекс программных средств, поддерживающих и координирующих взаимодействие всех ресурсов вычислительной сети как единой системы.

Особая роль в программном обеспечении вычислительных сетей отводится системному сетевому ПО, функции которого реализуются в виде распределенной операционной системы сети, которая включает в себя набор управляющих и обслуживающих программ.

4.7.3. Локальные компьютерные сети

Прогресс в развитии микропроцессорной техники сделал ее доступной массовому потребителю, а высокая надежность, относительно низкая стоимость и простота общения с пользователем — не профессионалом в области вычислительной техники послужили основой для организации систем распределенной обработки данных, включающих от десятка до сотен ПЭВМ,

объединенных в вычислительные сети. В отличие от вычислительных сетей, создаваемых на базе больших ЭВМ и охватывающих значительную территорию, сети на базе ПЭВМ получили название **локальных**, так как они ориентированы в первую очередь на объединение вычислительных машин и периферийных устройств, сосредоточенных на небольшом пространстве (например, в пределах одного помещения, здания, группы близлежащих зданий, расположенных на расстоянии нескольких километров друг от друга). Появление локальных вычислительных сетей (ЛВС) позволило значительно повысить эффективность применения ВТ за счет более рационального использования аппаратных, программных и информационных ресурсов вычислительной системы, значительного улучшения эксплуатационных характеристик (в первую очередь повышения надежности) и создания максимальных удобств для работы конечных пользователей.

Сравнительно низкая стоимость, высокая живучесть, простота комплексирования и эксплуатации, оснащенность современными операционными системами различного назначения, высокоскоростными средствами передачи данных, оперативной и внешней памятью большой емкости — все это способствовало быстрому распространению и широкому применению ЛВС для автоматизации управленческой деятельности в учреждениях, на предприятиях, а также для создания на их основе информационных, измерительных и управляющих систем автоматизации технологических и производственных процессов. Одной из главных проблем создания ЛВС является проблема аппаратной совместимости ВТ. В настоящее время вычислительные средства ЛВС в основном объединяются с помощью высоко- и низкоскоростных каналов передачи данных. Такие вычислительные сети получили название **свободносвязанных**, так как протекание вычислительных процессов в них может осуществляться асинхронно.

При незначительной удаленности вычислительного оборудования наиболее эффективным средством связи между отдельными аппаратными компонентами ЛВС является последовательный интерфейс. Его достаточно высокая пропускная способность позволяет иметь единственный канал передачи данных — моноканал; при этом работа всей системы осуществляется в режиме мультиплексирования.

Классификация ЛВС. Все множество видов ЛВС можно подразделить на четыре группы.

К *первой группе* относятся ЛВС, ориентированные на массового пользователя. Такие ЛВС объединяют в основном персональные ЭВМ с помощью систем передачи данных, имеющих низкую стоимость и обеспечивающих передачу информации на расстояние 100—500 м со скоростью 2400—19 200 бит/с.

Ко *второй группе* относятся ЛВС, объединяющие, кроме ПЭВМ, микропроцессорную технику, встроенную в технологическое оборудование (средства автоматизации проектирования, обработки документальной информации, кассовые аппараты и т. д.), а также средства электронной почты. Система передачи данных обеспечивает передачу информации на расстояние до 1 км со скоростью от 19 200 бит/с до 1 Мбит/с. Стоимость передачи данных в таких сетях примерно на 30 % превышает стоимость аналогичных работ в сетях первой группы.

К *третьей группе* относятся ЛВС, объединяющие ПЭВМ, мини-ЭВМ и ЭВМ среднего класса. Эти ЛВС используются для организации управления сложными производственными процессами с применением робототехнических комплексов и гибких автоматизированных модулей, а также для создания крупных систем автоматизированного проектирования, управления научными исследованиями и т. п. Системы передачи данных ЛВС третьей группы имеют среднюю стоимость и обеспечивают передачу информации на расстояние до нескольких километров со скоростью 120 Мбит/с.

ЛВС *четвертой группы* объединяют в своем составе все классы ЭВМ. Они применяются в сложных системах управления крупным производством и даже отдельной отраслью и включают в себя основные элементы всех трех описанных выше групп ЛВС. В рамках этой группы ЛВС могут применяться различные системы передачи данных, в том числе обеспечивающие передачу информации со скоростью от 10 до 50 Мбит/с на расстояние до 10 км. По своим функциональным возможностям ЛВС четвертой группы мало чем отличаются от региональных вычислительных сетей, обслуживающих крупные города, районы, области. В своем составе они могут содержать разветвленную сеть соединений между различными абонентами — отправителями и получателями информации.

По топологическим признакам ЛВС подразделяют на сети с общей шиной, кольцевые, иерархические, радиальные и многосвязные.

В ЛВС с *общей шиной* (рис. 4.5, *д*) одна из машин служит в качестве системного обслуживающего устройства, обеспечивающего централизованный доступ к общим файлам и базам данных, печатающим устройствам и другим вычислительным ресурсам. ЛВС данного типа приобрели большую популярность благодаря низкой стоимости, высокой гибкости и скорости передачи данных, легкости расширения сети (подключение новых абонентов к сети не сказывается на ее основных характеристиках). К недостаткам шинной топологии следует отнести необходимость использования довольно сложных протоколов и уязвимость в отношении физических повреждений кабеля.

Кольцевая топология (рис. 4.5, *б*) характеризуется тем, что информация по кольцу может передаваться только в одном направлении и все подключенные ПЭВМ могут участвовать в ее приеме и передаче. При этом абонент-получатель должен пометить полученную информацию специальным маркером, иначе могут появиться «заблудившиеся» данные, мешающие нормальной работе сети.

Как последовательная конфигурация кольцо особенно уязвимо в отношении отказов: выход из строя какого-либо сегмента кабеля приводит к прекращению обслуживания всех пользователей. Разработчики ЛВС приложили немало усилий, чтобы справиться с этой проблемой. Защита от повреждений или отказов обеспечивается либо замыканием кольца на обратный (дублирующий) путь, либо переключением на запасное кольцо. И в том и в другом случае сохраняется общая кольцевая топология.

Иерархическая ЛВС (конфигурация типа «дерево») представляет собой более развитый вариант структуры ЛВС, построенной на основе общей шины (рис. 4.5, *г*). «Дерево» образуется путем соединения нескольких шин с корневой системой, где размещаются самые важные компоненты ЛВС. Оно обладает необходимой гибкостью для того, чтобы охватить средствами ЛВС несколько этажей в здании или несколько зданий на одной территории, и реализуется, как правило, в сложных системах, насчитывающих десятки и даже сотни абонентов.

Радиальную (звездообразную) конфигурацию (рис. 4.5, *а*) можно рассматривать как дальнейшее развитие структуры «дерево с корнем» с ответвлением к каждому подключенному устройству. В центре сети обычно размещается коммутирующее устройство, обеспечивающее жизнеспособность системы. ЛВС подобной конфигурации наиболее часто находят применение в автоматизированных учрежденческих системах управления, использующих центральную базу данных. Звездообразные ЛВС, как правило, менее надежны, чем сети с общей шиной или иерархические, но эта проблема решается дублированием аппаратуры центрального узла. К недостаткам можно отнести и значительное потребление кабеля (иногда в несколько раз превышающее расход в аналогичных по возможностям ЛВС с общей шиной или иерархических).

Наиболее сложной и дорогой является *многосвязная топология,* в которой каждый узел связан со всеми другими узлами сети (рис. 4.5, *в*). Эта топология в ЛВС применяется очень редко, в основном там, где требуются исключительно высокие надежность сети и скорость передачи данных.

На практике чаще встречаются *гибридные* ЛВС, приспособленные к требованиям конкретного заказчика и сочетающие фрагменты шинной, звездообразной и других топологий.

Методы доступа в ЛВС. Под **методом доступа** понимается метод организации хранения и обмена данных в устройствах памяти, файлах, базах данных и сетях. По методам доступа выделяются такие наиболее распространенные сети, как Ethernet, ArcNet и Token Ring.

Метод доступа Ethernet (от англ. ether — эфир и net — сеть), пользующийся наибольшей популярностью, обеспечивает высокую скорость передачи данных и надежность. Для Ethernet используется *топология «общая шина»*. Поэтому сообщение, отправляемое одной рабочей станцией, принимается одновременно всеми остальными станциями, подключенными к общей шине. Но поскольку сообщение включает адреса станций отправителя и адресата, то другие станции это сообщение игнорируют. Ethernet — метод множественного доступа. При нем перед началом передачи рабочая станция определяет, свободен канал или занят. Если канал свободен, то станция начинает передачу. Скорость передачи данных в Ethernet — до 10 Мбит/с.

Дальнейшее развитие этот метод имеет в виде Fast Ethernet с еще большей скоростью передачи данных.

Метод доступа ArcNet получил распространение и силу дешевизны оборудования. Он используется в ЛВС со *звездообразной топологией*. Одна из ПЭВМ создает специальный маркер (сообщение специального вида), который последовательно передается от одной ПЭВМ к другой. Если станция передает сообщение другой станции, она должна дождаться маркера и добавить к нему сообщение, дополненное адресами отправителя и назначения. Когда пакет дойдет до станции назначения, сообщение будет отделено от маркера и передано станции.

Метод доступа Token Ring (от англ. token — маркер и ring — кольцо) рассчитан на *кольцевую топологию* и также использует маркер, передаваемый от одной станции к другой. Но при этом имеется возможность назначать разные приоритеты разным рабочим станциям. Маркер перемещается по кольцу, давая последовательно расположенным на нем компьютерам право на передачу. Если компьютер получает пустой маркер, он может заполнить его сообщение кадром любой длины, однако лишь в течение того промежутка времени, который отводит специальный таймер для нахождения маркера в одной точке сети. Кадр перемещается по сети, и каждая ПЭВМ регенерирует его, но только принимающая ПЭВМ копирует этот кадр в свою память и отмечает его как принятый, однако не выводит сам кадр из кольца. Эту функцию выполняет передающий компьютер, когда его сообщение вновь возвращается к нему. Тем самым обеспечивается подтверждение факта передачи сообщения.

Вернемся к вопросу о способах соединения персональных компьютеров в единый вычислительный комплекс. Самый простой из них — соединить компьютеры через последовательные порты. В этом случае имеется возможность копировать файлы с жесткого диска одного компьютера на другой, если воспользовался программой из операционной оболочки Norton Commander. Для получения прямого доступа к жесткому диску другого компьютера разработаны специальные сетевые платы (адаптеры) и программное обеспечение. В простых локальных сетях функции управления выполняются не на серверной основе, а по принципу соединения рабочих станций друг с другом, поэтому пользователю можно не приобретать специаль-

ные файловые серверы и дорогостоящее сетевое ПО. Каждая ПЭВМ такой сети может выполнять функции как рабочей станции, так и сервера.

В ЛВС с развитой архитектурой функции управления выполняет сетевая операционная система, устанавливаемая на более мощном, чем рабочие станции, компьютере (файловом сервере). Серверные сети подразделяют на *сети среднего класса* (до 100 рабочих станций) и *мощные* (корпоративные), объединяющие до 250 рабочих станций и более. Основным разработчиком сетевых программных продуктов для сервера ЛВС является фирма Novell.

Следует отметить, что наблюдается тенденция ускорения передачи данных до гигабитовых скоростей. К тому же требуется передавать данные типа высококачественного звука, речи и изображения. Все это ведет к постепенному вытеснению таких «старых» ЛВС, как Token Ring и ArcNet. Операционная система Windows NT фирмы Microsoft вытесняет с рынка ОС Unix.

В последние годы большой популярностью стали пользоваться виртуальные ЛВС VLAN. Они отличаются от обычных ЛВС тем, что не имеют физических ограничений. Виртуальные ЛВС определяют, какие рабочие станции включаются в физические группы на основе протокольной адресации, что позволяет располагать их в любом месте сети.

Модели взаимодействия в ЛВС. В серверных ЛВС реализованы две модели взаимодействия пользователей с рабочими станциями — «файл-сервер» и «клиент-сервер».

В модели *«файл-сервер»* сервер обеспечивает доступ к файлам базы данных для каждой рабочей станции, и на этом его работа заканчивается. Например, если используется база данных типа «файл-сервер» для получения сведений о налогоплательщиках, проживающих на какой-либо конкретной улице Москвы, по сети будет передана вся таблица по территориальному округу, и решать, какие записи в ней удовлетворяют запросу, а какие — нет, приходится самой рабочей станции. Таким образом, работа модели «файл-сервер» приводит к перегрузке сети.

Модель *«клиент-сервер»* лишена этих недостатков. В этом случае прикладная система делится на две — внешнюю (обращенную к пользователю), называемую *клиентом*, и внутрен-

нюю (обслуживающую), называемую *сервером*. Сервером является машина, обладающая ресурсами и предоставляющая их, а клиентом — потенциальный потребитель этих ресурсов. Роль ресурсов могут играть файловая система (файловый сервер), процессор (вычислительный сервер), база данных (сервер базы данных), принтер (принтер-сервер) и др. Так как сервер (или серверы) обслуживает одновременно многих клиентов, то на серверном компьютере должна функционировать многозадачная операционная система.

В модели «клиент-сервер» сервер играет активную роль, ибо программное обеспечение «заставляет» его «сначала подумать, а потом сделать». Потоки информации, текущие по сети, становятся меньшими, поскольку сервер сначала обрабатывает запросы, а затем посылает клиенту то, в чем он нуждается. Сервер также контролирует допустимость обращения к записям на индивидуальной основе, что обеспечивает бóльшую безопасность данных.

Если вспомнить сети, созданные на основе больших универсальных ЭВМ, то модель «большая ЭВМ-терминал» и есть модель «клиент-сервер». В модели «клиент-сервер», созданной на основе ПЭВМ, предлагается следующее:

— сеть содержит значительное количество серверов и клиентов;

— основу вычислительной системы составляют рабочие станции, каждая из которых функционирует как клиент и запрашивает информацию, которая находится на сервере;

— пользователь системы освобожден от необходимости знать, где находится требуемая ему информация: он просто запрашивает то, что ему нужно;

— система реализуется в виде открытой архитектуры, объединяющей ЭВМ различных классов и типов с различными системами.

4.7.4. Глобальные компьютерные сети

Сетевые технологии в настоящее время чрезвычайно разнообразны. К классификации вычислительных и информационных сетей, рассмотренной в параграфе 4.7.2, добавим класси-

фикацию по такому ключевому признаку, как охват территории. По охвату территории сети подразделяют на локальные, региональные (территориальные), федеральные и глобальные.

Использование персональных компьютеров в составе *локальных ВС* обеспечивает постоянное и оперативное взаимодействие между отдельными пользователями в пределах коммерческой либо научно-производственной структуры. Свое название ЛВС получила за то, что все ее компоненты (ПК, каналы коммуникаций, средства связи) физически размещаются на небольшой территории одной организации или ее отдельных подразделений.

Территориальной (региональной) называют технологию (сеть), компьютеры которой находятся на большом удалении друг от друга: как правило, от десятков до сотен километров. Иногда территориальную сеть называют *корпоративной* или *ведомственной*. Такая сеть обеспечивает обмен данными между имеющими доступ к ресурсам сети абонентами по телефонным каналам сети общего назначения, каналам сети «Телекс», а также спутниковым каналам связи. Количество абонентов сети не ограничено. Им гарантируются надежный обмен данными в режиме реального времени, передача факсов и телефонная связь по спутниковым каналам. Территориальные сети строятся по идеологии открытых систем. Их абонентами являются отдельные ПК, ЛВС, телексные, факсимильные и телефонные установки, сетевые элементы (узлы сети связи).

Основная задача *федеральной сети* — создание магистральной сети передачи данных с коммутацией пакетов и предоставление услуг по передаче данных в реальном масштабе времени широкому кругу пользователей, к числу которых относятся и территориальные сети.

Наконец, *глобальные сети* обеспечивают возможность общения по переписке и телеконференции. Основная задача глобальной сети — обеспечение не только доступа к компьютерным ресурсам, но и возможности взаимодействия между собой различных профессиональных групп, рассредоточенных на большой территории.

В 1997 г. в мире было зарегистрировано более 200 глобальных сетей, 56 из которых созданы в США, 16 — в Японии. Одна из первых глобальных сетей — ARPANet — охватывала всю тер-

риторию США, часть Европы и Азии. Сеть ARPANet доказала техническую возможность и экономическую целесообразность разработки крупных сетей для более эффективного использования ЭВМ и программного обеспечения. В Европе сначала были разработаны и внедрены международные сети, затем — национальные. В 1972 г, в Австрии была создана сеть МИПСА, в 1979 г. к ней подсоединились 17 стран Европы, СССР, США, Канада и Япония. Сеть была создана для проведения фундаментальных исследовательских работ по проблемам энергетики, продовольствия, сельского хозяйства, здравоохранения и др. Кроме того, в рамках МИПСА была внедрена технология, позволяющая всем национальным институтам развивать связь друг с другом.

К созданной в 1960-х гг. в Ленинградском отделении Академии наук СССР сети в 1985 г. подключилась региональная сеть «Северо-Запад» с академическими центрами Москвы и Риги. В 1980 г. была сдана в эксплуатацию система телеобработки статистической информации (СТОСИ), обслуживающая Главный ВЦ ЦСУ СССР в Москве и республиканские ВЦ в союзных республиках.

В течение последнего десятилетия получают все более широкое развитие глобальные вычислительные и информационные сети — уникальный симбиоз компьютеров и коммуникаций. Идет активное включение государств во всемирные сетевые структуры. Мировой системой компьютерных коммуникаций ежедневно пользуются более 30 млн человек. Возрастает потребность в средствах структурирования, накопления, хранения, поиска и передачи информации. Удовлетворению этих потребностей служат информационные сети и их ресурсы. Совместное использование ресурсов сетей (библиотек программ, баз данных, вычислительных мощностей) обеспечивается технологическим комплексом и средствами доступа. Глобальные сети (*Wide Area Network, WAN*) — *это телекоммуникационные структуры, объединяющие локальные информационные сети, имеющие общие протокол связи, методы подключения и протоколы обмена данными*. Каждая из глобальных сетей (Internet, Bitnet, DECnet и др.) организовывалась для определенных целей, а в дальнейшем расширялась за счет подключения локальных сетей, использующих ее услуги и ресурсы.

Крупнейшей глобальной информационной сетью является Internet. В настоящее время в русскоязычной литературе закрепились наименование *Internet* и его синоним — *Сеть*, которые пишутся с прописной буквы. Передача данных в этой Сети организована на основе протокола Internet IP (Internet Protocol), представляющего собой описание работы Сети, которое включает правила налаживания и поддержания связи в Сети, обращения с IP-пакетами и их обработки, описание сетевых пакетов семейства IP. Сеть спроектирована таким образом, что пользователь не имеет никакой информации о конкретной структуре Сети. Чтобы послать сообщение по Сети, компьютер размещает данные в некий «конверт», называемый, например, IP, с указанием конкретного адреса.

Процесс совершенствования Internet идет непрерывно, большинство новаций незаметны для пользователей. Любой желающий может получить доступ к Сети. В России подключение к Internet началось в начале 1990-х гг.

Архитектура сетевых протоколов TCP/IP (Transmission Control Protocol / Internet Protocol), на основе которых построен Internet, предназначена специально для объединенной Сети. Сеть может состоять из совершенно разнородных подсетей, соединенных друг с другом шлюзами. В качестве подсетей могут выступать локальные (Token Ring, Ethernet, пакетные радиосети и т. п.), национальные, региональные и специализированные сети, а также другие глобальные сети, например Bitnet или Sprint. К этим сетям могут подключаться машины разных типов. Каждая из подсетей работает в соответствии со своими специфическими требованиями и имеет свою природу связи, сама разрешает свои внутренние проблемы. Однако предполагается, что подсеть может принять пакет информации и доставить его по указанному в этой подсети адресу. Таким образом, две машины, подключенные к одной подсети, могут напрямую обмениваться пакетами, а если возникает необходимость передать сообщение машине другой подсети, то вступают в силу межсетевые соглашения, для чего подсети используют межсетевой язык — протокол IP. Сообщение передается по цепочке шлюзов и подсетей, пока оно не достигнет нужной подсети, где доставляется непосредственно получателю. Аналогом Internet в России является сеть EUnet/Relcom.

Основная задача сети Relcom — обеспечить, не только доступ к компьютерным ресурсам, но и взаимодействие различных профессиональных групп, рассредоточенных на большой территории.

В настоящее время сеть акционерного общества Relcom является скорее средством общения разработчиков новых решений, чем частью устойчивых общественных структур. Предполагается, что дальнейшее развитие глобальной сети приведет к появлению специализированных сетей, отражающих потребности конкретных групп общения (например, муниципальных, банковских, биржевых сетей) в обмене информацией. Relcom объединяет пользователей почти 2500 организаций, расположенных в более чем 200 городах России и государств ближнего зарубежья.

Узловые машины сети осуществляют передачу почтовых сообщений и новостей между регионами и распространение сообщений на своей территории. Пользовательские персональные машины под управлением операционных систем UNDC или MS DOS используют для общения с региональными узлами протокол UUPP. Скорость обмена — от 1200 до 9600 бит/с. Крупные региональные центры обмениваются сообщениями со скоростью 19,2 Кбит/с. Используются коммутируемые телефонные линии, специализированная телефонная сеть и выделенные линии, протоколы UUPP и TCP/IP (в зависимости от возможностей физических каналов).

Каждый узел сети является самостоятельным юридическим лицом. Координацию работ по развитию и эксплуатации сети осуществляет АО Relcom (зарегистрированная торговая марка фирмы «Демос+»). Профессиональное взаимодействие сотрудников узлов сети осуществляется в рамках специальной группы по интересам Ассоциации групп пользователей системы UNDC.

Relcom обеспечивает передачу электронной почты внутри страны и за рубеж абонентам сети Internet напрямую в сети EUnet, BITNET, MCI-mail, CompuServe и др. По соглашению с информационными агентствами пользователь сети Relcom может получать аналитические материалы по коммерческой деятельности, политические и экономические новости, обзоры материалов популярных изданий. Пользователь может также

знакомиться с состоянием рынка ценных бумаг, получая предложения и результаты биржевых торгов из разных регионов страны.

В сети Relcom распространяются материалы системы ClariNet, включающей в себя электронные версии различных газет и журналов со всего мира. Это означает доступ к коммерческим источникам информации, таким, как ClariNews (новости агентства UPI), TechWire (обзоры наиболее значимых событий в области науки, техники и технологии), ClariNews-Biz (анализ экономических показателей, биржевые отчеты, курсы валют и ценных бумаг и др.), NewsBytes (ежедневный электронный журнал, посвященный проблемам компьютерной индустрии).

В Relcom осуществляется переход на протоколы более высокого уровня, предоставляется такой вид услуг, как выделенный доступ по соответствующей линии. При таком доступе имеется возможность работы с протоколами FTP (File Transfer Protocol) — протокол передачи данных, стандарт пересылки данных в Internet, Telnet, а также рядом других протоколов и соответствующим прикладным и системным программным обеспечением.

Для дальнейшего развития услуг сети планируется расширить число информационных источников, организовать специализированные экспертные услуги, обеспечить возможность доставки электронных писем с использованием факсимильной связи. Техническое развитие сети прежде всего связывается с повышением пропускной способности каналов связи, широким переходом на протоколы более высокого уровня и расширением сервиса, предоставляемого пользователю.

Другим примером российской глобальной телекоммуникационной системы служит сеть «Спринт»: система, созданная с целью обмена финансовой и деловой информацией между абонентами сети. «Спринт» обеспечивает интегрированные решения в области телекоммуникаций, высокую надежность, скорость и мировое качество услуг связи. Официальным поставщиком услуг этой сети с 1995 г. является АО «Спринт-сеть». В число предлагаемых услуг входят: электронная почта; факсимильная, телексная и телетайпная связь; доступ в глобальную сеть Internet, к информационным ресурсам и финансовым базам данных; осуществление банковских платежей; услуги фи-

нансово-информационной системы Reuters; телекоммуникационные услуги на основе пластиковых карточек; создание глобальных и частных клиентских сетей. «Спринт-сеть» имеет свои центры доступа почти в 150 городах и ежедневно передает несколько десятков гигабайт информации между своими клиентами.

Для обеспечения доступа к глобальным сетям пользователю необходимо осуществить подключение к подсети с помощью определенных методов доступа, основанных на взаимосвязи протокола обмена и типа линии связи.

4.7.5. Сеть Internet

Internet — глобальная компьютерная сеть, или объединение сетей (прообразом Internet является сеть ARPAnet, созданная в 1969 г.). В настоящее время это одна из самых больших по количеству включенных в нее компьютеров сетей. Она имеет отлично развитый спектр сетевых услуг, в том числе телеконференции и электронную почту, работает в режимах off-line и on-line. Основой Сети является Internet-протокол TCP/IP, обеспечивающий связь между компьютерами. Он разработан Министерством обороны США.

Под протоколом понимается:

— совокупность правил и соглашений, определяющих параметры, форматы и процедуры обмена данными между физическими и логическими устройствами;

— список событий с указанием продолжительности, упорядоченный по времени. Обычно ведутся протокол связи и протокол работы системы.

Среди сетевых услуг, предлагаемых Internet, самыми распространенными являются передача файлов (по протоколу FTP), работа с гипертекстовыми документами (WWW) и электронная почта.

Internet объединяет около 70 тыс. независимых сетей; в 1995 г. к нему было подключено около 6,6 млн узловых компьютеров в более чем 110 странах и свыше 50 млн пользователей. На конец 1997 г., по данным NOP Research, число пользователей Internet достигало: в США — 51 млн (25 % населения), в Анг-

лии — 7 млн (16 %), в Германии — 9,5 млн (14 %), во Франции — 3 млн (8 %). В России, по данным «Computer Word», насчитывается около 1 млн (менее 1 %) пользователей Internet.

На август 2001 г., по данным отчета Американской исследовательской компании Nielsen/NetRatings, число пользователей Сети составило уже 459 млн человек. Из них на долю США и Канады приходится 40 %, на Европу, Ближний Восток и Африку — 27, на Азию — 22, на Латинскую Америку — 4 %. Больше всего пользователей сети Internet в Южной Корее, Швеции и Австралии — 65 % населения.

В Internet приходят за информацией. Ее источником являются ресурсы, расположенные на компьютерах Сети, которые, так же как и на любом не связанном с Сетью персональном компьютере, представляют собой информационные объекты, существующие в виде логически завершенных записей, или файлов. Существуют две важные категории файлов — файлы, содержащие исполняемые программы, и файлы, содержащие данные всевозможных типов (текст, графику, аудио и видео). Работа с программами требует наиболее серьезных навыков со стороны пользователя, в то время как общение с текстовыми документами в принципе допускает знание всего одной-единственной программы их просмотра. Естественно, что именно текстовые документы востребованы сегодня в Internet в наибольшей степени.

Пользователь Internet может получить доступ к ресурсам других сетей благодаря существованию межсетевых шлюзов. Под **шлюзом** (gateway) принято понимать специализированный узел (рабочую станцию, компьютер) локальной сети, обеспечивающий доступ других узлов данной локальной сети к внешней сети передачи данных и другим вычислительным сетям. Говоря о *межсетевом шлюзе,* часто подразумевают и аппаратные, и программные средства, обеспечивающие межсетевую связь.

Передача информации в Internet происходит небольшими порциями данных, которые называются **пакетами**; пакеты имеют строго определенную структуру. Сообщение может быть разбито на несколько пакетов, размер которых варьируется, но, как правило, не превышает 1500 байт.

Важнейшим условием функционирования Internet является стандартизованный свод правил передачи пакетов данных

в Сети и за ее пределы в рамках межсетевого обмена, закрепленный базовым транспортным протоколом TCP и межсетевым протоколом IP. Протокол TCP дает название всему семейству протоколов TCP/IP, главной задачей которых является объединение в сети пакетных подсетей через шлюзы. Каждая сеть работает по своим собственным законам, однако предполагается, что шлюз может принять пакет из другой сети и доставить его по указанному адресу. Реально пакет из одной сети передается в другую подсеть через последовательность шлюзов, что становится возможным благодаря реализации во всех узлах сети протокола межсетевого обмена IP.

Величину потока информации (объем последней измеряется в битах или байтах и единицах, им кратных), прошедшего за определенный промежуток времени через выделенный канал связи, шлюз или другую систему, принято называть **трафиком**.

В Internet каждой машине (host) приписан определенный адрес, по которому к ней и осуществляется доступ в рамках одного из стандартных протоколов, причем существует одновременно как числовая адресация (так называемый IP-адрес, состоящий из набора четырех чисел, разделенных точками, например: 144.206.160.32), так и более удобная для восприятия система осмысленных *доменных имен* (например: apollo. polyn.kiae.su). Host — это хост-компьютер, т. е. компьютер, управляющий работой сети, к которой подключается пользователь при работе в Сети. Чтобы обратиться к машине, можно использовать как ее IP-адрес, так и ее имя. Для упрощения работы в Сети служит специальная система DNS (Domain Name System), представляющая собой базу данных, которая обеспечивает преобразования доменных имен компьютеров в числовые IP-адреса, поскольку базовым элементом адресации для семейства протоколов TCP/TP являются IP-адреса, а доменная адресация выполняет роль сервиса.

Информационные ресурсы Internet — это вся совокупность информационных технологий и баз данных, доступных с помощью этих технологий и существующих в режиме постоянного обновления. К их числу относятся, например:

— электронная почта (E-mail);
— система телеконференций Usenet;
— система файловых драйвов FTP;

— базы данных WWW;
— базы данных Gopher;
— базы данных WAIS;
— информационные ресурсы LISTSERV;
— справочная служба WHOIS;
— информационные ресурсы TRICKLE;
— поисковые машины Open Text Index, Alta Vista, Yahoo, Lycos и др.

Internet — это главным образом возможность получить информацию в тот же момент, когда она нужна, т. е. в режиме on-line. Но если нет возможности работать в on-line, то для доступа к услугам большинства информационных серверов Internet можно воспользоваться электронной почтой, хотя в этом случае все будет происходить не так быстро, как в стандартном режиме TELNET, FTP или WWW, о которых будет сказано ниже.

Общий принцип доступа к любому информационному ресурсу через электронную почту заключается в том, что пользователь посылает сообщение почтовому роботу (специальному почтовому серверу), который реализует стандартный доступ к ресурсу и отправляет ответ по почте пользователю (рис. 4.6).

При такой схеме доступа общение между пользователем и почтовым роботом происходит в режиме работы электронной почты, а между почтовым роботом и сервером (FTP, WAIS или WWW) — по протоколу робота этого сервера.

Сеть открывает доступ к обоим видам этих ресурсов, если пользователь в состоянии ответить на следующие вопросы: как найти нужный информационный объект? Как его использовать — на удаленной машине или перенести на свой (локальный) компьютер? Какими программными средствами сделать нужный информационный объект воспринимаемым, т. е. прочитанным, озвученным и т. д.

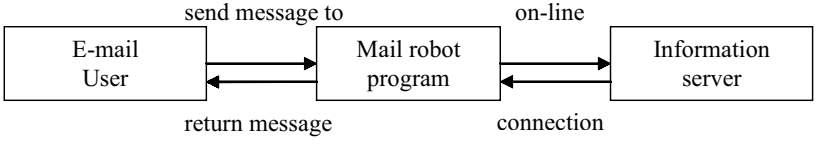


Рис. 4.6. Взаимодействие с информационным ресурсом через электронную почту

В силу колоссального объема и разнородности организации информационных ресурсов в Сети возникает ряд естественных проблем. Каждый ресурс имеет структуру определенного типа, базируется на машине со своей операционной системой (платформой) и специальной программой обслуживания доступа к ней — *программой-сервером*. Машину, непрерывно функционирующую в Сети, где исполняется такая программа, также часто называют *сервером*. Само соединение пользователя с сервером происходит с помощью соответствующей программы (*программы-клиента*), запускаемой на его компьютере, и выполняется такое соединение на основе заранее определенного свода правил, или *протокола взаимодействия* между клиентом и сервером. Таким образом, для начала работы в Сети необходимо:

— иметь какую-либо программу-клиент на своем компьютере;

— располагать адресом хотя бы одного сервера (например, из изданного полиграфическим способом книжного справочника, такого, как «Желтые страницы Internet»), к которому можно обратиться по протоколу, поддерживаемому собственной программой-клиентом.

— владеть набором команд, используемых в рамках данного протокола.

Иерархия протоколов Internet. Слово «протокол» в сетевых технологиях имеет смысл, близкий, но несколько отличный от значения «документ с записью всего происходящего». За ним стоит многозначное понятие, применяемое в разных контекстах, наиболее важным из которых для конечного пользователя является представление о протоколе как о некотором своде четко определенных правил, которые одинаково реализованы в различных системах (программах, шлюзах, пакетах данных и др.). Благодаря этому в местах взаимодействия систем (например, при инициировании соединения программы-клиента с программой-сервером или при попадании передаваемого пакета данных на машину-шлюз) все происходит по заранее определенному сценарию.

Пример. Чтобы пояснить понятие протокола, рассмотрим пример, не имеющий отношения к компьютерным сетям, а именно: обсудим взаимодействие двух предприятий, связан-

ных между собой деловым сотрудничеством. Между ними существуют многочисленные соглашения, в том числе, например, договоренность о регулярных поставках продукции одного предприятия другому. В соответствии с этой договоренностью начальник отдела продаж одного предприятия каждый месяц должен посылать сообщение начальнику отдела закупок второго предприятия о том, сколько и какого товара они могут поставить в этом месяце. В ответ на это сообщение начальник отдела закупок посылает заявку на требуемое количество продукции. Условленный порядок взаимодействия начальников в данном случае соответствует понятию «протокол уровня начальников». Начальники посылают свое сообщение и заявки через своих секретарей.

После того как сообщения переданы секретарям, начальников не волнует, каким образом эти сообщения будут перемещаться дальше — обычной или электронной почтой, факсом или с нарочным. Выбор способа передачи — уровень компетенции секретарей, они могут решать данный вопрос, не уведомляя об этом своих начальников, так как их протокол взаимодействия связан только с передачей сообщений, поступающих сверху, и не касается содержания этих сообщений. Отправив письмо, секретари считают свою функцию выполненной, разве что в правила работы хорошего секретаря входит еще и проверка получения сообщения адресатом.

При решении других вопросов начальники могут взаимодействовать по другим правилам, но это не повлияет на работу хороших секретарей, для которых не важно, какие сообщения отправлять, а важно, чтобы они дошли до адресата. Итак, в данном случае мы имеем дело с двумя уровнями — начальниками и секретарями, и каждый уровень имеет собственный протокол, который может быть изменен независимо от протокола другого уровня. Эта независимость протоколов друг от друга и делает привлекательным многоуровневый подход.

Как и в рассмотренном примере, по мере продвижения пакета данных по сети на каждом этапе его взаимодействия с другими сетевыми элементами отрабатывают протоколы разных уровней. Полную совокупность таких протоколов, необходимых для успешного взаимодействия разных элементов в рамках сети данного типа, принято называть **семейством** или

стеком. Internet работает под семейством протоколов TCP/IP, которое имеет многоуровневую структуру.

За долгие годы использования в сетях различных стран и организаций стек TCP/IP накопил большое количество протоколов и сервисов прикладного уровня. Проблема соединения разнородных сетей в единую сеть Internet была решена посредством применения многоуровневой системы протоколов. Так, протокол IP осуществляет минимальные услуги по перемещению данных в Сети (адресация, фрагментация, сборка). TCP, являясь протоколом следующего уровня, обеспечивает надежную доставку данных, а протокол TELNET отвечает за «взаимопонимание» приложений различных компьютеров. Протокол UDP (User Datagram Protocol) также отвечает за надежную достатку данных, а протокол верхнего уровня SMTP (Simple Management Transfer Protocol) обеспечивает работу электронной почты, протоколы SNPM (Simple Network Management Protocol) — управление сетью, FTP — передачу файлов, NFS — доступ к файловым системам удаленных компьютеров и т. д.

Таким образом, сеть Internet можно определить как совокупность ЛВС, удовлетворяющих протоколу TCP/IP, которая имеет общее адресное пространство, где у каждого компьютера есть свой уникальный IP-адрес.

Модель «клиент-сервер» как основа построения информационных сервисов Internet. В основу взаимодействия компонентов информационных сервисов Сети в большинстве случаев положена модель «клиент-сервер». Как правило, в качестве клиента выступает программа, которая установлена на компьютере пользователя, а в качестве сервера — программа, установленная у провайдера. В данном контексте под *провайдером* понимается организация (или частное лицо), которая ведет (поддерживает) информационные ресурсы.

При этом возможны два варианта организации самой информационной системы, которая обеспечивает доступ к информационному ресурсу. Большинство систем Internet построены по принципу взаимодействия «каждый с каждым». Например, в системе World Wide Wed (WWW) каждый пользователь может напрямую взаимодействовать с каждым сервером без посредников. Такой подход позволяет упростить всю технологическую схему построения системы, однако приводит к порождению

большого трафика в Сети. Альтернативным вариантом построения сети является, например, система Usenet, в которой пользователь может взаимодействовать только со «своим» сервером и не может обратиться к произвольному серверу в Сети. Однако доступ он получает ко всей информации, которая присутствует в данной информационной системе, так как серверы обмениваются ею между собой.

Наиболее «древние» услуги Internet: электронная почта (E-mail), TELNET и FTP. TELNET обеспечивает доступ к БД, каталогам библиотек, другим информационном услугам, создавая ощущение, что вы сидите за удаленным компьютером и пользуетесь всем, что он имеет. FTP — протокол передачи файлов — позволяет перемещать любые файлы между двумя ПК. Чтобы попасть в открытую для доступа область FTP-сервера, достаточно представиться анонимным пользователем и посредством FTP можно получить доступ к библиотеке Ватикана, многим университетам и даже послать письмо вице-президенту США; можно запросить справочные документы, учебники, программы. В Internet имеются материалы по 4500 темам, доступ к которым обеспечивается посредством системы Usenet.

Система Bitnet создает списки рассылки. Для поиска требуемых материалов можно воспользоваться системой Archie, которая представляет собой распределенную БД, хранящую каталоги по более чем 1000 хранилищам информации. Система предоставляет меню для поиска.

Для проведения досуга можно воспользоваться услугами системы IRC, а посредством Talk — побеседовать с удаленным пользователем. MUDS (Multi-User Dimension/Dungeons) позволяет войти в мир фантазий.

Так как нет организации, ведущей централизованный учет всех подключенных к Internet компьютеров, трудно описать ее протяженность и размеры. По оценкам фирмы IBM, ее размер достигает 3,2 млн host-ЭВМ, соединяющих 25 000 сетей и 100 млн пользователей.

4.7.6. Основные услуги компьютерных сетей

Электронная почта. Технология компьютерного способа пересылки и обработки информации, позволяющая поддерживать оперативную связь между сотрудниками, руководителями,

учеными, деловыми людьми и всеми желающими, получила название **электронной почты**. Электронная почта (E-mail) — это специальный пакет программ для хранения и пересылки сообщений между пользователями ЭВМ. Посредством электронной почты реализуется служба безбумажных почтовых отношений. Электронная почта представляет собой систему сбора, регистрации, обработки и передачи любой информации (текстовых документов, изображений, цифровых данных, звукозаписи и т. д.) по сетям ЭВМ и выполняет такие функции, как редактирование документов перед передачей; хранение их в специальном банке; пересылка корреспонденции; проверка и исправление ошибок, возникающих при передаче; выдача подтверждения о получении корреспонденции адресатом; получение и хранение информации в собственном «почтовом ящике»; просмотр полученной корреспонденции.

Электронная почта является услугой вычислительных сетей, и поставщики сетевых операционных систем комплектуют свои продукты средствами поддержки электронной почты.

В локальных сетях электронная почта обеспечивает передачу документов и успешно используется при автоматизации конторских работ. Для связи между сотрудниками всего офиса она оказывается удобнее телефона, так как позволяет передавать такую информацию, как отчеты, таблицы, диаграммы и рисунки.

Передача между терминалами сообщений, например фототелеграмм, также может рассматриваться как разновидность электронной почты. Однако для большинства конкретных случаев использование электронной почты предполагает передачу сообщений через специальные почтовые ящики, между которыми размещаются устройства обработки данных. **Почтовый ящик** — общая область памяти вычислительной сети, предназначенная для записи информации с помощью одной прикладной программы с целью ее дальнейшего использования другими прикладными программами, функционирующими в других узлах сети. Почтовый ящик представляет собой специально организованный файл для хранения корреспонденции. Он состоит из двух корзин: отправления и получения. Любой пользователь может обратиться к *папке получения* другого пользователя и сбросить туда информацию. Но просмотреть ее он не может. Из *папки отправлений* почтовый сервер забирает информацию

для рассылки другим пользователям. Каждый почтовый ящик имеет сетевой адрес. Для пересылки корреспонденции можно установить связь с почтовым ящиком в режиме on-line. Например, в сети Sprintmail пользователь, зарегистрировавшись и получив определенный статус, по телефонным каналам может входить в ближайший к нему узел сети и сообщаться с нужными абонентами в режиме on-line. Этот способ неудобен, так как необходимо ждать, пока будет включена ЭВМ получателя.

Более распространенным способом является выделение отдельных компьютеров в качестве почтовых отделений, называемых **почтовыми серверами**. При этом все компьютеры получателей подключены к ближайшему почтовому серверу, получающему, хранящему и пересылающему дальше по сети почтовые отправления, пока они не дойдут до адресата. Отправка адресату осуществляется по мере его выхода на связь с ближайшим почтовым сервером в режиме off-line. Примером может служить сеть Relcom. Пользователь передает сообщение вместе с адресом по телефонному каналу через модем на ближайший почтовый сервер в режиме on-line. Сообщение регистрируется, ставится в очередь и по первому свободному каналу передается на следующий почтовый сервер, пока адресат не заберет его в свой почтовый ящик. Почтовые серверы реализуют такие функции, как обеспечение быстрой и качественной доставки информации, управление сеансом связи, проверка достоверности информации и корректировка ошибок, хранение информации «до востребования» и извещение пользователя о поступившей в его адрес корреспонденции, регистрация и учет корреспонденции, проверка паролей при запросах корреспонденции, поддержка справочников с адресами пользователей.

Пересылка сообщений пользователю может выполняться в индивидуальном, групповом и общем режимах. В *индивидуальном режиме* адресатом является отдельный компьютер пользователя, хотя корреспонденция рассылается одновременно группе адресатов. Почтовые серверы имеют средства распознавания пользователей. В *общем режиме* корреспонденция отправляется всем пользователям — владельцам почтовых ящиков. Посредством группового и общего режимов можно организовать телеконференцию, электронные доски объявлений. Во избежание перегрузки почтовых ящиков в почтовых серве-

рах хранятся справочники адресов, содержащих фильтры для групповых и общих сообщений.

К преимуществу электронной почты относятся скорость и надежность доставки корреспонденции, относительно низкая стоимость услуг, возможность быстро ознакомить с сообщением широкий круг пользователей.

Электронная почта глобальных сетей передачи сообщений, где могут объединяться компьютеры самых различных конфигураций и совместимостей, обеспечивает:

— работу в режиме off-line, когда не требуется постоянного присутствия на почтовом узле. Достаточно указать специальной программе-почтовику (Mailer) время системных событий и адреса, где следует забирать почту;

— доступ к телеконференции (Echo Conference);

— доступ к файловым телеконференциям (File Echo Conference).

Файловые телеконференции отличаются от обычных тем, что в качестве сообщений в них служат не письма, а файлы. Например, создается файловая телеконференция, посвященная экономике, где каждый участник телеконференции может поместить свой файл, а другие участники его непременно получат.

Существуют и другие возможности, предоставляемые членам Сети. Можно, например, послать заказ на посылку или прием факса. Составляется обычное электронное письмо, оформленное должным образом, и посылается на адрес компьютерного узла, занимающегося факсимильными операциями. Текст этого письма в виде факса будет доставлен на факсимильный аппарат адресата.

Электронная почта в Internet. Электронная почта является чрезвычайно важным информационным ресурсом Internet. Помимо того что она представляет собой самое массовое средство электронных коммуникаций, через нее можно принять или послать сообщения еще в два десятка международных компьютерных сетей, часть из которых вовсе не имеют on-line сервиса (т. е. прямого подключения к Internet).

Электронная почта во многом похожа на обычную почтовую службу. Корреспонденция подготавливается пользователем на своем рабочем месте либо программой подготовки почты, либо обычным текстовым редактором. Затем пользова-

тель должен вызвать программу отправки почты (программа подготовки почты вызывает программу отправки автоматически), которая посылает сообщение на почтовый сервер отправителя. Тот, в свою очередь, посылает его на почтовый сервер адресата, где специальная программа занимается сортировкой почты и рассылкой ее по ящикам конечных пользователей. После запуска программы получения почты адресат устанавливает соединение со своим почтовым сервером и организует пересылку всех полученных на свое имя сообщений. Отметим, что почтовые серверы постоянно подключены к Сети, тогда как компьютеры участников переписки могут устанавливать соединение с ними по мере необходимости. Кроме того, получить и отправить почту можно через разные серверы Internet. При настройке программы работы с электронной почтой, независимо от ее интерфейса, необходима следующая информация от провайдера: имя сервера исходящей почты, имя сервера входящей почты, имя пользователя и пароль, а также типы протоколов, используемые при почтовом обмене.

Протокол Simple Mail Transfer Protocol (SMTP). Этот протокол разработан специально для электронной почты в Internet. Он является протоколом прикладного уровня и использует транспортный протокол TCP. Однако совместно с SMTP-протоколом применяется и протокол UUCP (Unix-to-Unix-Copy-Program), который хорошо подходит для использования телефонных линий связи.

В целом же общие рекомендации таковы: если имеется возможность надежно работать в режиме on-line (что и является нормой), то следует настраивать почту для работы по протоколу SMTP. Если линии связи не удовлетворяют заказчика или on-line используется чрезвычайно редко, то лучше применять UUCP.

Основой любой почтовой службы является система адресов. Без точного адреса невозможно доставить почту адресату. В Internet принята система адресов, которая базируется на доменном адресе машины. Например, для пользователя с именем tala с адресом citmgu.ru почтовый адрес будет выглядеть так: tala@citmgu.ru

Таким образом, адрес состоит из двух частей: идентификатора пользователя, который записывается перед знаком «коммерческого эй» — «@», и доменного адреса машины, который записывается после знака «@».

Взаимодействие в рамках SMTP строится по принципу двусторонней связи, которая устанавливается между отправителем и получателем почтового сообщения. При этом отправитель инициирует соединение и посылает запросы на обслуживание, а получатель на них отвечает. Фактически отправитель выступает в роли клиента, а получатель — в роли сервера.

Канал связи устанавливается непосредственно между отправителем и получателем сообщения. При таком взаимодействии почта достигает абонента в течение нескольких секунд после отправки.

Такая схема предполагает, что пользователь имеет почтовый ящик на машине-сервере, которая не выключается круглосуточно. Все почтовые сообщения складываются в этот почтовый ящик. По мере необходимости пользователь из своего почтового сервера обращается к почтовому ящику и забирает из него пришедшую на его имя почту. При отправке программа-клиент обращается непосредственно к серверу рассылки почты и передает отправляемые сообщения на этот сервер для дальнейшей рассылки.

Usenet — система телеконференций Internet. Она построена по принципу электронных досок объявлений, когда любой пользователь может поместить свою информацию в одну из групп новостей Usenet и эта информация станет доступной другим пользователям, которые на данную группу новостей подписаны. Именно этим способом распространяется большинство сообщений Internet, например списки наиболее часто задаваемых вопросов (FAQ) или реклама программных продуктов. Usenet — хорошее место для объявления международных конференций и семинаров. Но по Usenet можно получить и вирус, если заказывать и распаковывать все подряд, что приходит на ваш почтовый адрес.

FTP — система файловых архивов. Это огромное распределенное (т. е. расположенное на машинах Сети, в том числе и функционирующих на разных платформах) хранилище всевозможной информации, накопленной за последние 10—15 лет в Сети. Любой пользователь может прибегнуть к услугам анонимного доступа к этому хранилищу и скопировать интересующие материалы. Объем программного обеспечения в архивах FTP составляет терабайты информации, и ни один пользователь или администратор Сети не может просто физически

обозреть эту информацию. Кроме программ, в FTP-архивах можно найти стандарты Internet — RFC (Request for Comments), пресс-релизы, книги по различным отраслям знания, главным образом по компьютерной проблематике, и многое другое. Практически любой архив строится как иерархия директорий. Многие архивы дублируют информацию из других архивов (так называемые «зеркала» — mirrors). Чтобы получить нужную информацию, вовсе не обязательно ждать, когда она будет передана из Австралии или Южной Африки, можно поискать «зеркало» где-нибудь ближе, например в Финляндии или Швеции. Для этой цели существует специальная программа Archie, которая позволяет просканировать FTP-архивы и найти тот, который устраивает пользователя по составу программного обеспечения и коммуникационным условиям.

Word Wide Web (Всемирная паутина) — распределенная гипертекстовая информационная система, темпы развития которой стремительно нарастают. Word Wide Web (WWW) предоставляет удобный доступ к большинству информационных архивов Сети. Особенностью системы является механизм гипертекстовых ссылок, который позволяет просматривать материалы в порядке выбора этих ссылок пользователем. Многие интерфейсы данной технологии обеспечивают выбор интересующих материалов простым нажатием кнопки мыши на нужном слове или поле графической картинки. Система универсальных адресов предоставляет возможность проадресовать практически все информационные ресурсы Internet. Многие издательства взяли WWW на вооружение для электронных версий своих журналов. В WWW существует большое количество различного рода каталогов, которые позволяют ориентироваться в Сети. Кроме этого, пользователи могут выполнить даже удаленные программы или смотреть по Сети фильмы. Такой сервис не обеспечивается другими информационными системами Internet (подробнее о WWW см. ниже).

Gopher — еще одна распределенная информационная система Internet. В основу ее интерфейсов положена идея иерархических каталогов. Внешне Gopher выглядит как огромная файловая система, которая расположена на машинах Сети. Первоначально Gopher задумывалась как информационная система университета с информационными ресурсами факультетов, кафедр, общежитий и т. п. До сих пор ее основные информацион-

ные ресурсы сосредоточены в университетах. Gopher считается простой системой, легкой в установке, администрировании, достаточно надежной и защищенной. В России Gopher-серверы не так распространены, как во всем мире: профессионалы предпочитают Word Wide Web.

WAIS — распределенная информационно-поисковая система Internet. Родилась WAIS как перспективная разработка четырех ведущих американских компаний и первое время была коммерческим продуктом, пока не появилась ее свободно распространяемая версия free WAIS. В основу системы положен принцип поиска информации с использованием логических запросов, основанных на применении ключевых слов. Клиент «обшаривает» все серверы WAIS на предмет наличия в них документов, удовлетворяющих запросу. WAIS широко применяется как поисковая машина в других информационных серверах Internet, например WWW и Gopher. Наиболее известным проектом, где была применена WAIS, является электронная версия энциклопедии «Британика».

LISTSERV — это, строго говоря, не сервис Internet, а система почтовых списков сети BITNET (сеть образовательных учреждений). Однако это очень популярный ресурс в глобальных компьютерных сетях, и в Internet существуют шлюзы для доступа к нему. LISTSERV специально ориентирован на применение в качестве транспорта электронной почты. Доступ к нему в интерактивном режиме затруднен. В мире существуют многие сотни списков LISTSERV, которые организованы по группам интересов (например, группы разработчиков программ ядерно-физических расчетов EGS-4, группы любителей научной фантастики и др.). LISTSERV довольно сильно пересекается с Usenet, однако это не мешает сосуществованию обеих систем.

WHOIS-служба содержит информацию о пользователях Сети, их электронные и почтовые адреса, идентификаторы и реальные имена. В последнем случае дается краткое описание основных направлений их деятельности. WHOIS — распределенная система. Это значит, что запросы отправляются по всему множеству серверов WHOIS в Internet, если только не указан адрес конкретного сервера.

Trickle — это доступ по почте к архивам FTP, который организован через специальный шлюз. Этот шлюз имеет специальные навигационные средства для поиска нужной информации

в Сети. Пользователь может вести с ним своеобразный диалог по электронной почте, выбирая нужную информацию путем ввода специальных команд Trickle.

Поисковые машины Open Text Index, AltaVista, Yahoo, Lycos и другие представляют собой мощные информационно-поисковые системы, размещенные на серверах свободного доступа, специальные программы которых непрерывно в автоматическом режиме сканируют информацию Сети на основе заданных алгоритмов, проводят индексацию документов. В последующем поисковые машины предоставляют пользователю на основе созданных баз данных доступ к распределенной на узлах Сети информации через выполнение поискового запроса в рамках собственного интерфейса.

Гипертекст (от англ. hypertext) — это технология, обеспечивающая хранение текстового и графического материала и возможность быстрого доступа к различным частям текста за счет использования гиперссылки. Гипертекст позволяет создать компьютерный учебник, в котором не надо тратить время на поиск разъяснений и подсказок. В обычном учебнике текст располагается последовательно. Учебник, построенный как гипертекст, делает более доступными разные части текста, вплоть до одновременного появления на дисплее взаимосвязанных понятий с разных страниц и разных компьютеров.

Сетевые технологии существенно повышают интеллектуальные возможности человека, примером этого могут служить две ИТ — гипертекст и мультимедиа. Еще в 1945 г. научный советник президента США Г. Трумена В. Буш, проанализировав способы представления информации в виде отчетов, докладов, проектов, графиков, планов и поняв неэффективность такого размещения, предложил способ размещения информации по принципу ассоциативного мышления. На базе этого принципа была разработана модель гипотетической машины MEMEKC. Через двадцать лет Т. Нельсон реализовал этот принцип на ЭВМ и назвал его **гипертекстом**.

Гиперссылка (от англ. hyperlink) — специальные пометки в тексте, распознаваемые программой *(браузером)*, которая переходит к указанному фрагменту данного текста или к другому файлу, расположенному в общем случае на другом компьютере. Гиперссылки расставляет разработчик текста в соответствии с требованиями браузера.

Внешне гипертекст отличается от обычного текста тем, что часть слов или целые строки в нем, будучи выделены особым шрифтом или цветом, оказываются чувствительными к появлению на них указателя манипулятора мышь. При попадании на такую область текста указатель (часто стрелочка) изменяет первоначальный вид, становясь, например, ладошкой. Щелчок мыши в таком положении приводит к инициированию какого-либо события, чаще всего к загрузке в программу просмотра нового документа, привязанного так называемой *гипертекстовой ссылкой* к выделенной строке текста. В результате у пользователя появляется возможность самому выбирать порядок просмотра тех или иных страниц, двигаясь по перемежающимся между собой нитям — «паутинкам» ссылок. Если при этом компьютер подключен к глобальной сети Internet, то в сценарий просмотра могут входить ресурсы всего мира, доступ к которым происходит по протоколу работы с гипертекстом, или HTTP (Hyper Text Transfer Protocol). После сказанного становится понятным представление об этих ресурсах как о Всемирной паутине (WWW).

Поскольку нетривиальный характер взаимодействия клиента и сервера по протоколу HTTP с удаленными ресурсами Сети скрыт от конечного пользователя за интерфейсом дружественной программы просмотра гипертекстовых страниц браузером (от англ. browse — просматривать), начало работы в Web не представляет больших проблем.

Гипертекст не может корректно отображаться обычным текстовым редактором, хотя последний вполне пригоден для его подготовки. Специально разработанный язык гипертекстовой разметки HTML (Hyper Text Markup Language) позволяет превращать нужные элементы документа (включая не только текстовые поля, но и графику) в гипертекстовые ссылки.

Гипертекст обладает нелинейной сетевой формой организации материала, разделенного на фрагменты, для каждого из которых указан переход к другим фрагментам по определенным типам связей. При установлении связей можно опираться на разные основания (ключи), но в любом случае речь идет о смысловой (семантической) близости связываемых фрагментов. Следуя указанным связям, можно читать или осваивать материал в любом порядке, а не в одном-единственном. Текст теряет свою замкнутость, становится принципиально откры-

тым, в него можно вставлять новые фрагменты, указывая для них связи с уже имеющимися. Структура текста не разрушается, и вообще у гипертекста нет априорно заданной структуры. Таким образом, *гипертекст — это новая технология представления неструктурированного свободно наращиваемого знания.* Этим он отличается от других моделей представления информации.

Итак, под **гипертекстом** понимают систему информационных объектов (статей), объединенных между собой направленными связями, образующими Сеть. Каждый объект связывается с информационной панелью экрана, на которой пользователь может ассоциативно выбирать одну из связей. Объекты не обязательно должны быть текстовыми, они могут быть графическими, музыкальными, с использованием средств мультипликации, аудио- и видеотехники. Обработка гипертекста открыла новые возможности освоения информации, качественно отличающиеся от традиционных. Вместо поиска информации по соответствующему *поисковому ключу*, гипертекстовая технология предполагает перемещение от одних объектов информации к другим с учетом их смысловой связанности. Обработке информации по правилам формального вывода в гипертекстовой технологии соответствует запоминание пути перемещения по гипертекстовой сети.

Гипертекстовая технология ориентирована на обработку информации не вместо человека, а вместе с человеком, т. е. становится *авторской.* Она удобна тем, что пользователь сам определяет подход к изучению или созданию материала с учетом своих индивидуальных способностей, знаний, уровня квалификации и подготовки. Гипертекст содержит не только информацию, но и аппарат ее эффективного поиска. По глубине формализации информации гипертекстовая технология занимает промежуточное положение между документальными и фактографическими информационными системами.

Структурно гипертекст состоит из информационного материала, тезауруса гипертекста, списка главных тем и алфавитного словаря.

Информационный материал подразделяется на информационные статьи, состоящие из заголовка статьи и ее текста. Заголовок содержит тему или наименование описываемого объекта. Информационная статья содержит традиционные определения и понятия, должна занимать одну панель

и быть легко обозримой, чтобы пользователь мог понять, стоит ли ее внимательно читать или перейти к другим, близким по смыслу статьям. Текст, включенный в информационную статью, может сопровождаться пояснениями, примерами, документами, объектами реального мира. Беглый просмотр текста статьи упрощается, если эта вспомогательная информация визуально отличается от основной, например подсвечена или выделена другим шрифтом.

Тезаурус гипертекста — это автоматизированный словарь, отображающий смысловые отношения между лексическими единицами дескрипторного информационно-поискового языка и предназначенный для поиска слов по их смысловому содержанию. Термин «тезаурус» был введен в XIII в. флорентийцем Брунетто Лотики для названия энциклопедии. С латыни этот термин переводится как «сокровище», «запас», «богатство». В кибернетике тезаурус (от греч. thēsaurós) — это словарь, очищенный от неоднозначности. Тезаурус гипертекста состоит из тезаурусных статей. Тезаурусная статья имеет заголовок и список заголовков родственных тезаурусных статей, где указан тип родства. Заголовок тезаурусной статьи совпадает с наименованием информационной статьи и является наименованием объекта, описание которого содержится в информационной статье. Дескриптор (от лат. descriptor) — лексическая единица (слово, словосочетание) для описания основного содержания документа в информационно-поисковой системе. В отличие от традиционных тезаурусов-дескрипторов, тезаурус гипертекста содержит не только простые, но и составные наименования объектов. Формирование тезаурусной статьи гипертекста означает индексирование текста.

Полнота связей, отражаемых в тезаурусной статье, и точность установления этих связей в конечном итоге определяют полноту и точность поиска при обращении к данной статье гипертекста. Существуют следующие типы родства, или отношений: «вид — род», «род — вид», «предмет — процесс», «процесс — предмет», «целое — часть», «часть — целое», «причина — следствие», «следствие — причина» и т. д. Пользователь получает более общую информацию по родовому типу связи, а по видовому — специфическую информацию без повторения общих сведений из родовых тем. То есть глубина индексирования текста зависит от родо-видовых отношений.

Список заголовков родственных тезаурусных статей представляет собой локальный справочный аппарат, в котором указываются ссылки только на «ближайших родственников». Тезаурус гипертекста можно представить в виде сети: в узлах находятся текстовые описания объекта (информационные статьи), а ребра указывают на существование связи между объектами и на тип родства. В гипертексте поисковый аппарат не делится на тезаурус и массив поисковых образов (документов), как в обычных информационно-поисковых системах. В гипертексте весь поисковый аппарат реализуется как тезаурус гипертекста.

Список главных тем содержит заголовки всех справочных статей, для которых нет ссылок ни с отношениями «род — вид», ни с отношениями «часть — целое». Желательно, чтобы список занимал не более одной панели экрана.

Алфавитный словарь содержит перечень наименований всех информационных статей в алфавитном порядке.

Гипертексты, составленные вручную, используются давно: это справочники, энциклопедии, словари, снабженные развитой системой ссылок. Область применения гипертекстовых технологий очень широка: издательская деятельность, библиотечная работа, обучающие системы, разработка документации, законов, справочных руководств, баз данных, баз знаний и т. д. Наиболее распространенными системами являются HyperCard, HyperStudio, SuperCard, QuickTime фирмы Apple для персональных компьютеров Macintosh, Linkway — для IBM; из отечественных систем можно назвать Flexis II, автоматизированную систему формирования и обработки гипертекста (АСФОГ) и др. В большинстве современных программных продуктов вся помощь (help) составлена с использованием гипертекстовой технологии на базе меню.

Технология WWW. В течение последних десятилетий предпринималось немало попыток разработать концепцию универсальной информационной базы данных, в которой можно было бы не только получать информацию из любой точки земного шара, но и иметь удобный способ связи информационных сегментов друг с другом, так чтобы наиболее важные данные могли быть быстро найдены. В 1960-е гг. исследования в этой области породили понятие информационной Вселенной (docuverse = = documentation + universe), которая преобразила бы всю информационную деятельность, в частности в области образова-

ния. Но только в настоящее время появилась технология, воплотившая эту идею и предоставляющая возможности ее реализации в масштабах планеты.

WWW — это мировая виртуальная файловая система, т. е. широкомасштабная гипермедиасреда, ориентированная на предоставление универсального доступа к документам. Проект WWW возник в начале 1989 г. в Европейской лаборатории физики элементарных частиц (European Laboratory for Ptracticle Physics (CERN) in Geneva, Switzerland). Основное назначение проекта — предоставить пользователям-непрофессионалам доступ к информационным ресурсам. Результатом проекта WWW является предоставление пользователям сетевых компьютеров достаточно простого доступа к самой разнообразной информации.

Используя популярный программный интерфейс, проект WWW изменил процесс просмотра и создания информации. Идея заключается в том, что по всему миру хаотично разбросаны тысячи информационных серверов и любую машину, подключенную к Internet в режиме on-line, можно преобразовать в сервер и начинить его информацией. С любого компьютера, подключенного к Internet, можно свободно установить сетевое соединение с таким сервером и получать от него информацию.

Первый такой сервер был организован в CERN, там же с целью развития и поддержки стандартов WWW-технологий был создан The World Wide Web Consortium (или W3C). WWW-сервер The W3C's Web site является интегрирующим сервером по поддержке Web-технологий Internet.

Позднее к проекту подключились и многие другие организации. Большой вклад в развитие Web-технологий внес Национальный центр суперкомпьютерных приложений (National Centre for Supercomputing Applications — NCSA).

Информационный WWW-сервер использует гипертекстовую технологию. Для записи документов в гипертексте применяется специальный, но очень простой язык HTML, который позволяет управлять шрифтами, отступами, вставлять цветные иллюстрации, поддерживает вывод звука и анимации. В стандарт языка также входит поддержка математических формул.

Вся польза WWW состоит в создании гипертекстовых документов, и если вас заинтересовал какой-либо пункт в таком документе, то достаточно «ткнуть» в него курсором для получе-

ния нужной информации. Также в одном документе можно делать ссылки на другие документы, написанные другими авторами или даже расположенные на другом сервере. Одно из главных преимуществ WWW, по сравнению с другими средствами поиска и передачи информации, — «многосредность». В WWW можно увидеть на одной странице одновременно текст и изображение, звук и анимацию.

Для удобства ввода информации предусмотрены специальные формы, меню. Программы просмотра позволяют получать доступ не только к WWW-серверам, но и к другим службам Internet. С их помощью можно путешествовать по Gopher-серверам, искать информацию в WAIS-базах, получать файлы из файловых серверов по протоколу FTP. Протокол обмена поддерживается сетевыми новостями Usenet NNTP.

Таким образом, WWW — это новая информационная технология для работы в Сети, которая совмещает сетевую технологию с гипертекстом и мультимедиа и реализована на базе сети Internet. Серверы сети WWW (Web-серверы) могут быть построены на разных платформах.

Web-сервер разбит на Web-страницы посредством языка гипертекстовой разметки HTML. Для перемещения по Web-страницам и передачи гипертекстовых документов по Сети разработан протокол HTTP (Hyper Text Transfer Protocol). Для поиска Web-страницы с нужным документом разработаны программы поиска и просмотра, называемые **навигаторами** или **браузерами**. Они обеспечивают интерфейс пользователя с WWW. При этом стиль оформления экрана и форма представления документа задаются пользователем.

Web-сервер содержит Web-страницы с информацией любого типа (тексты, электронные документы, мультимедийные объекты), редактор разметки HTML, браузеры, программы, обеспечивающие протоколы TCP/IP, HTTP и др., сетевую операционную систему, инструменты для организации дискуссий, гипертекстовые СУБД и многие другие инструменты, обеспечивающие дружественный интерфейс пользователя с Internet.

Web-технология заключается в следующем. Пользователь посредством редактора HTML создает гипертекстовый документ (ГТ-документ). Он размещается на Web-сервере. Администратор делает ссылку в каталоге Web-сервера на Web-страницу, чтобы браузер смог ее найти. Любой другой пользователь

посредством браузера может получить доступ к данной Web-странице.

Разработано множество браузеров. Примером могут служить Navigator Netscape или Mosaic. Navigator Netscape обеспечивает простоту просмотра Web-страниц и не требует общей базы данных для разных пользователей. Написанный на языке Java искатель HotJava, помимо навигации, обеспечивает свободную миграцию программ.

Объектно-ориентированные языки Java и HotJava разработаны фирмой Sun Microsystems. Появление языка Java произвело переворот в области создания, применения и распространения программ и мультимедийной информации. Программы, написанные на языке Java, можно переносить на разные платформы, т. е. на разные типы компьютеров, выполнять в мультипроцессорных системах, загружать по Сети и выполнять другие программы в машине пользователя, а также загружать одновременно по Сети всем пользователям.

В настоящее время WWW — самый популярный и интересный сервис Internet, самое удобное средство работы с информацией. Самое распространенное имя для компьютера в Internet сегодня — WWW, больше половины потока данных Internet приходится на долю WWW. Количество серверов WWW нельзя оценить сколько-нибудь точно, но по некоторым оценкам их более 300 тыс. Скорость роста WWW даже выше, чем у самой сети Internet.

WWW работает по принципу «клиент-сервер», точнее — «клиент-серверы»: существует множество серверов, которые по запросу клиента возвращают ему гипермедийный документ — документ, состоящий из частей с разнообразным представлением информации, в котором каждый элемент может являться ссылкой на другой документ или его часть. Эти ссылки в документах WWW организованы таким образом, что каждый информационный ресурс в глобальной сети Internet однозначно адресуется, и документ, который вы читаете в данный момент, способен ссылаться как на другие документы на этом же сервере, так и на документы (и вообще на ресурсы Internet) на других компьютерах Internet. Причем пользователь не замечает этого и работает со всем информационным пространством Internet как с единым целым. Ссылки WWW указывают не только на до-

кументы, специфичные для самой WWW, но и на прочие сервисы и информационные ресурсы Internet. Более того, большинство программ-клиентов WWW не просто понимают такие ссылки, но и являются программами-клиентами соответствующих сервисов: FTP, Gopher, сетевых новостей Usenet, электронной почты и т. д. Таким образом, программные средства WWW являются универсальными для различных сервисов Internet, а сама информационная система WWW играет интегрирующую роль.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ТЕМАМ 4.1—4.7

1. **По каким признакам классифицируют информационные технологии?**
2. **В чем отличие новых (современных) ИТ от традиционных?**
3. **Каковы назначение и основные функции текстового редактора?**
4. **Что вы знаете о графических редакторах?**
5. **Что такое электронная таблица?**
6. **Как вы представляете базу данных?**
7. **Какие способы организации баз данных вы знаете?**
8. **Раскройте понятие СУБД.**
9. **Дайте определение понятия «мультимедиа».**
10. **Расскажите о технических средствах мультимедийных технологий.**
11. **Каково назначение вычислительных сетей?**
12. **Дайте классификацию ВС.**
13. **Изобразите схемы построения ВС.**
14. **Как осуществляется передача данных в сетях ЭВМ?**
15. **Что такое коммутация каналов?**
16. **Перечислите преимущества и недостатки коммутации сообщений и пакетов.**
17. **Какие виды интерфейсов вы знаете?**
18. **В чем назначение операционной системы сети?**
19. **Расскажите об электронной почте.**
20. **Какие признаки положены в основу классификации ЛВС?**
21. **Каким образом организуется обмен информацией в ЛВС?**
22. **Перечислите сходства и различия между локальными и глобальными вычислительными сетями.**
23. **Дайте характеристику методам доступа в ЛВС.**
24. **В чем состоит отличие моделей «файл-сервер» и «клиент-сервер»?**
25. **Назовите отечественные и зарубежные ЛВС.**
26. **Охарактеризуйте основные методы защиты информационно-вычислительных ресурсов сети от несанкционированного доступа.**
27. **Какие программные средства применяются для обеспечения защиты в вычислительных сетях?**

28. **Что такое компьютерный вирус?**
29. **Каковы задачи службы безопасности вычислительных сетей?**
30. **Раскройте содержание механизма обеспечения защиты информации в вычислительных сетях.**
31. **Что нового дают пользователям глобальные информационные сети?**
32. **Какие информационные ресурсы содержит Internet?**
33. **Дайте характеристику основных ресурсов Internet.**
34. **Раскройте понятие «протокол».**
35. **Объясните иерархию протоколов Internet.**
36. **Расскажите о схеме «клиент-сервер» для информационных серверов Internet.**
37. **Расшифруйте аббревиатуру WWW.**
38. **Что такое гипертекст?**
39. **Как организована информация в WWW?**
40. **Что такое Web-сервер и Web-страница?**
41. **Как организована электронная почта?**
42. **Дайте характеристику почтового сервера.**

Тема 4.8

ТЕКСТОВЫЙ ПРОЦЕССОР WORD

4.8.1. Основы работы с текстовым процессором

Самый популярный текстовый процессор Microsoft Word имеет мощные и полезные возможности, благодаря которым можно создать любой документ, будь то простая служебная записка, Web-документ или 500-страничный отчет.

Запуск программы. Чтобы запустить Word в Windows, выполните следующие действия:

— щелкните на кнопке *Пуск;*

— в появившемся меню выберите пункт *Программы;*

— в следующем меню выберите: Microsoft Word.

Другой способ запустить Word — с помощью ярлыка Microsoft Word на *Рабочем столе*.

Знакомство с рабочим окном программы Word. Открыв Word, вы увидите окно пустого документа (см. рис. 4.1), в которое можно вводить текст.

При работе необходимо сообщать Word, что именно нужно сделать. Для этого существует несколько способов:

— выбрать необходимый пункт в основном меню с помощью мыши или активизировать меню, нажав клавишу *<Alt>,* и выполнить команду с помощью клавиатуры;

— вызвать необходимое действие с помощью «горячих» клавиш. Например, чтобы открыть файл, следует нажать комбинацию клавиш *<Ctrl > + <o>;*

— выбрать необходимое действие с помощью контекстного меню, вызвать которое можно щелчком правой кнопкой мыши;

— выполнить команду, нажав на соответствующую кнопку на панели инструментов.

Настройка внешнего вида рабочего окна Word. После загрузки программы Word на экране появится стандартный вид окна, который представлен на рис. 4.1. Но его можно изменить, выбрав соответствующий пункт в меню *Вид,* установить или убрать некоторые панели инструментов (обычно установлены панели *Стандартная* и *Форматирование*) и линейки, установить необходимый масштаб. Масштаб можно быстро изменить, выбрав нужный из списка элемента *Масштаб* на панели инструментов *Стандартная*.

Разделение экрана. При желании можно разделить рабочую область окна на две части, чтобы были видны разные фрагменты одного документа одновременно. Каждая часть прокручивается независимо от другой и имеет свою собственную полосу прокрутки. При этом можно редактировать документ в любой части окна.

Чтобы разделить рабочую область окна:

— выполните команду меню *Окно → Разделитель.* Посередине рабочей области появится горизонтальная полоса;

— установите ее в нужное место и зафиксировуйте левой кнопкой мыши.

Чтобы убрать разделение, выполните команду меню *Окно → Снять разделение.*

Переход из одной части окна в другую осуществляется щелчком в нужной части или клавишей *<Tab>*. Чтобы изменить размер частей окна, указатель мыши устанавливают на полосу разделения и перетаскивают ее на новое место.

4.8.2. Работа с текстом

Создание и редактирование текстового документа. Чтобы создать новый документ, следует или нажать клавишу *Создать* на панели инструментов *Стандартная,* или выбрать пункт *Создать* в меню *Файл.*

Для ввода текста или вставки в документ рисунка, таблицы и т. д. необходимо установить курсор в ту строку, где должен размещаться объект, и либо ввести текст с клавиатуры, либо дать команду Word вставить рисунок, таблицу и т. д. При наборе текста, когда вы достигнете правого поля страницы, Word автоматически перейдет на новую строку. Нажимайте клавишу <*Enter*> только тогда, когда необходимо начать новый абзац или вставить пустую строку.

Если щелкнуть на кнопке *Непечатаемые знаки* на панели инструментов *Стандартная,* то на экране появятся специальные символы конца абзаца. Они показывают, где точно начинается и заканчивается абзац в вашем документе (обычно эти символы невидимы). Щелкните на этой кнопке еще раз, чтобы скрыть символы конца абзаца. Для слияния двух абзацев в один нужно просто удалить этот символ.

Для удаления символов используют клавиши:

— <*Backspace*>, чтобы удалить символы слева от курсора;

— <*Delete*>, чтобы удалить символы справа от курсора.

Для перемещения курсора по документу можно использовать клавиши со стрелками или полосу прокрутки либо щелкнуть левой кнопкой мыши в нужном месте.

Чтобы установить курсор:

— в начало или конец строки: нажмите соответственно клавишу <*Home*> или <*End*>;

— в начало или конец документа: нажмите соответственно комбинацию клавиш <*Ctrl*> + < *Home* > или <*Ctrl*> + <*End*>;

— влево или вправо на одно слово: нажмите соответственно комбинацию клавиш <*Ctrl*> + <←> или <*Ctrl*> + <→>;

— вверх или вниз на высоту страницы: нажмите соответственно клавишу <*Page Up*> или <*Page Down*>;

— в начало или в конец страницы: нажмите соответственно комбинацию клавиш <*Ctrl*> + <*Page Up*> или <*Ctrl*> + <*Page Down*>;

— в место последнего редактирования: нажмите комбинацию клавиш <*Shift*> + <*F5*> один или несколько раз.

Выделение текста. Часто при работе требуется выделить весь текст или его часть. Например, чтобы установить полужирный шрифт на заголовке, нужно его выделить, а затем щелкнуть на кнопке *Полужирный*. Текст или его фрагменты можно выделить с помощью мыши и клавиатуры.

Чтобы выделить с помощью мыши:

— слово: дважды щелкните на слове;

— несколько строк или весь текст: установите указатель мыши в начало текста, щелкните левой кнопкой и, удерживая ее, перемещайте указатель по тексту;

— предложение: нажмите клавишу <*Ctrl*> и, удерживая ее, щелкните где-нибудь на предложении;

— строку: щелкните слева от строки на полосе выделения;

— абзац: дважды щелкните слева от абзаца на полосе выделения;

— весь текст: нажмите клавишу <*Ctrl*> и, удерживая ее, щелкните слева от текста на полосе выделения.

Чтобы выделить с помощью клавиатуры:

— фрагмент текста: установите курсор в начало выделяемого блока, нажмите клавишу <*Shift*> и, удерживая ее, выделите текст с помощью клавиш перемещения курсора;

— весь документ: нажмите комбинацию клавиш <*Ctrl*> + <*5*> на малой цифровой клавиатуре.

Удаление, копирование и перемещение фрагментов документа. Фрагментом документа может быть текст, объект, рисунок и т. д.

Чтобы удалить фрагмент, нужно выделить его и нажать клавишу <*Delete*>.

Чтобы скопировать фрагмент, следует выделить его, а затем:

— выполнить команду меню *Правка* → *Копировать* (при этом выделенный объект сохраняется в буфере обмена операционной системы) или нажать комбинацию клавиш <*Ctrl*> + <*Insert*>;

— установить курсор там, куда необходимо скопировать фрагмент;

— выполнить команду меню *Правка* → *Вставить* (при этом объект можно вставлять несколько раз, пока он находится в буфере) или нажать комбинацию клавиш <*Shift*> + <*Insert*>.

Чтобы переместить фрагмент, необходимо выделить его, а затем:

— выполнить команду меню *Правка* → *Вырезать* (при этом объект также сохраняется в буфере обмена и одновременно удаляется с экрана);

— установить курсор там, куда необходимо переместить фрагмент;

— выполнить команду меню *Правка* → *Вставить* или нажать комбинацию клавиш <*Shift*> + <*Insert*>.

Для копирования и перемещения можно также использовать кнопки на панели инструментов *Стандартная*.

Еще один способ — использование контекстного меню, вызвать которое можно после выделения фрагмента, щелкнув правой кнопкой мыши.

Исправление неправильных команд. Если выполнена какая-либо неправильная команда (например, вы удалили не то, что нужно, и т. д.), то всегда можно отменить последнее действие. Для этого:

— или выполните команду *Правка → Отменить;*

— или щелкните на кнопке *Отменить ввод* на панели инструментов *Стандартная*.

Сохранение документа и выход из программы. При создании документа в Word по умолчанию он временно сохраняется в памяти компьютера под именем «Документ №», где № — порядковый номер, начиная с первого. Но лучше всего присвоить ему имя, особенно если вы работаете в сети.

Чтобы сохранить документ:

— выполните команду *Файл → Сохранить как...;*

— в появившемся диалоговом окне выполните действия, указанные на рис. 4.7;

— после выполненных действий файл сохранится на диске и в папке, которые вы указали.

Если ваш документ уже был ранее сохранен, то после внесения изменений его можно сохранить снова. Для этого:

— либо выполните команду меню *Файл → Сохранить;*

— либо щелкните на кнопке *Сохранить* на панели инструментов *Стандартная*.

Выход из программы. Чтобы выйти из программы:

— или выполните команду меню *Файл → Выход;*

— или щелкните на кнопке *Закрыть* в правом верхнем углу экрана;

— или нажмите комбинацию клавиш *<Alt> + <F4>;*

— или выполните двойной щелчок на системном меню.

Установка защиты и автосохранение. Если требуется защитить ваш документ от несанкционированного просмотра, то перед его сохранением:

— выполните команду меню *Сервис → Параметры;*

— выполните действия, указанные на рис. 4.8.

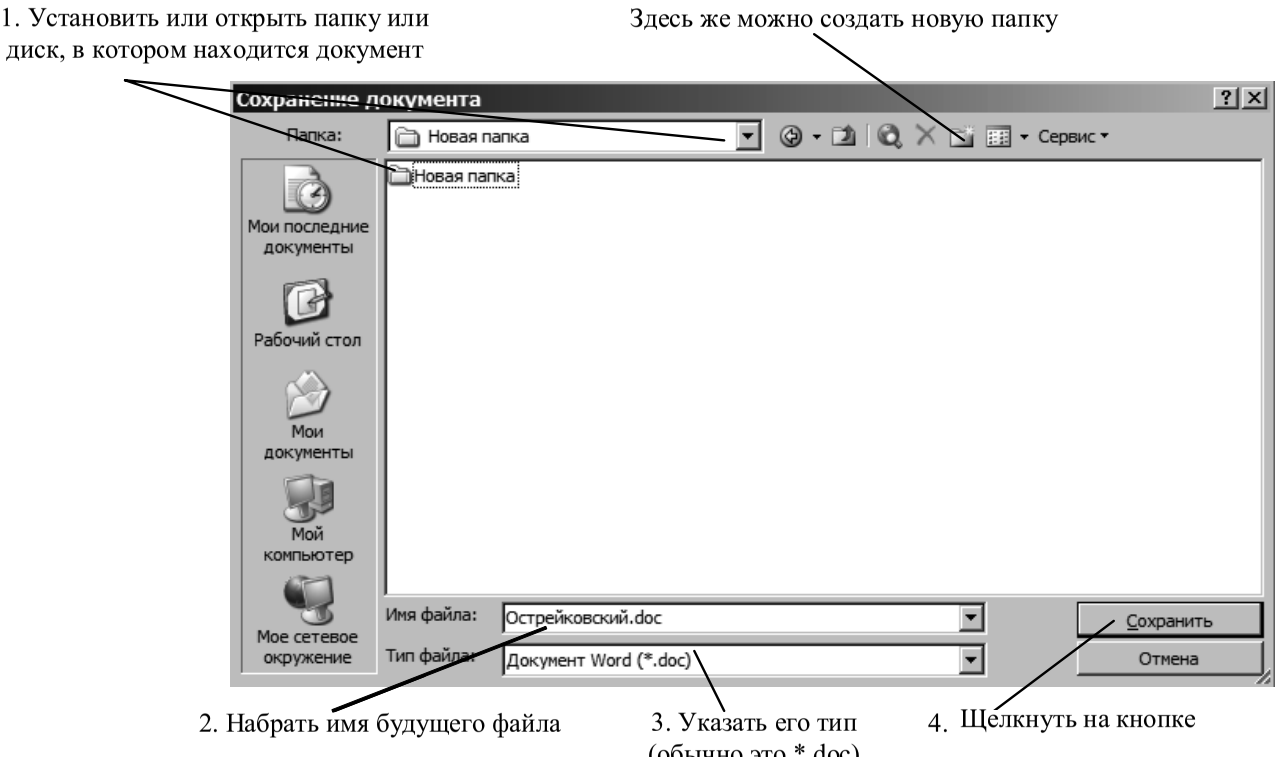


Рис. 4.7. Сохранение документа

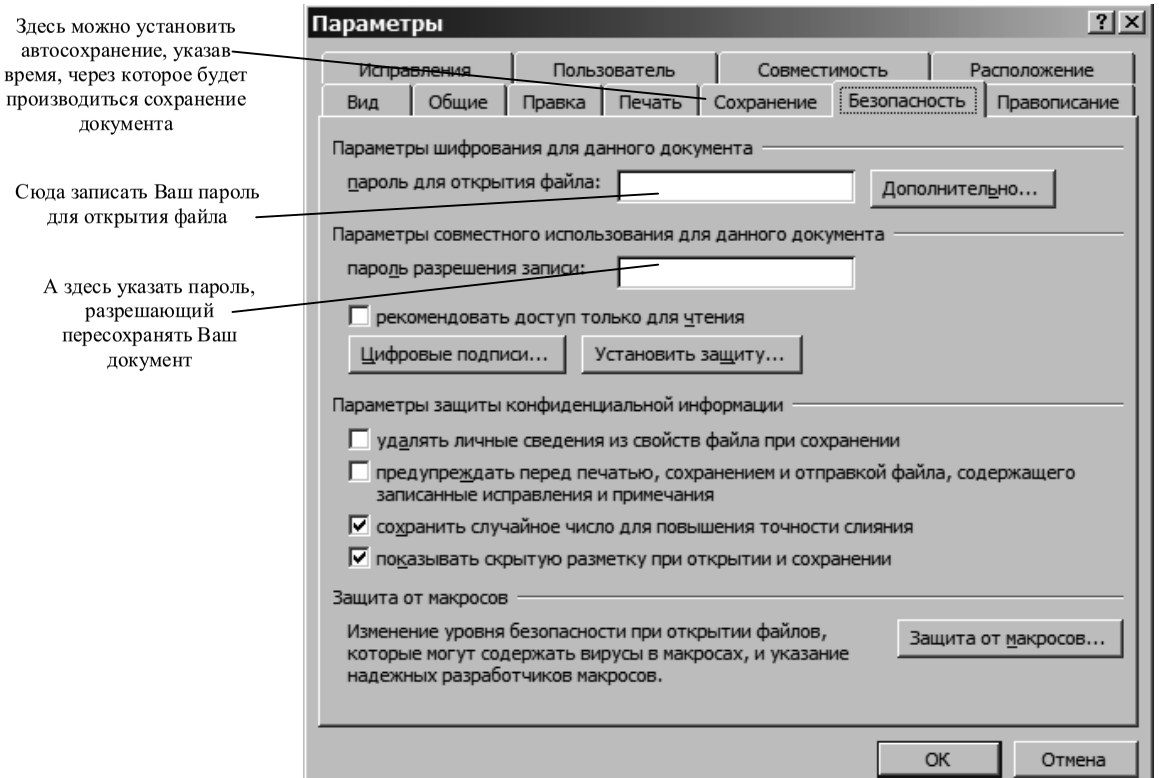


Рис. 4.8. Установка паролей при сохранении документа

Открытие документа. С помощью Word вы можете открыть любой документ, созданный в Word или других текстовых редакторах. Для этого:

— выполните команду *Файл → Открыть* либо щелкните на кнопке *Открыть* на панели инструментов С*тандартная;*

— в появившемся диалоговом окне выполните действия, указанные на рис. 4.9.

Вызов справки. Существует два вида помощи:

— вызвать *Помощника,* который дает советы в процессе работы. Для этого щелкните на кнопке *Справка по Microsoft Word* на панели инструментов *Стандартная;*

— нажать клавишу <*F1*>.

Изменение шрифта. Word располагает широким выбором шрифтов, которые можно использовать в документах. Каждый шрифт определяется видом его символов. Существует ряд стандартных шрифтов, например Courier, Arial, Times New Roman, Bookman Old Style и т. д. Можно изменить тип шрифта, его размер (он указывается в пунктах (сокращенно — пт): 1 пт = $^1/_{72}$ дюйма, 1 дюйм ≈ 25,5 мм), начертание и многое другое.

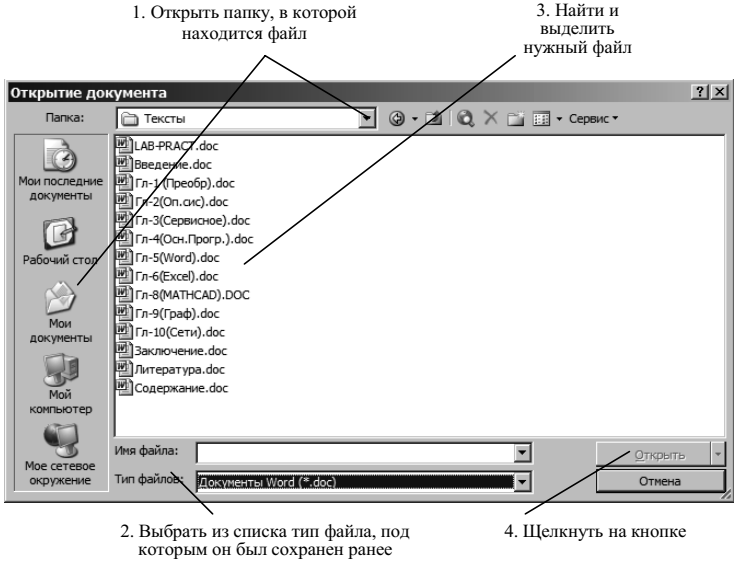


Рис. 4.9. Открытие документа

Чтобы изменить тип шрифта, вначале надо выделить фрагмент текста (слово, строку, абзац и т. д.). Если текст еще не набран, то нужно установить курсор там, где он будет размещаться, а затем:

— выполнить команду *Формат → Шрифт;*

— в диалоговом окне выполнить необходимые установки (рис. 4.10).

Для быстрого изменения типа, начертания, размера и стиля шрифта можно использовать панель инструментов *Форматирование*.

Настройка панелей инструментов. Чтобы добавить в панель инструментов дополнительные кнопки и придать им нужные команды:

— выполните команду меню *Вид → Панели инструментов → Настройка;*

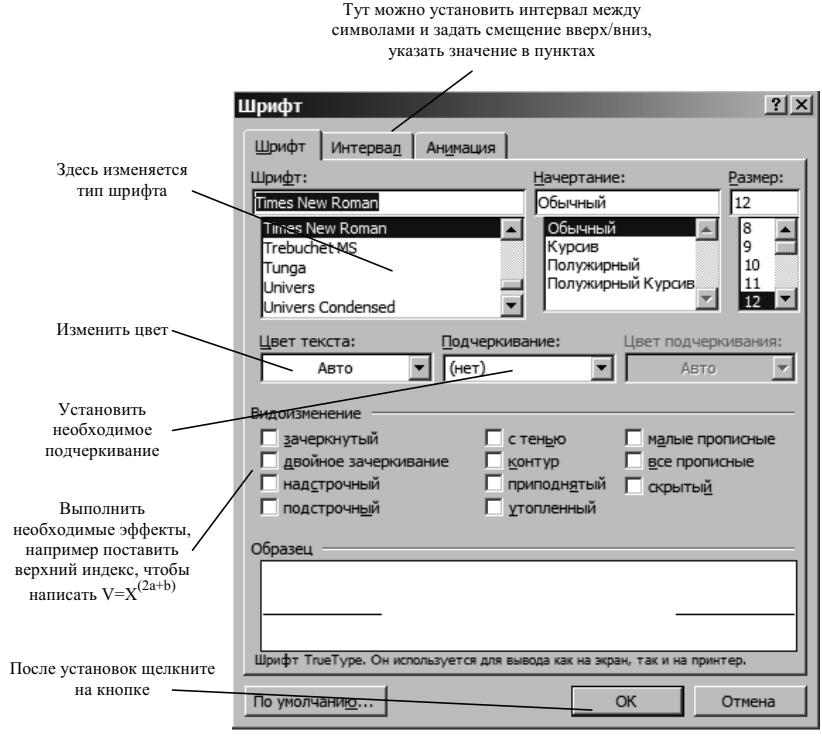


Рис. 4.10. Изменение шрифта

— в диалоговом окне переключитесь на вкладку *Команды* и из списка *Категории* выберите нужную, например *Формат;*

— в этой же вкладке из списка *Команды* выделите нужную команду, щелкнув левой кнопкой мыши, и, не отпуская ее, переместите курсор на панель инструментов в то место, где вы хотели бы установить кнопку с этой командой; затем отпустите левую кнопку мыши. После этого кнопка-команда должна остаться в указанном месте.

Чтобы удалить кнопки из панели инструментов:

— на соответствующей панели (или на удаляемой кнопке) щелкните правой кнопкой мыши;

— в контекстном меню выберите: *Настройка;*

— в диалоговом окне переключитесь на вкладку *Команды;*

— захватив левой копкой мыши удаляемую кнопку, перетащите ее в сторону диалогового окна.

Обрамление и фоновые узоры. Для быстрого обрамления выделенного текста или любого другого объекта (рисунка, таблицы и т. д.) найдите на панели инструментов *Форматирование* соответствующую кнопку, откройте предложенный список (рис. 4.11) и выберите тип обрамления.

Чтобы снять обрамление, выделите фрагмент и щелкните на соответствующей кнопке (рис. 4.11).

Границы и заливка. Эта возможность Word используется для более качественной настройки рамок и заливки внутри выделенных фрагментов. Вначале фрагмент выделите, а затем:

— выполните команду меню *Формат → Границы и заливка* и щелкните на ярлычке вкладки *Граница;*

— выберите желаемый тип рамки и стиль линии из списка *Тип,* нужный цвет — из списка *Цвет* и нужную толщину линий — из списка *Ширина*; установите область применения.

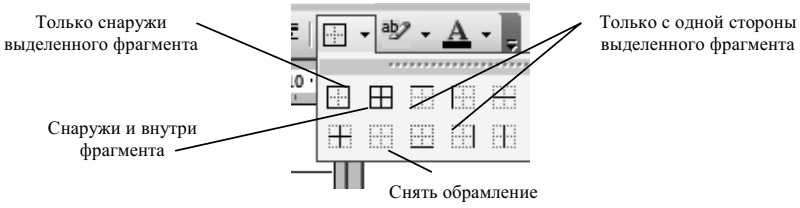


Рис. 4.11. Обрамление фрагментов текста

Закладка *Страница* предназначена для установки рамки на всю страницу. При этом вместо линии можно выбрать рисунок из списка *Рисунок*.

Если вы хотите как-то обозначить фрагмент текста, то следует применить заливку к этому фрагменту. Для этого вначале необходимо выделить текст, а затем:

— выполнить команду меню *Формат → Границы и заливка* и щелкнуть на закладке *Заливка;*

— выбрать тип узора, а потом цвет фона.

Чтобы все убрать, воспользуйтесь этим же диалоговым окном.

Форматирование абзаца. **Отступы и выравнивание текста**. При наборе текста обычно Word автоматически переходит на новую строку, когда вы достигнете правого поля. Текст при этом разбивается по словам в месте пробела или дефиса. Чтобы предотвратить это разбиение, вместо обычного вставьте *Неразрывный пробел* (<*Ctrl*> + <*Shift*> + <*пробел*>) или *Неразрывный дефис* (<*Ctrl*> + <*Shift*> + <-> (*дефис*)).

Если требуется начать новый абзац, нажмите клавишу <*Enter*>.

Отступ — это расстояние между краем текста абзаца и полями всего документа. Чтобы быстро установить отступ, лучше всего использовать горизонтальную линейку (рис. 4.12).

Если необходимо установить отступ для одного абзаца, поместите курсор в любое место на нем.

Если вы устанавливаете отступ для нескольких абзацев, выделите их, а затем:

— или перетащите метки отступов на линейке в нужные позиции;

— или щелкните на кнопке *Увеличить отступ* на панели инструментов *Форматирование;* при этом отступ изменится на $^1/_2$ дюйма;

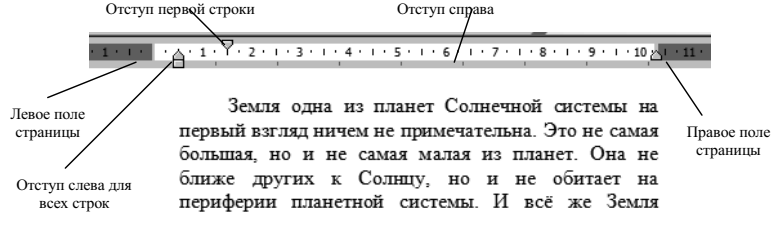


Рис. 4.12. Форматирование отступов

— или просто нажмите клавишу <*Tab*>.

Для более точных настроек выполните команду меню *Формат → Абзац,* чтобы открыть диалоговое окно, показанное на рис. 4.13.

Выравнивание текста. Существует четыре вида выравнивания текста относительно отступов абзаца (ширины ячейки в таблице или ширины страницы, если отступы равны нулю): *По левому краю, Правому краю, По центру* и *По ширине*. *Выравнивание по ширине* означает выравнивание одновременно левого и правого края строк абзаца за счет вставки дополнительных пробелов между буквами.

Чтобы выровнять текст, установите курсор внутри абзаца, выделите нужный фрагмент и щелкните на нужной кнопке (*По левому краю, По центру* и т. д.) на панели инструментов *Форматирование:*

Поиск и замена фрагментов текста. Чтобы найти в вашем документе какой-либо фрагмент текста (слово, набор символов, символ и т. д.), можно дать команду, при которой Word автоматически просмотрит ваш документ целиком (или только выделенный фрагмент) в поисках указанного текста. Для этого:

— выполните команду меню *Правка → Найти,* затем введите искомое слово;

— если диалоговое окно сокращенное, щелкните на кнопке *Больше;*

— выберите направление поиска и необходимые установки. Например, если вы выбрали *Подстановочные знаки,* то в строке *Найти* можно указать шаблоны: *?* — произвольный один символ или * — несколько последовательных символов.

Чтобы найти и заменить в вашем документе какой-либо фрагмент текста, выполните команду *Правка → Заменить,* затем введите искомое слово и далее следуйте командам диалогового окна.

Проверка орфографии. В Word имеется возможность проверять и исправлять написание слов. При этом Word сравнивает слова в документе со словарем; если встречаются неизвестные слова или слова с ошибками, то они выделяются. Можно пропустить выделенное слово, отредактировать его или занести в словарь.

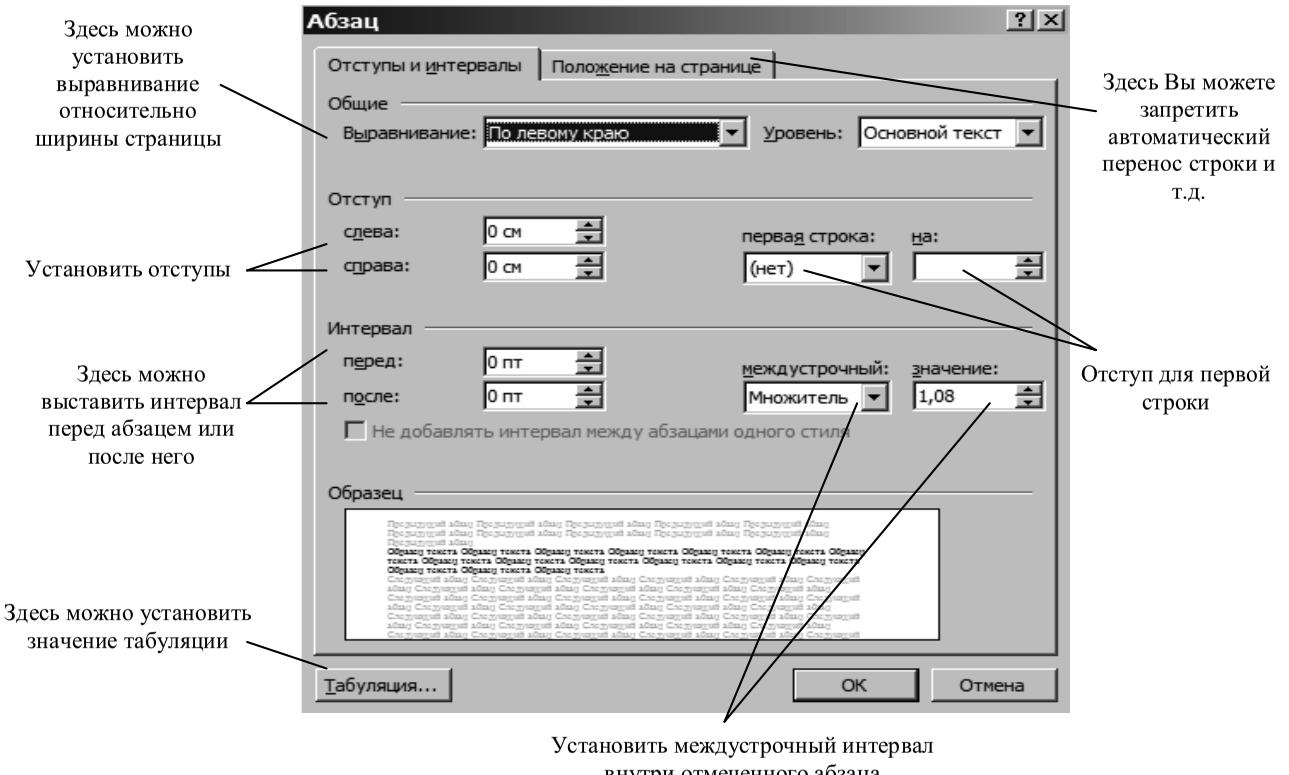


Рис. 4.13. Форматирование абзаца

Текст проверяется, начиная с позиции, на которой установлен курсор. Чтобы проверить орфографию всего документа или выделенной части:

— или выполните команду меню *Сервис → Правописание;*

— или нажмите клавишу *<F7>;*

— или щелкните на кнопке *Правописание* на панели инструментов *Стандартная.*

Использование тезауруса. При наборе текста, чтобы избежать повторений слов или просто обогатить свой лексикон, можно использовать *Тезаурус* — словарь синонимов и антонимов. Чтобы применить *Тезаурус:*

— установите курсор на интересующем вас слове в документе;

— выполните команду *Сервис → Язык → Тезаурус* или нажмите комбинацию клавиш *<Shift> + <F7>;*

— в диалоговом окне (рис. 4.14) выполните предлагаемую последовательность действий.

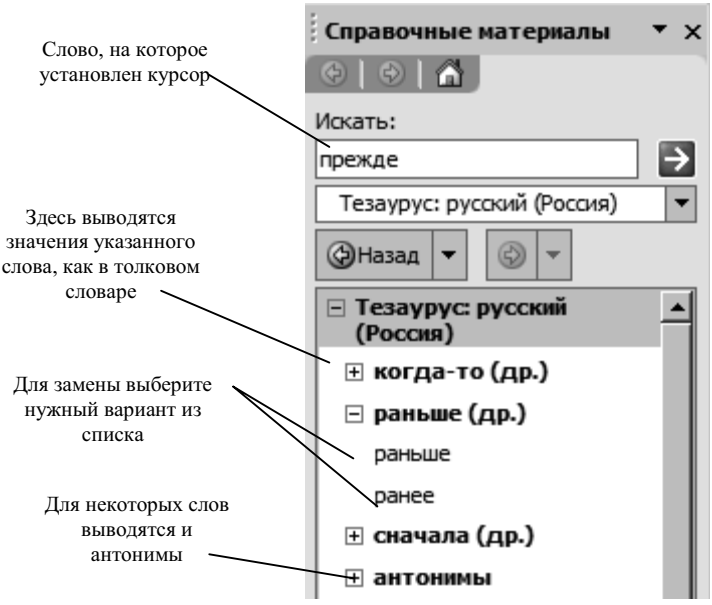


Рис. 4.14. Определение синонимов и антонимов слов

Автозамена — это средство, которое автоматически находит ошибки при наборе текста и исправляет их. В Word есть множество элементов *Автозамены,* используемых по умолчанию, но можно дополнить этот список своими элементами. Текст *Автозамены* может быть обычным (т. е. отформатированным так же, как и текст абзаца, в который он будет вставлен) или же сохраняющим собственное форматирование.

Чтобы создать элемент *Автозамены,* выполните команду *Сервис → Автозамена* и после открытия диалогового окна установите необходимые флажки, например:

— *Исправлять ДВе ПРописные буквы в начале слова;*

— *Делать первые буквы предложений прописными;*

— *Устранять последствия случайного нажатия cAPS LOCK;*

— *Заменять при вводе.*

Далее см. рис. 4.15.

Автотекст — это средство, которое позволяет сохранять часто используемые текстовые блоки или графические объекты с определенным именем и вставлять их в документ по мере надобности или автоматически.

Чтобы создать элемент *Автотекста:*

— выделите объект (текст или рисунок), который станет элементом *Автотекста;*

— выполните команду *Вставка → Автотекст → Создать;*

— введите имя вашего *Автотекста.* Теперь он сохранится под этим именем.

Чтобы вставить *Автотекст:*

— установите курсор в том месте документа, где нужно поместить *Автотекст;*

— выполните команду *Вставка → Автотекст → Автотекст;*

— найдите в списке имя своего *Автотекста* и щелкните на кнопке *Вставить.*

Можно также использовать *Автотекст при Автозамене* (кроме рисунков). Для этого:

— выделите текст, который станет элементом *Автотекста;*

— выполните команду *Сервис → Автозамена;*

— в появившемся диалоговом окне выберите вкладку *Автотекст;*

— система сама заполнит поле *Имя элемента*, а в поле *Образец* можно просмотреть содержимое этого *Автотекста;*

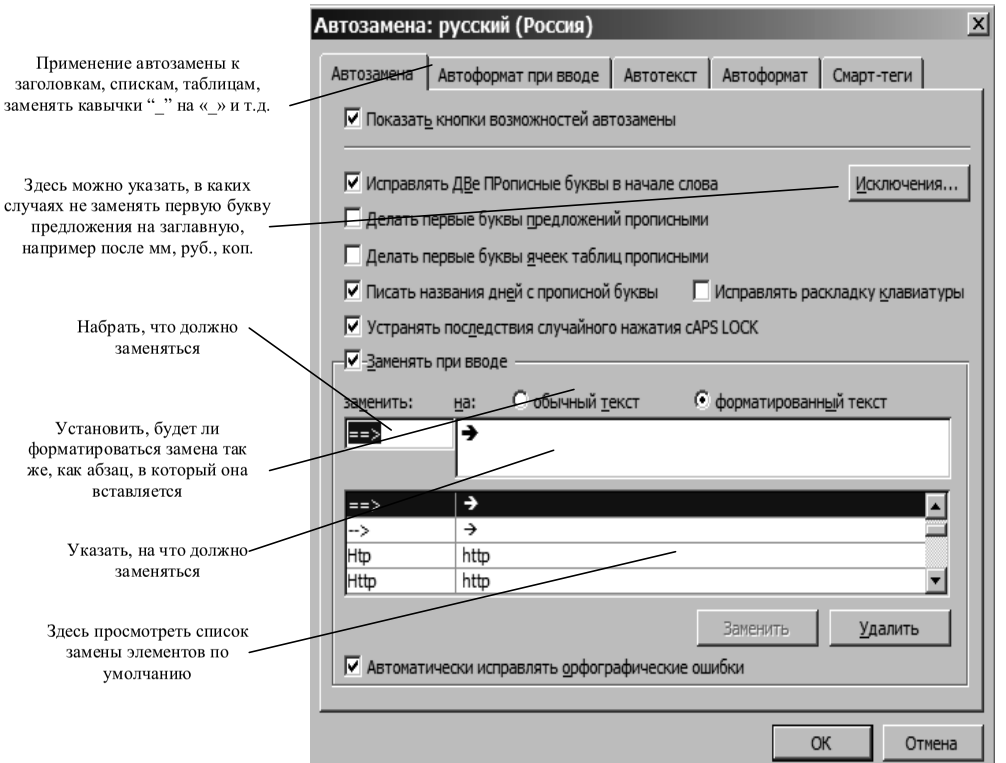


Рис. 4.15. Автозамена и Автотекст

— щелкните на кнопке *Добавить*.

Чтобы вставить такой *Автотекст:*

— в нужном месте документа наберите имя *Автотекста* (обычно оно совпадает с первыми словами его содержимого);

— после набора нескольких первых букв Word распознает и выведет на экран вводимый *Автотекст*. Чтобы вставить его в текст, надо нажать клавишу *<Enter>;* чтобы проигнорировать его, следует продолжить вводить текст.

Вставка символов. Чтобы вставить символ в документ:

— выполните команду меню *Вставка → Символ*. Откройте вкладку *Символы;*

— из раскрывшегося списка *Шрифт* выберите нужную группу символов: Symbol (греческий алфавит, стрелки, математические символы), Wingdings (🕮, ☎, 🕒, ✂ и т. д.), Times New Roman (¼, ©, ®, ¥, ¿ и т. д.), Webdings (🏂, 🚗, 🖋, 🖧 и т. д.);

— выбрав нужный символ, щелкните на кнопке *Вставить*. После вставки необходимых символов окно можно закрыть.

Чтобы вставить специальный символ, откройте вкладку *Специальные символы*. Здесь также можно вставить выбранный символ или посмотреть, какая комбинация клавиш нужна для его вставки.

Нумерация страниц, верхние и нижние колонтитулы. *Колонтитул* представляет собой одну или несколько строк, помещаемых в начале или конце каждой страницы документа. Колонтитулы обычно содержат номера страниц, название глав или параграфов, наименование и адрес фирмы и т. д. Они могут быть различными для четной и нечетной страниц, для первой страницы и последующих. Колонтитулы помогают ориентироваться в документе, а также могут служить дополнительным средством рекламы.

Чтобы пронумеровать страницы, выполните команду меню *Вставка → Номера страниц*. После этого установите необходимый формат нумерации.

Чтобы создать колонтитул, выполните команду меню *Вид → Колонтитулы* и следуйте рекомендациям рис. 4.16.

Чтобы установить колонтитул только на четные или только на нечетные страницы, выполните команду меню *Файл → Параметры страницы → Макет* и установите флажок *Различать колонтитулы четных и нечетных страниц*. После этого по-

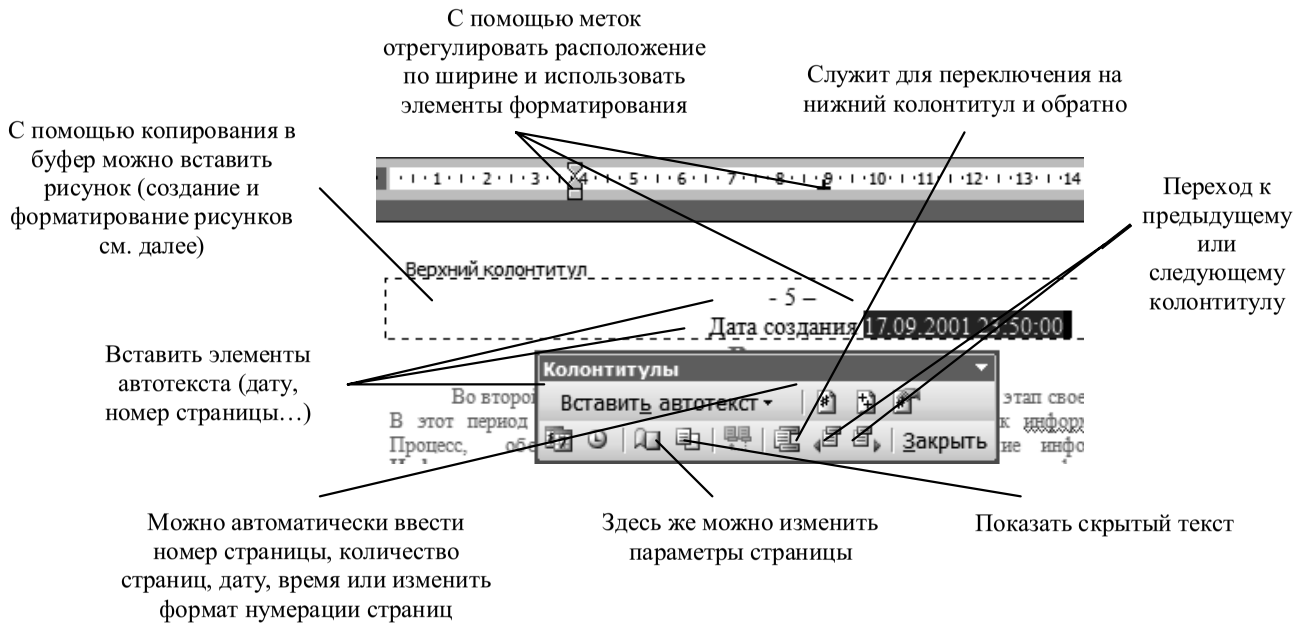


Рис. 4.16. Установка колонтитулов

местите курсор на страницу с нужным номером (четную/нечетную) и создайте колонтитул. Он автоматически установится только на указанных (четных или нечетных) страницах.

Создание и использование стилей. *Стиль* — это набор параметров форматирования, имеющий свое имя. Например, стиль может включать в себя шрифт Arial размером 12 пунктов, отступ величиной 1 дюйм, двойной междустрочный интервал и выравнивание по обоим краям. Определив стиль, можно быстро применить его к любому тексту документа. В Word есть несколько определенных стилей, но можно создавать и собственные. Различают два типа стилей: стиль абзаца и стиль символа.

Чтобы установить стиль абзаца, установите курсор где-нибудь в его пределах. Если абзацев несколько, выделите необходимые абзацы. Затем на панели инструментов *Форматирование* откройте список стилей и выберите подходящий. Имейте в виду, что напротив названия стиля абзаца стоит значок «¶», а напротив стиля — значок «**a**».

Чтобы создать или изменить стиль:

— выполните команду меню *Формат → Стиль;*

— щелкните на кнопке *Создать* или *Изменить*.

Создание колонок в документе. В Word есть возможность разбивать текст на колонки, которые обычно используются в газетных статьях. Можно применить форматирование колонок ко всему документу, к текущему разделу или к выделенному тексту. Если необходимо разбить на колонки только часть документа, то выделите те строки, которые будут в колонках, а затем:

— выполните команду *Формат → Колонки;*

— в области окна *Тип* выберите формат и количество колонок;

— в поле *Применить* укажите, к какой части документа необходимо применить разбиение на колонки (ко всему документу, к текущему разделу, к выделенному разделу, до конца документа);

— если необходимо, установите флажок *Разделитель* для разделения колонок;

— также можно установить ширину для каждой колонки и расстояние между ними (если колонок несколько) или уста-

новить флажок *Одинаковая ширина* для колонок равной ширины;

— подтвердите установки щелчком на кнопке *OK*.

Чтобы быстро разбить текст на колонки, воспользуйтесь панелью инструментов. Чтобы убрать разбиение на колонки, выделите их и установите одну колонку.

4.8.3. Работа с *Редактором формул*

Если нужно набрать в вашем документе какую-нибудь формулу, лучше всего воспользоваться *Редактором формул*. Для этого:

— установите курсор в том месте документа, где должна быть формула;

— выполните команду *Вставка → Объект,* а затем выбрать вкладку *Создание;*

— в списке *Тип объекта* выберите *Microsoft Equation 3.0;*

— в появившемся окне редактирования формул выберите шаблон из нижнего ряда панели инструментов и заполните поля;

— из верхнего ряда панели инструментов выберите нужный символ и введите его в текст;

— чтобы вложить один шаблон в другой, установите курсор в нужное место в окне *Редактор формул,* а затем выберите нужный шаблон на панели инструментов;

— перемещаясь по элементам шаблона с помощью клавиш перемещения курсора или мыши, введите формулу.

Чтобы выйти из *Редактора формул,* просто щелкните за пределами окна редактирования.

Чтобы отредактировать набранную формулу, щелкните на ней два раза левой кнопкой мыши или выделите данный объект и выполните команду *Правка → Объект → Формула → Изменить* или команду *Открыть*.

В *Редакторе формул* можно записывать короткие фразы, не относящиеся к математическим обозначениям. Для таких записей удобно использовать режим стиля *Текст*. В этом режиме символы алфавита отформатированы, как обычный текст, а добавление интервалов между словами возможно с помощью

длинной клавиши (*Пробел*). Заметьте, чтобы ввести в математическом стиле пробел, необходимо выбрать из верхнего ряда панели инструментов соответствующий символ.

Чтобы добавить фразу:

— поместите курсор в начало нового текста;

— выполните команду *Текст* в меню *Стиль;*

— введите текст.

Чтобы вернуться к математическому стилю, выполните команду *Математический* в меню *Стиль.*

4.8.4. Работа с графическими объектами

Графическим объектом называют рисунок, который хранится на диске в файле графического формата. В Word есть возможность использовать графические файлы, созданные разными приложениями. Кроме этого, в Word существует библиотека рисунков, которые можно вставлять в документы. Также прямо в Word можно создавать простейшие рисунки, блок-схемы и т. д., используя панель инструментов *Рисование.*

Вставка рисунков из файла. Чтобы вставить рисунок из графического файла в документ:

— установите курсор в том месте, где нужно поместить рисунок;

— выполните команду *Вставка → Рисунок → Из файла;*

— выберите нужный рисунок и выполните команду *Добавить.*

Вставка картинок. Чтобы вставить картинку в документ, сначала установите курсор в том месте, где нужно ее поместить, и выполните команду *Вставка → Рисунок → Картинки.* Затем выберите нужную картинку.

Создание рисунка. Чтобы создать простейший рисунок или схему в вашем документе, выберите свободное место и щелкните на кнопке вызова панели инструментов *Рисование* (если ее еще нет на экране). После этого на экране появится панель инструментов *Рисование,* с помощью которой можно создать рисунок.

Создание надписи в рисунке. Если в вашем рисунке должен находиться текст, лучше всего создать элемент, который

называется *Надпись*. Для этого щелкните на кнопке *Надпись,* нарисуйте прямоугольник и введите в него текст, используя все элементы форматирования (выравнивание, шрифты и т. д.).

Форматирование элементов рисунка. После рисования каждого элемента вашего рисунка (надписи, линии, овалы, стрелки и т. д.) каждый из них можно дополнительно отдельно отформатировать, например убрать рамку у надписи, установить точно размеры элемента или настроить более качественно вид стрелки и т. д. Для этого, щелкнув правой кнопкой мыши, вызовите контекстное меню форматируемого элемента и выполните команду *Формат автофигуры.* В появившемся диалоговом окне произведите необходимые установки.

Расположение элементов и комбинирование их в один рисунок. После рисования и форматирования каждого элемента необходимо правильно расположить их между собой, особенно при наложении их друг на друга, и соединить все элементы в один рисунок.

При наложении элементов друг на друга один элемент можно расположить на переднем плане, а другой — на заднем; то же самое относится и к тексту. Для этого выделите один элемент и установите для него порядок видимости, выполнив команду *Действия → Порядок*.

Можно также выровнять или повернуть выделенные элементы, выбрав соответствующие команды.

Для соединения всех элементов в один рисунок (это нужно для того, чтобы в дальнейшем все действия можно было применять к общему рисунку, например копировать, изменить его размер) выделите их (удерживая нажатой клавишу *<Shift>,* щелкайте последовательно на каждом элементе) и выполните команду *Группировать* в контекстном меню.

Копирование, перемещение, изменение размеров и форматирование объекта. Объектом может служить созданный рисунок, вставленное из файла изображение и т. д.

Чтобы скопировать объект, нужно его выделить и занести в буфер обмена любым из рассмотренных ранее способов (см. «Удаление, копирование и перемещение фрагментов документа»). После этого объект можно вставить в любое место текуще-

го документа или в другой документ, например открытый в графическом редакторе, для последующих изменений.

Чтобы переместить объект, выделите его и, не отпуская левую кнопку мыши, передвигайте в нужном направлении.

Чтобы изменить размеры объекта, выделите его и с помощью мыши, используя курсоры (<↔> и др.) на границах объекта, передвигайте в нужном направлении.

При форматировании созданного рисунка вызовите его контекстное меню и выполните команду *Формат объекта.* В появившемся диалоговом окне сделайте необходимые установки уже для всего рисунка в целом. Например, установите нужное обтекание его текстом или уточните его размеры.

Если вы вставили рисунок из файла или из буфера обмена, например скопированный из другого приложения, то при выделении данного объекта появляется панель инструментов *Настройка изображения* (если она не появилась, выполните команду *Вид → Панели инструментов → Настройка изображения*), в которой, кроме рамок и обтекания текстом, можно настроить яркость, контрастность и др.

4.8.5. Работа с таблицами

Таблица позволяет упорядочивать данные в виде строк и столбцов. Каждый элемент таблицы называется **ячейкой**. Информация, хранимая в ячейке, не зависит от других элементов. Поэтому всегда можно изменить форматирование и размер каждой ячейки, а также информацию в ней. Ячейка таблицы может содержать, кроме другой таблицы, текст, рисунок и другие объекты.

Чтобы вставить таблицу в документ:

— установите курсор там, где нужно разместить таблицу;

— выполните команду *Таблица → Вставить → Таблица;*

— в появившемся диалоговом окне введите количество строк и столбцов будущей таблицы;

— при желании здесь же можно установить автоформатирование таблицы, щелкнув на кнопке *Автоформат,* и выбрать подходящий формат из списка (использование автоформатирования будет рассмотрено далее).

После этого заданная таблица появится на экране.

Выделение ячеек, заполнение и редактирование таблицы. При вводе текстовой информации, если она не умещается по ширине, текст автоматически переносится по словам, т. е. высота строки увеличивается. Вводимая информация располагается внутри каждой ячейки относительно ее полей, которые можно всегда изменить, как и поля страницы. Перейти с одной ячейки на другую можно с помощью клавиши <*Tab*> или с помощью мыши.

Чтобы быстро очистить ячейки таблицы, выделите их и нажмите клавишу <*Del*>.

Добавление и удаление строк и столбцов в таблице. Чтобы вставить строку, установите курсор в любое место строки, перед которой нужно вставить новую, а затем выполните команду *Таблица → Добавить строку.*

Чтобы вставить столбец, выделите тот столбец, перед которым будет вставлен новый, а затем выполните команду *Таблица → Добавить столбец.* Если требуется добавить столбец справа от таблицы, то перед этим выделите правый крайний мнимый столбец.

Чтобы удалить строку (столбец), сначала выделите ее, а затем выполните команду *Таблица → Удалить столбец/строку.* Можно удалить ячейку или группу ячеек; для этого их нужно выделить и выполнить команду *Таблица → Удалить ячейки,* а затем в появившемся окне указать, в какую сторону сдвинуть остальные ячейки, чтобы заполнить освободившуюся область.

Форматирование информации внутри ячеек. При создании определенного формата на ячейку он автоматически устанавливается и на то, что находится в этой ячейке. Например, если мы выделим всю таблицу и установим размер шрифта: 12, то в любом месте таблицы шрифт будет равен этому значению.

При форматировании текста в ячейке можно использовать все возможности Word, связанные с форматированием шрифта, абзаца и т. д.

Чтобы расположить текст внутри ячейки по горизонтали, выберите с помощью кнопок панели инструментов тип вырав-

нивания (*По левому краю, По правому краю, По ширине* или *По центру*), но сначала выделите нужные ячейки.

Чтобы расположить текст внутри ячейки по вертикали, выделите ячейки, вызовите контекстное меню и выполните команду *Направление текста*.

Обрамление и установка фоновых узоров таблицы. При вставке таблицы в документ она автоматически обрамляется снаружи и внутри одинарной тонкой линией, толщина которой обычно составляет 0,5 пт. При желании толщину линии можно изменить: установить другую толщину, или выбрать другой стиль линии, или совсем ее убрать.

Для быстрого обычного обрамления ячеек необходимо их выделить и воспользоваться кнопкой *Внешние границы* на панели инструментов *Форматирование*.

Если обрамление таблицы убрать, то на экране останется сетка (при печати на принтере она будет невидимой), которую также можно удалить, выполнив команду *Таблица → Скрыть сетку*.

Для более точной настройки границ таблицы:

— выделите таблицу или нужные ячейки;

— выполните команду *Формат → Границы и заливка;*

— выберите вкладку *Границы;*

— выполните необходимые установки, руководствуясь рис. 4.17.

Если нужно выделить некоторые ячейки вашей таблицы, то можно воспользоваться такой возможностью Word, как наложение фонового узора. Для этого:

— выделите группу ячеек (это может быть строка или столбец);

— выполните команду *Формат → Границы и заливка;*

— выберите вкладку *Заливка;*

— в поле *Тип узора* установите его тип и процентное содержание;

— в поле *Цвет фона* выберите нужный цвет;

— в поле *Применить* установите: *Ячейки*.

Автоматическое форматирование таблиц. Чтобы не выделять элементы таблицы различными цветами, фонами и границами вручную, Word предлагает множество заготовленных форма-

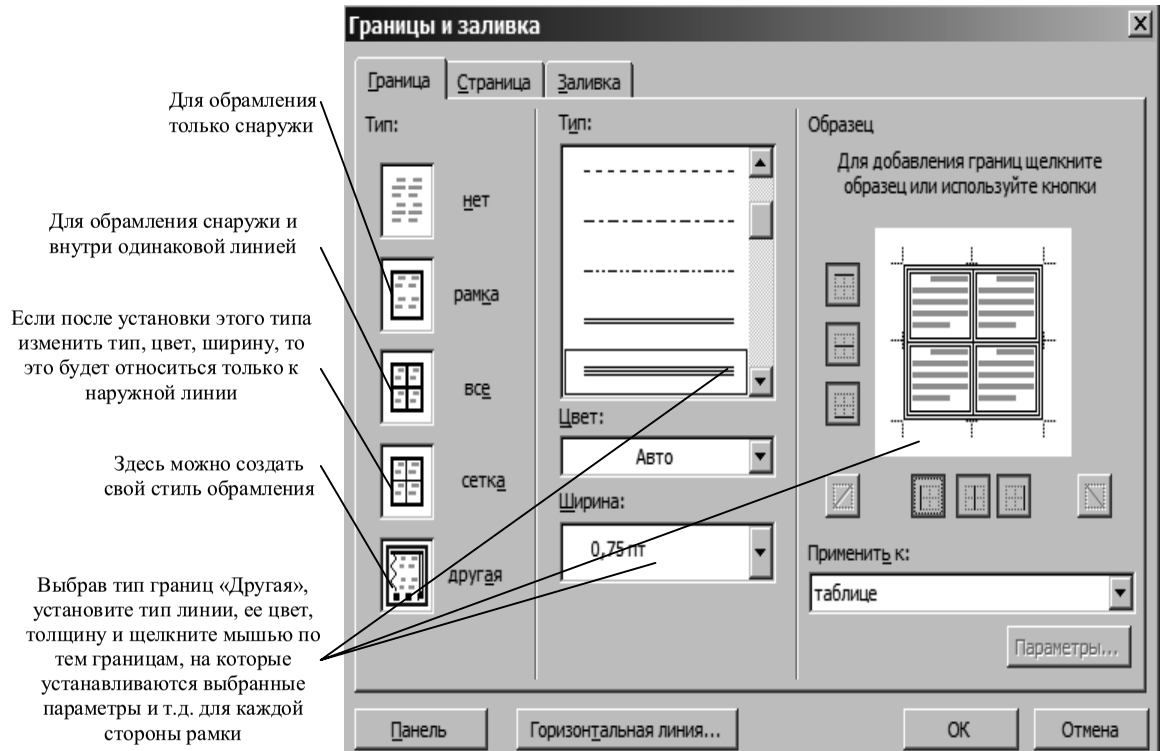


Рис. 4.17. Установка обрамления в таблице

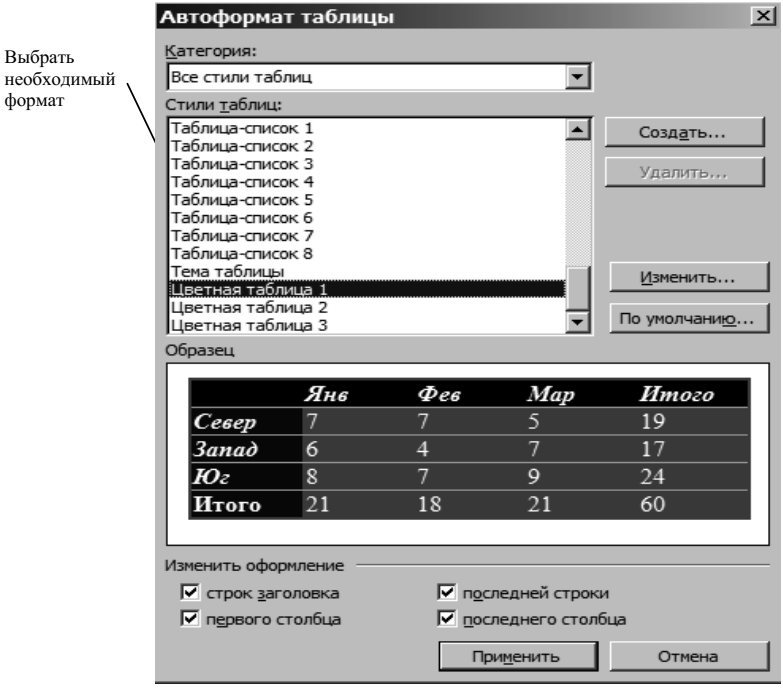


Рис. 4.18. Установка автоформата на таблицу

тов, с помощью которых можно легко изменить внешний вид таблицы. Для этого:

— поместите курсор в любую ячейку таблицы;

— выполните команду *Таблица → Автоформат;*

— в появившемся диалоговом окне введите необходимые установки, руководствуясь рис. 4.18.

4.8.6. Обработка числовой информации в таблицах

Word позволяет не только располагать в табличной форме числовые данные, но и производить вычисления. Чтобы произвести вычисления в таблице:

— выделите ячейку, в которую будет помещен результат;

— выполните команду меню *Таблица → Формула;*

— если в поле *Формула* предложена неподходящая формула, удалите ее;

— в списке *Вставить_функцию* выберите функцию. Например, для сложения чисел выберите: SUM.

— введите в формулу адреса ячеек. Например, для суммирования содержимого ячеек A1 и B4 введите формулу: =SUM(A1,B4);

— в поле *Формат числа* введите формат для чисел. Например, для отображения чисел в виде процентов выберите формат: 0,00 %.

При выполнении вычислений в таблицах ссылки на ее ячейки имеют вид: A1, A2, B1, B2 и т. д., где буква указывает столбец, а номер — строку.

Чтобы ссылаться на ячейки в формулах, используют запятую в качестве разделителя ссылок на отдельные ячейки и двоеточие — для разделения первой и последней ячеек, определяющих диапазон.

Чтобы ссылаться на ячейки в другой таблице или на ячейку из внешней таблицы, нужную таблицу помечают с помощью закладки. Например, формула: *=average(Table2 b:b)* усредняет значения в столбце B таблицы, помеченной закладкой *Table2.*

Чтобы назначить закладку:

— выделите элемент, которому следует назначить закладку;

— выполните команду меню *Вставка → Закладка;*

— в поле *Имя закладки* введите с клавиатуры или выберите нужное имя;

— щелкните на кнопке *Добавить.*

Если в вашей таблице присутствуют формулы, то в любой момент их можно просмотреть. Для этого выделите число, которое является результатом вычисления, и, щелкнув правой кнопкой мыши, вызовите контекстное меню, в котором выполните команду *Коды/значение полей.*

Здесь же можно обновить результат данного поля (например, после изменения данных).

4.8.7. Построение диаграмм

Чтобы создать диаграмму в документе:

— выделите таблицу;

— выполните команду *Вставка → Объект → Создание;*

— в списке *Тип объекта* выберите: *Диаграмма* Microsoft Graph 2000.

После этого система Microsoft Graph отображает диаграмму, а также таблицу MS Graph, содержащую связанные с ней данные. Кроме этого, на экране появится панель инструментов *Стандартная* для настройки диаграмм. После создания диаграммы можно ввести новые данные в таблицу MS Graph, импортировать данные из текстового файла, импортировать лист Microsoft Excel (табличный процессор Excel подробно рассмотрен в теме 4.9) или скопировать данные из другой программы. Сведения об особенностях организации данных при построении точечной диаграммы содержатся в справочной системе Microsoft Graph.

После выхода из системы Microsoft Graph диаграмма будет вставлена в текстовый документ как рисунок. Для ее редактирования нужно дважды щелкнуть левой кнопкой мыши на этом рисунке.

Вначале можно настроить внешний вид диаграммы (изменить ее тип, установить нужный объемный вид, цвет фона и т. д.), вызвав контекстное меню для области диаграммы. Когда область диаграммы настроена, можно настроить каждый элемент диаграммы отдельно. Для этого нужно его выделить, вызвать контекстное меню этого элемента и выполнить необходимые изменения.

ПРАКТИЧЕСКИЕ РАБОТЫ

Работа № 1. Операции с текстом

Цель работы: научиться создавать, сохранять и редактировать документы, применяя форматирование абзацев, страниц и текста.

Задание:

1. Создайте документ, в который скопируйте текст из справочной информации на тему «Выделение текста и рисунков с помощью мыши» и отформатируйте его.

2. Установите следующие параметры страницы: поля сверху и снизу — 1,5 см, слева — 3 см, справа — 2 см.

3. Отработайте различные способы быстрого выделения фрагментов текста (символов, слов, строк, абзаца, предложения) и всего текста.

4. Наберите следующее выражение: $F(x, y^{(k)}, y^{(k+1)}, ..., y^{(n)}) = 0$; установите для него интервал между символами 2 пт.

Методика выполнения работы:

1. Осуществите запуск Word.

2. Создайте новый документ и сохраните его под именем My_text.

3. В свой документ скопируйте текст из справочной информации о выделении текста в документах. Для этого:

— вызовите *Справку* и в закладке *Предметный указатель* по ключевому слову *Выделение* найдите и выберите главу *Выделение текста и рисунков;*

— в появившемся диалоговом окне выберите: *Выделение текста и рисунков с помощью мыши;*

— выделите весь предложенный текст с помощью команды меню *Правка → Выделить все;*

— скопируйте выделенный текст в буфер обмена, нажав комбинацию клавиш <*Ctrl*> + <*Insert*>;

— выйдя из *Справки*, вставьте этот текст в свой документ, нажав комбинацию клавиш <*Shift*> + <*Insert*>.

4. Установите единицы измерения — сантиметры. Для этого выполните команду *Сервис → Параметры → Общие → Сантиметры.*

5. Используя клавишу <*Enter*> для разбиения строки на две, клавишу <*Delete*> — для удаления лишнего текста, а также панель инструментов *Форматирование,* метки на линейке и способы выделения фрагментов текста, отредактируйте вставленный текст следующим образом:

— заголовок — полужирным шрифтом;

— заголовки абзацев — подчеркнутым курсивом;

— для первого абзаца: отступ слева — 1 см, шрифт — Times New Roman;

— для второго абзаца: отступ слева — 2 см, шрифт — Arial;

— для третьего абзаца: отступ слева — 3 см, шрифт — Fixedsys;

— для четвертого абзаца: выравнивание — *По центру* страницы, шрифт — Courier;

— для пятого абзаца: выравнивание — *По правому краю* страницы, отступ справа — 1 см; шрифт выберите самостоятельно.

6. Сохраните ваш документ в своей папке, указав при этом имя файла (например: Лаб_раб_1 или другое). Для этого выполните команду *Файл → Сохранить как...*

7. Заключите последний абзац вашего документа в рамку.

8. Наберите следующее выражение: $F(x, y^{(k)}, y^{(k+1)}, ..., y^{(n)}) = 0$; чтобы поставить символы в верхний индекс, выполните команду *Формат → Шрифт → Верхний индекс;* установите интервал между символами 2 пт, выполнив команду *Формат → Шрифт →* закладка *Интервал →* 2 пт.

Работа № 2. Автоматизация работы с текстом

Цель работы: научиться открывать и сохранять созданные ранее документы; редактировать набранный текст, выполняя при этом операции копирования и перемещения, используя элементы поиска и замены слов, *Автотекста, Автозамены, Тезауруса* и проверку орфографии.

Задание:

1. На базе документа, созданного в практической работе № 1, выполните действия по удалению, перемещению и копированию фрагментов текста различными способами.

2. Проверьте орфографию.

3. Создайте элементы *Автозамены* и *Автотекста* при вводе. С помощью поиска и замены найдите определенное слово или набор букв и замените его. Найдите значение и синонимы слова *Прежде.*

4. С помощью режима вставки символов наберите следующее выражение: $\Sigma(\alpha \pm \beta) \cdot \varphi/\eta$.

Методика выполнения работы:

1. Откройте документ, созданный в предыдущей практической работе (команда меню *Файл → Открыть*).

2. Скопируйте последний абзац (вместе с рамкой) и вставьте его между заголовком и первым абзацем, используя мышь. Для этого:

— выделите абзац;

— правой кнопкой мыши с помощью приема *Перетаскивание* поместите указатель мыши под заголовок;

— в контекстном меню выполните команду *Копировать*.

3. Скопируйте второй абзац в конец всего текста, используя только клавиши клавиатуры. Для этого:

— выделите абзац;

— скопируйте его в буфер обмена (нажав комбинацию клавиш <*Ctrl*> + <*C*> или <*Ctrl*> + <*Insert*>);

— поставьте курсор в конец текста;

— выполните команду<*Ctrl*> + <*V*> или <*Shift*> + <*Insert*>.

4. Удалите предпоследний абзац (вместе с рамкой) любым способом (с помощью клавиатуры, основного или контекстного меню):

— выделите абзац;

— щелкните на кнопке *Вырезать* на панели инструментов *Стандартная*.

5. Переместите на место удаленного абзаца его копию (вместе с рамкой) из начала текста с помощью мыши любым способом (перетаскиванием левой или правой кнопкой), текст предварительно выделите.

6. Проделайте ряд самостоятельных упражнений по копированию и перестановке слов в предложениях и букв — в словах.

7. Проверьте орфографию с помощью панели инструментов *Стандартная* и с помощью команды меню *Сервис* → *Правописание*. Не забудьте перед этим установить курсор в начало текста (если проверка осуществляется от курсора вниз).

8. Создайте элемент *Автозамены* и используйте его при наборе и редактировании текста. Предположим, что в тексте постоянно требуется набирать выражение «к левому краю». Чтобы создать элемент *Автозамены,* наберите это выражение и выделите его.

9. Выполните команду *Сервис* → *Автозамена* → вкладка *Автотекст*.

10. Щелкните на кнопке *Добавить*.

11. Проверьте, как работает режим *Автозамены*. Для этого в свободном месте документа начинайте набирать первые буквы вашего А*втотекста* до его появления на экране.

12. Создайте элемент *Автотекста,* в котором будет находиться один из абзацев, и на свободном месте вашего документа вставьте его. Для этого:

— выделите абзац в тексте;

— выполните команду *Вставка → Автотекст → Создать;*

— в строке *Имя элемента* диалогового окна введите условное название для выделенного абзаца и щелкните на кнопке *Добавить;*

— вставьте абзац, используя режим *Автотекста*. Для этого выполните команду *Вставка → Автотекст → Автотекст →* вкладка *Автотекст;*

— в списке *Имя элемента* выберите ваш элемент и щелкните на кнопке *Вставить.*

13. Ознакомьтесь с режимами поиска и замены слов (символов). В тексте найдите определенное слово или набор букв и замените его (команда меню *Правка → Найти*).

14. Найдите значение и синонимы слова *Прежде* с помощью команды *Сервис → Язык → Тезаурус.*

15. С помощью режима вставки символов (*Вставка → Символ*) наберите следующее выражение: $\Sigma(\alpha \pm \beta) \cdot \varphi/\eta$.

16. Сохраните ваш файл в своей папке под новым именем, например: Лаб_раб_2.

Работа № 3. Элементы издательской работы

Цель работы: добавление колонтитулов и рисунков в документ. Создание многоколонного текста и стилей.

Задание:

1. На базе документа, созданного в практической работе № 2, установите: верхнее поле страницы — 3 см, расстояние от края до верхнего колонтитула — 1 см.

2. Создайте на всех четных страницах колонтитул.

3. Измените форматирование текста таким образом, чтобы представить его в виде одного абзаца.

4. Представьте этот текст в виде трех колонок с разделителями. Расстояние между колонками — 0,6 см.

5. Создайте и примените к заголовку многоколонного текста свой стиль шрифта.

6. Создайте и примените к многоколонному тексту свой стиль абзаца.

7. Создайте небольшой рисунок в графическом редакторе Microsoft Paint и вставьте его в свой текстовый документ.

Методика выполнения работы:

1. Откройте документ, созданный в практической работе № 2, и измените следующие параметры страницы для всего документа: верхнее поле — 3 см, от края до верхнего колонтитула — 1 см (меню *Файл* → *Параметры страницы* → *Поля*);

2. Сохраните этот документ под новым именем, например: Лаб_раб_3.

3. Создайте в нем на всех четных страницах колонтитул. Для этого:

— в *Параметрах страницы* установите: *Различать колонтитулы четных и нечетных страниц;*

— установите курсор на четную страницу и выполните команду *Вид* → *Колонтитул*. В этот колонтитул вставьте из файла рисунок, а также с помощью *Автотекста* занесите номер страницы, имя вашего документа, дату его создания и впишите свою фамилию, имя и отчество.

Внесенную информацию отформатируйте следующим образом:

— нумерацию страниц — *По центру;*

— имя документа, дату и фамилию — *По правому краю;*

— на всю информацию установите начертание шрифта и цвета.

4. Измените формат текста, который был скопирован вами из справочной информации в предыдущей практической работе, следующим образом:

— уберите все рамки;

— установите: стиль абзаца — *Обычный,* шрифт — Times New Roman, размер шрифта — 12, начертание — *Обычное*. Выровняйте текст *По левому краю* страницы;

— представьте данный текст, кроме заголовка и последнего предложения, как один абзац. Для этого удалите все символы конца абзаца («¶»).

5. Представьте текст в виде трех колонок равной ширины с разделителем, расстояние между колонками — 0,6 см. Для этого:

— выделите текст;

— выполните команду меню *Формат* → *Колонки;*

— установите нужное количество колонок, а также поставьте галочки в окна *Разделитель* и *Колонки одинаковой ширины;*

— установите расстояние между колонками: 0,6 см.

6. Создайте и примените к заголовку многоколонного текста свой стиль шрифта. Стиль можно выбрать и установить на выделенный текст по его названию в *Списке стилей* на панели инструментов. Для этого:

— выполните команду меню *Формат* → *Стиль* → *Создать;*

— в появившемся окне введите название, например вашу фамилию, и установите стиль символа;

— выполните команду *Формат* → *Шрифт*. В появившемся окне введите интервал между символами (отличный от обычного), цвет, цвет и узор фона, а также размер и начертание. Эти параметры выберите самостоятельно.

7. Создайте и примените к многоколонному тексту свой стиль абзаца. Стиль должен иметь название, например ваше имя, интервал между строками (отличный от обычного), абзацный отступ, выравнивание, шрифт. Эти параметры выберите самостоятельно. Для создания стиля абзаца нужно:

— выполните команду меню *Формат* → *Стиль* → *Создать;*

— в появившемся окне с клавиатуры введите название, например ваше имя, и установите *Стиль* абзаца;

— щелкнув на кнопке *Формат* и выбрав *Шрифт,* введите необходимые установки для шрифта. Щелкнув на кнопке *Формат* и выбрав *Абзац,* введите необходимые установки для абзаца, т. е. интервал между строками (отличный от обычного), абзацный отступ, выравнивание и т. д.

В дальнейшем созданный вами стиль вы можете выбрать (по его названию) в *Списке стилей* на панели инструментов и установить на любой выделенный абзац.

8. Создайте небольшой рисунок в графическом редакторе Microsoft Paint и вставьте его в свой текстовый документ. Для этого можно воспользоваться двумя способами:

— или в графическом редакторе после создания рисунка скопировать выделенную область рисунка в буфер обмена и вставить в свой документ;

— или сохранить созданный рисунок на диске и вставить в свой документ из файла.

9. Сохраните этот документ.

Работа № 4. Применение *Редактора формул* и создание графических объектов

Цель работы: изучить основные возможности *Редактора формул.* Освоить создание графических объектов в Word.

Задание 1: используя возможности *Редактора формул,* наберите следующее выражение:

$$\begin{bmatrix} x^1 \\ x^2 \\ \vdots \\ x^i \\ \vdots \\ x^n \end{bmatrix} = \frac{1}{D} \begin{bmatrix} A_1^1 & A_1^2 & \dots & \dots & A_1^n \\ A_2^1 & A_2^2 & \dots & \dots & A_2^n \\ \dots & \dots & \dots & \dots & \dots \\ A_i^1 & A_i^2 & \dots & \dots & A_i^n \\ \dots & \dots & \dots & \dots & \dots \\ A_n^1 & A_n^2 & \dots & \dots & A_n^n \end{bmatrix} \cdot \begin{bmatrix} b^1 \\ b^2 \\ \vdots \\ b^i \\ \vdots \\ b^n \end{bmatrix}.$$

Методика выполнения задания:

1. Создайте новый документ и сохраните его в свою рабочую папку.

2. Зайдите в *Редактор формул,* выполнив команду *Вставка* → *Объект* → вкладка *Создание.*

3. Создайте вектор. Для этого в списке *Тип объекта* выберите: Microsoft Equation 3.0, затем:

— в появившемся окне редактирования формул выберите *шаблон 1* из нижнего ряда панели инструментов (рис. 4.19, *а*) и установите в него курсор;

— в появившемся окне введите требуемые размеры матрицы: число строк — 6, число столбцов — 1;

— установите курсор в первое поле и введите символ: *x;*

— выберите *шаблон 1,* указанный на рис. 4.19, *б*, переместите в него курсор и введите символ: 1;

— заполните таким же образом второе, четвертое, шестое поля;

— в третьем и пятом полях выберите символ из верхнего ряда панели инструментов.

4. Переместите курсор за закрывающуюся скобку, с клавиатуры введите знак равенства («=»), установите шаблон — *дробь,* с клавиатуры введите символы: 1 и D.

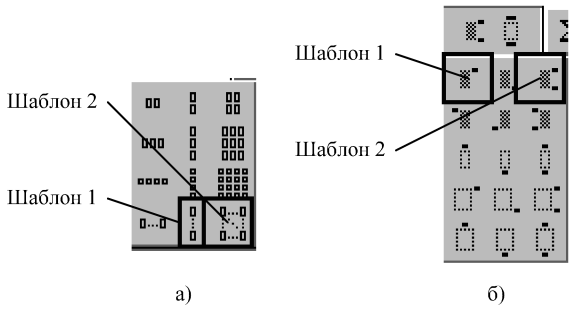


Рис. 4.19. Панели инструментов *Редактора формул*

5. Чтобы создать матрицу:

— в списке *Тип объекта* выберите: *Microsoft Equation 3.0;*

— в появившемся окне редактирования формул выберите: *шаблон 2* (рис. 4.19, *а*); в появившемся окне введите требуемые размеры матрицы: число строк — 6, число столбцов — 5;

— установите курсор в первое поле и введите символ: A;

— для набора верхних и нижних индексов используйте *шаблон 2* (рис. 4.19, *б*).

6. Таким же образом заполните остальные поля матрицы, но, чтобы облегчить работу, воспользуйтесь возможностью копирования. Для этого:

— выделите нужную область и скопируйте ее в буфер обмена, нажав комбинацию клавиш <*Ctrl*> + <*Insert*>;

— поставьте курсор во второе поле и вставьте скопированную область из буфера обмена, нажав комбинацию клавиш <*Shift*> + <*Insert*>. Далее нужно просто изменить значения.

Оставшуюся часть формулы заполните сами.

Задание 2:

1. Создайте рисунок по образцу, представленному на рис. 4.20, с надписями и заголовком.

2. Все элементы рисунка сгруппируйте в единое целое.

Методика выполнения задания:

1. На свободном месте документа нарисуйте оси. Для этого на панели инструментов *Рисование* щелкните на кнопке *Линии,* перенесите курсор мыши в область документа и проведите линию.

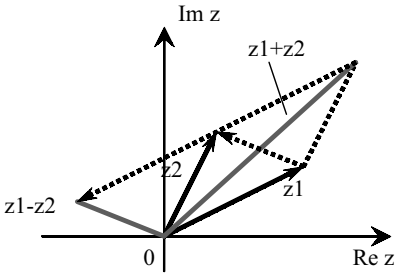


Рис. 4.20. Создание рисунка по образцу

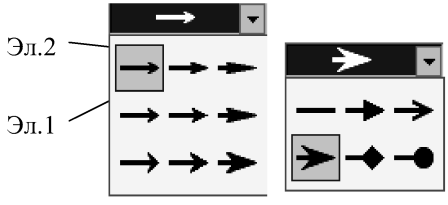


Рис. 4.21. Выбор типа линий

2. Щелкните на вашей линии правой кнопкой мыши и в контекстном меню выберите пункт *Формат автофигуры*.

3. В появившемся диалоговом окне выберите закладку *Цвета и Линии* и установите:

— в поле *Толщина* — 0,75 пт;

— в полях *Конец стрелки* и *Размер стрелки* — элементы рис. 4.21.

4. Нарисуйте линию на месте вектора $z1$ (угол наклона возьмите произвольный); после этого установите: толщину линии — 2 пт и вогнутую стрелку.

5. Нарисуйте линию на месте вектора $z2$ и преобразуйте ее так же, как предыдущую.

6. Выделите этот вектор, скопируйте его в буфер обмена и тут же вставьте. Появившуюся копию переместите, соединив ее конец с началом вектора $z1$. Измените шаблон этой линии на квадратные точки, убрав стрелку.

7. Теперь постройте результирующую линию сложения комплексных чисел ($z1 + z2$), после этого установите: толщину линии — 2,5 пт, цвет — сизый.

8. Соедините оба конца векторов $z1$ и $z2$ линией, как показано в образце, и затем измените шаблон этой линии на квадратные точки. Скопируйте получившуюся линию и переместите копию на место результирующей линии вычитания комплексных чисел ($z1 - z2$), затем установите: толщину линии — 2,5 пт, цвет — сизый.

9. Нарисуйте линию, соединив оба конца результирующих линий, как показано на рис. 4.20, изменив ее шаблон, установив толщину 2 пт и стрелку.

10. Проверьте получившийся рисунок и сгруппируйте.

11. После установки надписей снимите с них обрамление, фон должен быть прозрачным. При занесении текста используйте нижний индекс (желательно установить соответствующую кнопку на панель инструментов).

12. Сгруппируйте все элементы вашего рисунка вместе с надписями и названием; для этого, удерживая нажатой клавишу *<Shift>,* выделите все элементы рисунка и выполните команду *Действие → Группировать.*

Работа № 5. Создание таблиц и списков

Цель работы: изучить создание и форматирование таблиц в текстовых документах, а также создание и редактирование маркированных, нумерованных и многоуровневых списков.

Задание:

1. В новом документе создайте таблицу, установив заданные размеры, фоновые узоры, цвета, обрамления снаружи и внутри таблицы.

2. Произведите заполнение таблицы, оптимально подобрав размер и тип шрифта, чтобы не нарушить установленные размеры таблицы.

3. Произведите выравнивание текста внутри ячеек таблицы: *По центру, По горизонтали* и *По вертикали.*

4. Создайте копию таблицы ниже на этой же странице.

5. Преобразуйте скопированную таблицу в текст.

6. Создайте многоуровневые списки. Вид списков должен соответствовать образцу, представленному на рис. 4.22.

Производительные силы общества
1) Личный фактор производства
    а) Рабочая сила
        (а) Труд людей
2) Вещественный фактор производства
    а) Средства производства
        (а) Средства труда
        (б) Предметы труда

Производительные силы общества
❖ Личный фактор производства
    ➢ Рабочая сила
        ♦ Труд людей
❖ Вещественный фактор производства
    ➢ Средства производства
        ♦ Средства труда
        ♦ Предметы труда

Рис. 4.22. Образец многоуровневых списков

Методика выполнения работы:

1. Откройте новый документ и установите следующие параметры страницы:

— отступ слева — 1,2 см;

— отступ справа — 0,5 см.

2. Создайте таблицу (ее вид должен соответствовать образцу, представленному на рис. 22), состоящую из 7 строк и 10 столбцов. Для этого выполните команду *Таблица → Вставить таблицу*.

3. Произведите объединение ячеек первой строки и внесите в нее информацию согласно образцу. Для этого выделите нужные ячейки и выполните команду *Таблица → Объединить ячейки*.

4. Измените следующие параметры таблицы:

— высота первой строки — 1,19 см, остальные — минимум;

— ширина первого столбца — 0,94 см, второго — 3,25 см, остальных — 1,75 см.

5. Введите с клавиатуры соответствующий текст в ячейки таблицы, следуя образцу; установите: шрифт — Times New Roman, размер для первой строки — 12 пт, для второй — 11 пт, для остальных — 10 пт.

6. Затените ячейки таблицы, выбрав соответствующий узор и цвет фона, как указано в образце. Для этого выделите нужные ячейки и выполните команду *Формат → Границы и заливка → Заливка*.

7. Оформите линии сетки с помощью команды *Формат → Границы и заливка → Границы*.

8. Произведите выравнивание информации внутри ячеек таблицы: *По центру, По горизонтали* и *По вертикали*.

9. Создайте копию вашей таблицы ниже на этой же странице. Для этого выделите таблицу и скопируйте ее в буфер обмена, а затем установите курсор в нужное место и вставьте таблицу.

10. Преобразуйте скопированную таблицу в текст с помощью команды *Таблица → Преобразовать → Преобразовать в текст,* и наоборот: *Таблица → Преобразовать → Преобразовать в таблицу*.

11. Добавьте в документ многоуровневые списки. Вид списков должен соответствовать образцу. Для этого выполните команду *Формат → Список → Многоуровневый*. Выбрав нужный

вариант списка, щелкните на кнопке *Изменить*. Далее настройте нужные уровни списка.

12. Напечатайте первый элемент списка (рис. 4.22) и нажмите клавишу <*Enter*>.

13. Перейдите на второй уровень, нажав клавишу <*Tab*>. После появления цифры формата второго уровня введите следующий элемент. Заполнение третьего и четвертого уровней произведите самостоятельно.

14. Чтобы перейти с четвертого уровня на третий и на второй, нажмите комбинацию клавиш <*Shift*> + <*Tab*>.

15. Заполните список до конца и по аналогии с предыдущим примером создайте второй список. Для этого вначале создайте копию первого списка и на его основе измените форматирование на многоуровневое маркированное.

Работа № 6. Вычисления в Word. Построение диаграмм

Цель работы: вычисления в таблицах. Построение диаграмм в текстовых документах.

Задание:

1. Откройте новый документ и создайте таблицу по образцу, представленному на рис. 4.23.

2. По данным таблицы постройте четыре диаграммы в соответствии с образцом, представленным на рис. 4.24.

Производство продукции городского молокозавода

	Творог	*Сметана*	*Кефир*
1991	50	260	322
1993	105	266	370
1995	120	250	339
1998	115	400	296

Рис. 4.23. Образец таблицы для вычисления в Word

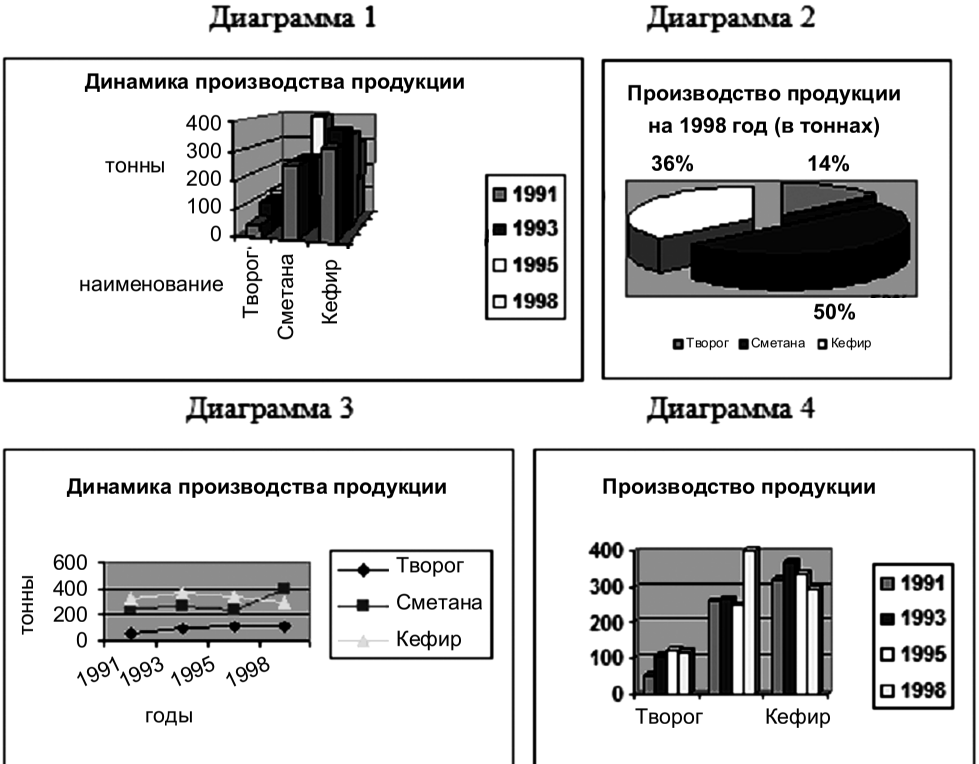


Рис. 4.24. Типы диаграмм

3. Справа от таблицы добавьте пустой столбец с заголовком «Итого за год», в котором с помощью формул подсчитайте, сколько всего продукции было произведено за каждый указанный в таблице год.

4. Внизу таблицы добавьте пустую строку с заголовком «Итого», в которой подсчитайте, сколько тонн каждого продукта было произведено за все указанные в таблице годы.

Методика выполнения работы:

1. Чтобы вставить в документ таблицу, выполните команду *Таблица → Добавить*. При запросе о ее размерах введите требуемое количество строк и столбцов.

2. Вид таблицы достигается установкой на нее *Автоформата*. Для этого выделите таблицу и выполните команду *Таблица → Автоформат*.

3. Не снимая выделения таблицы, произведите ее обрамление снаружи и внутри.

4. Заполните таблицу данными.

5. При выполнении работы установите предлагаемый формат: шрифт на заголовок — полужирный курсив, а на таблицу — курсив; выравнивание — *По центру, По горизонтали* и *По вертикали*.

6. Чтобы создать диаграмму, выделите таблицу и выполните команду *Вставка → Объект → Диаграмма Microsoft Graph*. На экране должны появиться таблица данных и диаграмма. Если в этой таблице данные будут не совпадать с теми, которые записаны в вашей таблице, то просто удалите их и скопируйте свои; можно скопировать сразу всю таблицу.

7. Щелкая левой кнопкой мыши по диаграмме, выделяя каждый ее элемент (стены, ряды, оси значений, оси категорий, линии сетки, легенды, элементы внутри легенды и т. д.) и вызывая щелчком правой кнопкой мыши контекстное меню выделенных элементов, настройте их отдельно согласно образцам (см. 4.8.6. «Обработка числовой информации в таблицах» и 4.8.7. «Построение диаграмм»).

8. Справа в таблице добавьте столбец с заголовком «Итого за год», в котором с помощью формул подсчитайте, сколько всего продукции было произведено за каждый указанный в таблице год (см. в 4.8.5: «Добавление и удаление строк и столбцов

в таблице», а также в 4.8.6: «Обработка числовой информации в таблицах» и в 4.8.7: «Построение диаграмм»).

9. Внизу таблицы добавьте пустую строку с заголовком «Итого», в которой подсчитайте, сколько тонн каждого продукта было произведено за все указанные в таблице годы.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ТЕМЕ 4.8

1. **Что нужно сделать, чтобы быстро выделить с помощью мыши слово, строку, несколько строк, предложение, абзац, весь документ?**
2. **Как установить интервал между символами в тексте, например равный 1,2 пт?**
3. **Какие вы знаете способы копирования фрагментов текста и рисунков?**
4. **Чем отличается перетаскивание объекта левой кнопкой мыши от перетаскивания правой?**
5. **Как установить или убрать обрамление текста, обрамление с определенных сторон, а также создать свой стиль рамки?**
6. **Что нужно сделать, чтобы установить рамку на страницу, соблюдая стандартные параметры: по 0,5 см — до верхнего, нижнего и правого края, 2 см — от рамки до левого края?**
7. **Что нужно сделать, чтобы изменить цвет и узор выделения текста?**
8. **Можно ли установить пароль на открытие файла и его редактирование?**
9. **Как можно выделить и скопировать текст, используя клавиши клавиатуры (не заходя в меню)?**
10. **Где и как можно применить эффекты шрифта — нижний индекс, скрытый текст?**
11. **Как отменить автоматическую проверку орфографии и грамматики?**
12. **Может ли режим поиска и замены слов заменять и удалять буквы в словах? Различается ли при этом регистр? Что для этого нужно сделать?**
13. **Как, используя режим поиска и замены, найти слова (символы), напечатанные курсивом, размером 12 пт, определенным цветом, и изменить их начертание на обычное, полужирное, подчеркнутое, размер — 16 пт, цвет — синий?**
14. **Что нужно сделать, чтобы найти антоним указанного слова?**
15. **Можно ли присвоить символу комбинацию клавиш и как это сделать?**

16. **Какими способами можно установить нумерацию страниц? В каком месте страницы?**
17. **Что нужно сделать, чтобы установить колонтитул только на первой странице?**
18. **Может ли колонтитул размещаться в центре страницы?**
19. **Как создать нижний колонтитул и как его убрать?**
20. **Какую информацию можно занести в колонтитул? Можно ли занести в него таблицу?**
21. **Какими способами можно разделить текст на колонки и сколько колонок можно создать в тексте?**
22. **Как можно изменить ширину колонок и установить между ними разделители?**
23. **После создания рисунка в графическом редакторе, например в Microsoft Paint, какими способами можно вставить его в свой документ?**
24. **Чем отличается *Стиль абзаца* от *Стиля шрифта?* Как создать свой стиль?**
25. **Как можно отключить в редакторе формул курсивное начертание символов в стиле *Математический?***
26. **Что нужно сделать, чтобы изменить в формуле шрифт с установленного по умолчанию Times New Roman на какой-нибудь другой и увеличить размеры символов и индексов?**
27. **Для чего в редакторе формул предназначен стиль *Текст* и какие еще стили существуют в редакторе формул?**
28. **Какими способами можно установить пробел в редакторе формул?**
29. **Что необходимо сделать, чтобы изменить формат линии при рисовании, например установить стрелку, и как изменить ее тип и размер?**
30. **Как сгруппировать элементы рисунка в единое целое и повернуть изображение?**
31. **В каких случаях и для чего применяется сетка в таблице и при рисовании? Можно ли показать ее на экране?**
32. **Что нужно сделать, чтобы добавить в документ таблицу? Какого рода информацию можно в нее занести?**
33. **Как добавить в конец таблицы дополнительный столбец или строку?**
34. **Как изменить ширину у нескольких столбцов и высоту — у нескольких строк одновременно?**
35. **Как перенести или скопировать информацию из одной ячейки в другую?**
36. **Что нужно сделать, чтобы произвести выравнивание текста внутри ячеек таблицы по вертикали и по горизонтали?**

37. **Какими способами можно установить многоуровневый *Список?***
38. **Как изменить цвет маркера или номера элемента *Списка?* Что нужно сделать, чтобы добавить маркер из таблицы символов?**
39. **Что происходит при преобразовании таблицы в текст и обратно?**
40. **Что нужно сделать, чтобы изменить ориентацию текста в таблице?**
41. **Как изменить расстояние между столбцами в таблице?**
42. **Как изменить местоположение легенды у диаграммы и убрать у нее обрамление?**
43. **Можно ли прямо в диаграмме изменить значение какого-нибудь параметра?**
44. **Как изменить цвет и узор для любого ряда данных в диаграмме? Как установить подписи данных?**
45. **Какой по умолчанию устанавливается фон области диаграммы и как его изменить?**
46. **Если в таблице показаны формулы, как просмотреть их значения? Как просмотреть формулы, если показаны значения?**
47. **Какую ссылку в формулах таблицы нужно написать, если необходимо выделить всю строку или столбец?**
48. **Как в формуле обратиться к ячейкам другой таблицы?**
49. **Будет ли автоматически пересчитываться результат при изменении исходных данных в таблице?**

Тема 4.9
ТАБЛИЧНЫЙ ПРОЦЕССОР EXCEL

Табличными процессорами называются прикладные программы, предназначенные для работы с электронными таблицами. В настоящее время известно много таких программ: Excel, Lotus 1-2-3, QuattroPro, SuperCalc, Multiplan, Суперплан, АБАК и т. д.

Электронная таблица представляет собой прямоугольную матрицу, состоящую из ячеек, каждая из которых имеет свой номер. Рассмотрим структуру электронных таблиц на примере работы с электронной таблицей **Excel.**

Программа Microsoft Excel предназначена для работы с таблицами данных, преимущественно числовыми. При формировании таблицы выполняются ввод, редактирование и форматирование текстовых, числовых данных, а также формул (см. рис. 4.2).

Документ Excel называется **рабочей книгой**. Рабочая книга представляет собой набор *рабочих листов,* каждый из которых имеет табличную структуру и может содержать одну или несколько таблиц.

В окне документа в программе Excel отображается только текущий рабочий лист, с которым и ведется работа. Каждый рабочий лист имеет название, оно отображается на ярлычке листа, расположенном в нижней части листа. С помощью ярлычков можно переключаться к другим рабочим листам, входящим в ту же рабочую книгу.

Чтобы переименовать рабочий лист, надо щелкнуть правой кнопкой мыши на его ярлычке и выполнить команду *Переименовать.*

Чтобы выделить несколько рабочих листов подряд, необходимо выделить первый рабочий лист и, удерживая нажатой клавишу <*Shift*>, выделить последний лист.

Чтобы выделить несколько несмежных рабочих листов, надо выделить первый рабочий лист и, удерживая нажатой клавишу <*Ctrl*>, выделять каждый требуемый лист.

4.9.1. Ячейки и их адресация

Электронные таблицы состоят из столбцов и строк. Столбцы озаглавлены прописными буквами латинского алфавита и их двухбуквенными комбинациями (A, B, C, ..., AA, ..., IV). Строки озаглавлены цифрами (1, 2, 3, ...). Всего рабочий лист может содержать до 256 столбцов и до 65 536 строк.

Место пересечения столбца и строки называется **ячейкой**. Каждая ячейка имеет свой уникальный *адрес,* состоящий из имени столбца и номера строки, например: A28, P45 и т. п.

Формат указания адреса ячейки называется **ссылкой.** Одна из ячеек всегда является активной и выделяется рамкой активной ячейки. Эта рамка в программе Excel играет роль курсора. Операции ввода и редактирования всегда проводятся в активной ячейке. Адрес и содержимое текущей ячейки выводится в строке ввода электронной таблицы. Переместить рамку активной ячейки можно клавишами управления курсором или мышью. Данные в ячейке могут быть основными, т. е. не зави-

сящими от других значений ячеек в таблице, и производными, т. е. определяемыми по значениям других ячеек с помощью вычислений.

Диапазон (блок) ячеек. В электронных таблицах работают как с отдельными ячейками, так и с группой ячеек, которые образуют блок. В качестве блока могут рассматриваться строка или часть строки, столбец или часть столбца, а также прямоугольник, состоящий их нескольких строк, столбцов или их частей. Адрес блока задается указанием ссылок первой и последней его ячеек, между которыми ставится разделительный символ, например «:» (двоеточие) или «..» (две точки). Каждая команда табличного процессора требует указания блока ячеек, в отношении которого она будет выполнена.

Блок используемых ячеек можно выделить двумя путями: непосредственно набором с клавиатуры начального и конечного адресов ячеек, формирующих диапазон, либо выделением соответствующей части таблицы с помощью клавиш управления курсором.

4.9.2. Вычисления в Excel

Вычисления в электронных таблицах Excel осуществляются с помощью формул. Формула может содержать числовые константы, ссылки на ячейки и функции в Excel, соединенные знаками математических операций. Ссылки на ячейки можно задать разными способами. По умолчанию ссылки на ячейку в формулах рассматриваются как относительные.

Относительные ссылки — это ссылки, которые при копировании формулы изменяются автоматически в соответствии с относительным расположением исходной ячейки и создаваемой копией (например: H4).

Абсолютные ссылки — это ссылки, которые при копировании не изменяются (например: H4).

Смешанные ссылки — это ссылки, которые сочетают в себе и относительную и абсолютную адресацию (например: $H4, H$4).

Для изменения способа адресации при редактировании формулы надо выделить ссылку на ячейку и нажать клавишу (например: <*F4*>).

Функции в Excel предназначены для вычисления базовых величин, необходимых при проведении сложных финансовых, статистических, математических и других расчетов. Методика использования функций требует соблюдения определенной технологии:

1. На рабочем листе в отдельных ячейках подготавливаются значения основных аргументов функции.

2. Осуществляется вызов *Мастера функций* с помощью команды *Вставка* → *Функция* или щелчком на кнопке f_x на панели инструментов *Стандартная*.

3. Выполняется выбор *категории функции*. В списке *Функция* содержится полный перечень доступных функций выбранной категории. В нижней части окна появляются краткий синтаксис и справка о назначении выбранной функции. Гиперссылка *Справка по этой функции* вызывает экран справки для встроенной функции, на которой установлен курсор. Кнопка *Отмена* прекращает работу *Мастера функций*. При щелчке на кнопке *OK* осуществляется переход к работе с диалоговым окном выбранной функции. Кнопка *OK* переносит в строку формулы синтаксическую конструкцию выбранной встроенной функции.

4. В списке требуемой функции выполняется выбор, в результате которого появляется диалоговое окно для ввода аргументов. Для каждой функции существует регламентированный по составу и формату значений перечень аргументов.

5. В поле ввода диалогового окна можно вводить как ссылки на адреса ячеек, содержащих собственно значения аргументов, так и сами значения аргументов.

6. Если аргумент является результатом расчета другой встроенной функции Excel, можно организовать вычисление *вложенной* встроенной функции, вызвав *Мастера функций* щелчком на одноименной кнопке, расположенной перед полем ввода аргументов.

7. Отказ от работы со встроенной функцией осуществляется щелчком на кнопке *Отмена*.

8. Завершение ввода аргументов и запуск расчета значения встроенной функции выполняется щелчком на кнопке *OK*.

9. Формула начинается со знака «=» (знака равенства). Далее следует имя функции, а в круглых скобках указываются ар-

гументы в последовательности, соответствующей синтаксису функции. В качестве разделителей аргументов используется выбранный при настройке Windows разделитель; обычно это точка с запятой («;») или запятая («,»). Например, в ячейку C13 введена формула: =ДОХОД(B16;B17;0.08;47.727;100;2;0).

Отдельные аргументы функции могут быть как константами, так и ссылками на адреса ячеек.

Подбор параметров Excel. Вычислительные возможности Excel позволяют решать как прямые, так и обратные задачи, выполнять исследование области допустимых значений аргументов, подбирать значения аргументов под заданное значение функции.

Для подбора параметров выполняется команда *Сервис* → *Подбор параметра*. В диалоговом окне задается требуемое значение функции: в поле *Изменение значения ячейки* указывается адрес ячейки, содержащей значение одного из аргументов функции. Excel решает и обратную задачу: подбор значения аргумента для заданного значения функции. В случае успешного завершения подбора выводится окно, в котором указан результат: *Текущее значение* функции для подобранного значения аргумента, новое значение аргумента функции содержится в соответствующей ячейке.

При щелчке на кнопке *OK* подобранное значение аргумента сохраняется в ячейке аргумента, при щелчке на кнопке *Отмена* происходит восстановление значения аргумента. При неуспешном завершении подбора параметра выдается соответствующее сообщение о невозможности подбора аргументов.

4.9.3. *Диспетчер сценариев* в Excel

Для вариантных финансовых расчетов, основанных на задании различных значений аргументов функции, целесообразно воспользоваться сценарным подходом, реализованным средствами Excel.

Диспетчер сценариев применяется для создания списка значений для подстановки в изменяемые ячейки листа. Каждый *сценарий* — это набор предположений, который можно использовать для прогнозирования результатов пересчета листа. Ис-

пользуя *Диспетчер сценариев,* можно создавать несколько сценариев, в каждом из которых содержится до 32 значений подстановки в ячейки листа; присваивать имена; сохранять и выполнять сценарии листа; создавать итоговые отчеты по сценариям; объединять и скрывать сценарии; защищать их от изменений; автоматически отслеживать изменения сценария.

Сценарий — это именованная совокупность значений изменяемых ячеек. Для ячеек, являющихся аргументами функций, можно задавать различные значения. Команда *Сервис → Сценарий* вызывает диалоговое окно *Диспетчер сценариев* для ячеек текущего рабочего листа.

В окне *Сценарии* представлен список сценариев текущего рабочего листа. Возможно объединение сценариев, находящихся в открытых книгах или на других листах текущей рабочей книги, при щелчке на кнопке *Объединить.* Для создания нового сценария следует щелкнуть на кнопке *Добавить,* при этом появляется новое диалоговое окно.

В поле *Название сценария* вводится имя нового сценария: последовательность символов (максимальная длина имени не более 255 знаков).

В окне *Примечание* можно записать текст, поясняющий сценарий. По умолчанию сюда заносятся имя пользователя и дата создания сценария. Имя пользователя можно изменить, выполнив команду *Сервис → Параметры →* вкладка *Общие →* поле *Имя пользователя.*

С помощью переключателя *Запретить изменения* реализуется защита значений изменяемых ячеек от редактирования. Переключатель *Скрыть* позволяет не показывать имя сценария в списке. При щелчке на кнопке *ОК* появляется диалоговое окно для ввода значений изменяемых ячеек.

Для просмотра результатов подстановки значений изменяемых ячеек по определенному сценарию в диалоговом окне *Диспетчера сценариев* следует выбрать из списка имя сценария и щелкнуть на кнопке *Вывести.*

Excel выполняет подстановку значений изменяемых ячеек сценария и производит расчет значения функции. Все изменения будут отражены на рабочем листе в ячейках, содержащих формулы и имеющих ссылки на изменяемые ячейки сценария; новые результатные значения также будут выведены.

Кнопка *Закрыть* обеспечивает выход из окна *Диспетчера сценариев,* при этом в изменяемых ячейках сохраняются значения последнего участвовавшего в просмотре сценария. Для подготовки отчетов по сценариям щелчком на кнопке *Отчет* вызывается диалоговое окно для выбора типа итогового отчета. В поле *Ячейки результата* указываются адреса ячеек, значения которых зависят от изменяемых ячеек сценариев.

Формируется два вида отчетов:

— *итоги сценария* — табличный отчет, содержащий для каждого сценария состав изменяемых ячеек и значения выбранных результатных ячеек;

— *сводная таблица* результатов подстановки значений в изменяемые ячейки и вычислений результатов подстановки.

4.9.4. Информационная технология в Excel

Задачи анализа требуют применения информационной технологии выполнения сортировки, фильтрации и консолидации данных в электронных таблицах; формирования разнообразных итогов и сводок; выполнения расчетов в таблицах для данных, удовлетворяющих заданным условиям.

Списки в Excel. Для решения задач обработки всевозможных прайс-листов компьютерных фирм в Excel необходимо представить электронную таблицу в виде списка, или базы данных.

Списки в Excel — это таблицы, строки которых содержат однородную информацию. В Excel список называется **базой данных (БД)**, при этом строки таблицы — это *записи базы данных,* а столбцы — *поля записей*.

Чтобы превратить таблицу Excel в список, или базу данных, необходимо присвоить столбцам однострочные имена, которые будут использоваться как имена полей записей базы данных. Следует иметь в виду, что строка имен полей может состоять из нескольких строк заголовка, размещенных в одной строке таблицы Excel.

При создании списка на рабочем листе Excel необходимо соблюдать следующие правила:

1. На одном рабочем листе не следует помещать более одного списка, так как некоторые операции, например фильтрация, работают в определенный момент только с одним списком.

2. Следует отделять список от других данных рабочего листа хотя бы одним свободным столбцом или одной свободной строкой. Это поможет Excel автоматически выделять список при выполнении *фильтрации* или при *сортировке* данных.

3. Список может занимать весь рабочий лист: 65 536 строк, 256 столбцов.

4. Имена полей списка должны располагаться в первой строке таблицы; Excel использует эти имена при создании отчетов, поиске и сортировке данных.

5. Форматирование заголовков столбцов для имен полей (включая тип данных, шрифт, формат, выравнивание, рамку и стиль прописных букв) должно отличаться от того форматирования, которое использовалось для данных списка.

6. Чтобы отделить имена полей от данных, следует поместить по нижнему краю ячеек строки рамку с именами столбцов. Нельзя использовать пустые строки или пунктирные линии.

7. Каждый столбец списка должен содержать во всех строках однотипные данные.

8. Не следует вводить дополнительные пробелы в начале ячеек данных, поскольку они влияют на сортировку и поиск.

Сортировка списков в Excel в заданном порядке выполняется с помощью команды *Сортировка* в меню *Данные;* предварительно выделяется весь список с заголовками столбцов, кроме итоговых строк таблицы (если они присутствуют). Включение заголовков столбцов в область выделения облегчает настройку сортировки, так как эти имена можно использовать в качестве ключей сортировки. В противном случае ключами сортировки будут стандартные имена столбцов таблицы Excel.

В диалоговом окне, которое открывается выполнением команды *Сортировка*, можно задать до трех ключей с указанием порядка сортировки. Сортировка выполняется сначала по первому ключу, затем — в строках с совпадающим значением первого ключа — по второму ключу, наконец — в строках с одинаковыми значениями первого и второго ключей — по третьему ключу.

Пользовательский порядок сортировки по возрастанию или по убыванию можно задать по заданному ключу. Чтобы применить пользовательский порядок сортировки, надо щелкнуть на кнопке *Параметры* в диалоговом окне *Сортировка*.

Этот режим позволяет установить порядок сортировки по первому ключу — обычному или определяемому пользователем, задать учет кодировки строчных и прописных букв (учет регистра символов), а также направление сортировки — по возрастанию или убыванию.

Задать пользовательский порядок сортировки можно также во вкладке *Списки* диалогового окна *Параметры*, которое открывается по команде *Параметры* в меню *Сервис*.

Фильтрация списков в Excel. Для выбора в списке части информации с некоторым условием необходимо использовать *фильтр*. Имеются две разновидности этой команды, задаваемые параметрами *Автофильтр* и *Усиленный (Расширенный) фильтр*.

Автофильтр. Для использования *Автофильтра* вначале надо выделить область списка или весь список, причем обязательно с заголовками столбцов. При этом Excel преобразует имена столбцов списка в имена полей записей базы данных. По команде *Данные → Фильтр → Автофильтр* в строке заголовков таблицы появляются кнопки с раскрывающимися списками значений.

Автофильтр предполагает использование критериев поиска типа «сравнение». Существует два типа сравнения:

— по точному, или шаблонному, значению;

— по условию отбора.

Точное значение для сравнения выбирается из раскрывающегося списка для указанного поля. Excel в Microsoft Office 97 при выполнении команды *Данные → Фильтр → Автофильтр* формирует списки значений полей, в которых может находиться до 999 элементов (в отличие от Excel 7.0, когда список содержал до 250 элементов).

При выборе сравнения по условию задается критерий отбора, состоящий из двух частей, связанных между собой логической связкой И либо ИЛИ.

Каждая часть условия включает:

— оператор отношения: =, <>, <=, >=, <, >);

— значение, которое может выбираться из списка или содержать шаблонные символы: «*», «?».

Можно задать условия для отбора нескольких столбцов независимо друг от друга, фильтрация выполняется по всем условиям одновременно. Все записи, не прошедшие через фильтр,

будут скрыты. Отфильтрованные записи можно выделить и скопировать в другое место, очистить содержимое, удалить и т. д.

Форма данных. Excel позволяет работать с отдельными записями списка с помощью простой экранной формы. Выполнять основные операции обработки записей списка (просмотр, поиск или фильтрация по критериям, создание новых и удаление уже существующих записей списка).

При установке курсора в область списка и выполнении команды *Данные → Форма* на экран выводится форма, в составе которой имена полей — названия столбцов списка. Для просмотра записей используются полоса прокрутки или кнопки *Далее, Назад;* выводится индикатор номера записи. При просмотре записей возможно их редактирование. Поля, не содержащие формул, доступны для редактирования, а вычисляемые или защищенные поля не редактируются.

Структурирование и группировка данных для формирования итогов в Excel. После того как список отсортирован, можно применить команду *Итоги* из меню *Данные* для создания промежуточных и общих итогов в списке. По этой команде открывается диалоговое окно *Промежуточные итоги*. В нем задаются поле, при каждом изменении значения которого будут вычисляться итоговые значения, и операция, которая будет применяться к значениям полей, отмеченных в списке *Добавить итоги по*.

Создание промежуточных итогов основано на предварительной сортировке записей списка, при этом важен порядок сортировки: состав и подчиненность ключей сортировки. Если сортировка была выполнена по полям (поле1, поле2, поле3), т. е. поле1 является самым старшим в сортировке, поле2 определяет порядок сортировки строк списка при одинаковых значениях поля1, а поле3 задает порядок сортировки при одинаковых значениях и поля1, и поля2, — то и подведение итогов имеет свой жесткий порядок: поле1, поле2, поле3, или: поле1, поле2, или: поле1.

Если таблица уже содержит итоговые строки, в нее можно добавить новые итоговые значения, рассчитанные с помощью другой функции. Для этого в окне *Промежуточные итоги* следует снять флажок *Заменить текущие итоги* и задать нужное поле и функцию (операцию). Два других флажка позволяют

размещать итоги под или над строками данных и выводить каждую группу значений данных на отдельном листе.

Если промежуточные итоги больше не нужны, то список можно привести в исходное состояние; для этого достаточно щелкнуть на кнопке *Отмена*. Но отмена срабатывает лишь в том случае, если после форматирования итогов не было других изменений списка; в противном случае следует щелкнуть на кнопке *Убрать все,* которая возвращает список в исходное состояние.

Команда меню *Данные → Итоги* позволяет выполнять следующие действия:

— по отдельному полю списка, используемому в качестве поля группировки, можно осуществлять формирование итогов различных видов операций (функций);

— для одинаковых значений полей группировки можно формировать итоги по одному или нескольким полям списка; при этом вид операции определяет, какие поля могут использоваться для подведения итогов. Так, для операций *Сумма, Среднее, Максимальное, Минимальное* и т. п. могут выбираться поля только числового типа, для операции *Количество значений* — поля любого типа (числовые, текстовые, даты).

Консолидация данных. Другим способом получения итоговой информации является *консолидация* — агрегирование согласно выбранной функции обработки данных, представленных в исходных *областях-источниках.* Результат консолидации находится в *области назначения.* Области-источники могут находиться на различных листах или рабочих книгах. В консолидации может участвовать до 255 областей-источников, а сами источники могут быть закрыты во время консолидации.

Для консолидации данных курсор устанавливается в область места назначения. Выполняется команда *Данные → Консолидация,* выбирается вариант и задаются условия консолидации. Существуют следующие варианты консолидации:

— по расположению для одинаково организованных источников (фиксированное расположение);

— по категориям для различающихся по расположению данных;

— консолидация внешних данных.

При консолидации *по расположению* все источники имеют одинаковое расположение данных источников, что позволяет

использовать ссылки на файлы и ячейки для консолидации таблицы (метки категорий данных в выделяемые области-источники не включаются). Данные имеют одинаковую структуру, фиксированное расположение ячеек и могут быть консолидированы с определенной функцией обработки (максимальное, минимальное, среднее значение и т. п.) по их расположению.

При консолидации *по категориям* области-источники содержат однотипные данные, но не одинаково организованные в различных областях-источниках. Для консолидации данных по категориям используются метки строк и столбцов, которые должны совпадать (метки включаются в выделенные области-источники). Метки и консолидируемые данные должны находиться в непосредственной близости друг с другом. Указывается тип меток — в верхней строке и левом столбце.

При консолидации *внешних данных* следует щелкнуть на кнопке *Пролистать,* в диалоговом окне *Пролистать* выбрать файл, содержащий области-источники, для добавления к списку, а затем добавить ссылку на ячейку или указать имя блока ячеек.

Переключатель *Создавать связи с исходными данными* создает при консолидации связи области назначения к областям-источникам.

Сортировка данных. В Microsoft Excel предусмотрен следующий порядок для сортировки данных по возрастанию: числа, текст, логические значения, значения ошибки, пустые ячейки. Для сортировки по убыванию используется обратная очередность, за исключением пустых ячеек, которые всегда помещаются в конец отсортированного списка.

Работа с диаграммами. Диаграмма создается с помощью *Мастера диаграмм*, вызываемого командой *Вставка → Диаграмма* или щелчком на соответствующей кнопке на панели инструментов *Стандартная.* При выборе опции *На этом листе* создается внедренная диаграмма. При выборе опции *На новом листе* автоматически добавится новый лист, на котором будет размещена создаваемая диаграмма.

Обычно перед вызовом *Мастера диаграмм* выделяется интервал ячеек — область данных для построения диаграммы. *Мастер диаграмм* осуществляет построение новой диаграммы в интерактивном режиме за несколько шагов:

— *шаг 1.* Указание блока ячеек с исходными данными для построения диаграммы. Блок ячеек может включать как сами данные, так и дополнительную информацию, которая используется в качестве названий исходных данных (легенд), указаний меток по оси X. Блок ячеек может содержать несмежные ячейки одного рабочего листа (выделяются при нажатии клавиши <*Ctrl*>);

— *шаг 2.* Выбор типа диаграмм. Excel позволяет построить диаграммы 14 стандартных типов;

— *шаг 3.* Выбор формата диаграммы указанного типа. Можно просмотреть результаты выбора, щелкнув на кнопке *Просмотр результатов;*

— *шаг 4.* Задание параметров диаграммы: расположение данных, способ использования первой строки или столбца (метки осей, текст легенды, название диаграммы).

Если блок ячеек для построения диаграммы содержит несколько строк или столбцов, можно различным образом определить понятие ряда. Ряд может соответствовать данным одного столбца или одной строки. Если интервал включает не только числовые данные, следует указать, сколько строк (ряды в строках) или столбцов (ряды в столбцах) отводится для меток оси X и сколько столбцов (ряды в строках) или строк (ряды в столбцах) используется при формировании легенды;

— *шаг 5.* Добавление легенды, ввод названия диаграммы и подписей к ее осям. Диаграмма может и не содержать легенд, если они не вошли в интервал выделения, но легенду можно добавить при редактировании рядов диаграммы.

Редактирование диаграмм. Созданные диаграммы можно корректировать вплоть до изменения состава и способа представления исходных данных, на основании которых она построена. Редактирование осуществляется с помощью как *Мастера диаграмм,* так и команд меню и инструментов панели *Диаграмма.*

Использование *Мастера диаграмм*. Данный вид корректировки диаграмм обеспечивает изменение:

— исходного интервала ячеек, на основании которого построена диаграмма;

— ориентации рядов;

— числа строк и столбцов, отводимых для меток оси X и названия легенды в диаграмме.

Предварительно следует выделить объект диаграммы. Затем щелчком на кнопке *Мастер диаграмм* запускается режим корректировки, состоящий из двух шагов:

— *шаг 1.* Корректировка интервала ячеек для диаграммы;

— *шаг 2.* Корректировка ориентации в рядах, определение меток оси X и легенд.

Работа с диаграммами с помощью команд меню. При активизации диаграммы происходит изменение состава режимов *Главного меню,* появляются специальные режимы, содержащие команды корректировки диаграмм. Вместо меню *Данные* появляется меню *Диаграмма,* которое содержит следующие команды:

Тип диаграммы — для изменения типа диаграммы для отдельной последовательности данных, группы или всей диаграммы в целом;

Исходные данные — для добавления или изменения выделенного ряда данных или отдельного элемента диаграммы;

Параметры диаграммы — для изменения стандартных параметров выбранного типа диаграмм. Изменения могут затрагивать такие элементы, как сетка, оси, подписи данных и заголовок диаграммы;

Размещение — для выбора расположения выделенных объектов на листе;

Добавить данные — для добавления выделенного ряда данных или точек на диаграмму;

Линия тренда (меню *Вставка*) — для добавления или изменения линии тренда на диаграммах различных типов.

Алгоритм изменения диапазона ячеек, используемый для создания диаграммы:

— выберите изменяемую диаграмму;

— выполните команду *Исходные данные* в меню *Диаграмма,* а затем выберите вкладку *Диапазон данных;*

— убедитесь, что выделена полностью вся ссылка в поле *Диапазон данных;*

— на рабочем листе выберите ячейки, содержащие данные, которые должны появиться в диаграмме;

— чтобы название столбца или строки для новых данных появилось в диаграмме, в выбираемые ячейки нужно включить те, которые содержат это название.

Настройка параметров печати. Печать готового документа на принтере — заключительный этап работы с электронными таблицами. Щелчок на кнопке *Печать* на панели инструментов *Стандартная* осуществляет автоматическую печать рабочего листа с параметрами настройки принтера, заданными по умолчанию. По умолчанию область печати совпадает с заполненной частью рабочего листа и представляет собой прямоугольник, примыкающий к верхнему левому углу рабочего листа. Если эти параметры надо изменить, то выполняется команда *Файл → Печать,* которая открывает диалоговое окно *Печать.*

Если часть данных не нужно выводить на печать, то можно задать область печати. **Область печати** — это заданный диапазон ячеек, который выдается на печать вместо рабочего листа.

ПРАКТИЧЕСКИЕ РАБОТЫ

Работа № 1. Редактирование рабочей книги

Цель работы: создание и сохранение электронной таблицы (рабочей книги). Изучение способов работы с данными в ячейке (форматирование содержимого ячеек, выбор диапазона ячеек и работа с ними, редактирование содержимого ячеек). Изучение возможностей *Автозаполнения.*

Задание: создайте рабочую книгу и изучите способы работы с данными в Excel.

Методика выполнения работы:

1. Создайте новую рабочую книгу (кнопка *Создать* на панели инструментов *Стандартная* или команда *Файл → Создать*).
2. Переименуйте текущий рабочий лист (дважды щелкните на ярлыке текущего рабочего листа и переименуйте его).
3. Добавьте еще один рабочий лист в рабочую книгу (щелкните правой кнопкой мыши на ярлыке листа и в контекстном меню выберите команду *Добавить*).
4. Сохраните созданный вами файл под именем book.xls в своем каталоге (команда *Файл → Сохранить*).

Таблица 4.4

Экзаменационная ведомость

№ п/п	Фамилия, инициалы	№ зачетной книжки	Оценка	Фамилия экзаменатора
1	Иванов И.И.	120		Иващенко И.И.
2	Петров В.В.	131		Иващенко И.И.
3	Сидоров С.С.	145		Иващенко И.И.
4	Федоров Ф.Ф.	119		Иващенко И.И.
5	Фролов Е.Е.	149		Иващенко И.И.
6	Демидов Д.Д.	121		Иващенко И.И.

5. Создайте таблицу по предложенному образцу (табл. 4.4). Для этого:

— в ячейку A1 введите с клавиатуры заголовок таблицы: *Экзаменационная ведомость;*

— в ячейку A3 введите: *№ п/п;*

— в ячейку B3 введите: *Фамилия, имя, отчество;*

— в ячейку C3 введите: *№ зачетной книжки;*

— в ячейку D3 введите: *Оценка;*

— в ячейку E3 введите: *Фамилия экзаменатора*.

6. Отформатируйте ячейки шапки табл. 4.4. Для этого:

— выделите блок ячеек— A3:E3;

— выполните команду *Формат → Ячейки* и перейдите к вкладке *Выравнивание*;

— в диалоговом окне *Выравнивание* выберите опции: *Горизонтальное — По центру; Вертикальное — По верхнему краю;* переключатель — *Переносить по словам,* а во вкладке *Шрифт* измените начертание букв и размер шрифта.

7. Измените ширину столбцов, в которые не поместились введенные данные. Для этого можно перетащить границы между строками и столбцами или навести указатель мыши на границу между заголовками столбцов и дважды щелкнуть левой кнопкой мыши. Для более точной настройки следует выполнить команду *Формат → Строка (Столбец)* и активизировать подходящую команду из открывающегося меню.

8. Сделайте рамку таблицы *(Панель инструментов →* кнопка *Обрамление)*.

9. Присвойте каждому студенту свой порядковый номер, используя *маркер заполнения*. Для этого:

— сделайте текущей первую ячейку столбца *№ п/п* и введите в нее цифру 1;

— введите цифру 2 в следующую ячейку этого столбца;

— выделите блок, состоящий из двух заполненных ячеек;

— установите указатель мыши на правый нижний угол выделенного блока. Указатель мыши примет вид черного крестика — это *маркер заполнения*. Перетащите его при нажатой правой кнопке мыши вниз или выполните команду *Правка → Заполнить → Прогрессия*).

10. Заполните столбец *Фамилия экзаменатора*. Воспользуйтесь методом *автозавершения* (он состоит в том, что Excel «угадывает» слово, которое будет вводить пользователь) или заполните ячейки с помощью *маркера заполнения*.

11. Скопируйте таблицу на другой рабочий лист с помощью буфера обмена. Для этого:

— выделите таблицу или диапазон ячеек;

— правой клавишей мыши вызовите контекстное меню;

— выполните в нем команду *Копировать;*

— перейдите на другой лист;

— установите курсор в первую ячейку предполагаемой таблицы;

— выполните команду *Вставить* в контекстном меню.

12. Добавьте в новую таблицу одну строку и один столбец. Для этого:

— выделите диапазон ячеек по столбцу;

— щелкните правой кнопкой мыши и в открывшемся контекстном меню выполните команду *Добавить ячейки;*

— то же самое повторите для строки.

13. Внесите в таблицу два изменения:

— очистите колонку с фамилией экзаменатора;

— озаглавьте эту колонку: *Подпись экзаменатора*.

14. Отсортируйте в новой таблице столбцы 2 и 3 по возрастанию (меню *Данные → Сортировка* или на панели инструментов *Стандартная* кнопка *Сортировать по возрастанию* (*Сортировать по убыванию*)).

15. Распечатайте созданный документ (*Файл → Печать*).

Работа № 2. Построение диаграмм

Цель работы: создание и построение диаграмм в Excel.

Задание: на основе данных, приведенных в табл. 4.5, постройте несколько типов диаграмм, наглядно показывающих итоги сессии.

Таблица 4.5

Сведения о результатах сдачи сессии на факультете

Средний балл по группе				
Группа	Информатика	Математика	История	Физика
И-123	4,2	3,8	4,5	4,3
И-124	4	4,4	4,4	4,2
И-125	3,9	4	4	3,9
И-126	4,3	4,4	4,4	4,1
И-127	3,8	4	4	3,9
И-128	3,3	3,9	3,9	3,6
И-129	4,5	4,8	4,8	3,9

Методика выполнения работы:

1. На *Листе 1* создайте таблицу *Сведения о результатах сдачи сессии на факультете;* внесите в нее данные.

2. Постройте на отдельном листе диаграмму типа *Столбчатая* или *График* для всех групп и всех предметов. Для этого:

— выделите всю таблицу;

— выполните команду меню *Вставка* → *Диаграмма* или воспользуйтесь кнопкой *Мастер диаграмм* на панели инструментов *Стандартная*.

3. Внесите название диаграммы, обозначения осей, добавьте легенду.

4. Постройте диаграммы и сравните результаты сдачи сессии по информатике, математике и физике. Для этого:

— выделите столбцы *Группа, Информатика, Математика* и, удерживая нажатой клавишу <*Ctrl*>, выделите столбец *Физика;*

— выберите тип диаграммы: *График*.

5. Измените результаты сдачи сессии и проверьте, как это отразилось на построенных диаграммах.

6. Отчет о работе представьте в виде диаграмм на отдельных листах рабочей книги.

Работа № 3. Формулы в Excel

Цель работы: создание и использование простых формул в Excel.

Задание 1: торговая фирма имеет в своем ассортименте следующие товары: телевизоры стоимостью $ 300, видеомагнитофоны стоимостью $ 320, музыкальные центры стоимостью $ 550, видеокамеры стоимостью $ 700, видеоплееры стоимостью $ 198 и аудиоплееры стоимостью $ 40. В январе было продано: телевизоров — 10, видеомагнитофонов — 5, музыкальных центров — 6, видеокамер — 2, видеоплееров — 7, аудиоплееров — 4. Используя возможности Excel, требуется найти сумму выручки от продажи в рублях и долларах.

Методика выполнения задания:

1. Создайте таблицу, внесите в нее исходные данные задачи.

2. Для подсчета выручки от продажи в долларах в ячейки столбца Е внесите соответствующие формулы. В формулах использована относительная адресация ячеек. Формула вводится лишь в одну ячейку, а остальные формулы в столбце получены с помощью *Автозаполнения*.

3. Подсчитайте выручку от продажи в рублях. В формулах использована смешанная и абсолютная адресация ячеек. Для введения абсолютного и смешанного адресов необходимо после введения ссылки нажать клавишу *<F4>* и из предлагаемых вариантов выбрать нужный.

4. Подсчитайте сумму выручки от продажи всех видов товаров (выделить столбец и щелкнуть на кнопке *Автосумма* на панели инструментов *Стандартная* или установить курсор в последнюю ячейку столбца Е в строку *Итого* и воспользоваться кнопкой *Вставка функции* на той же панели инструментов. В окне *Мастера функций* следует выбрать: СУММ из категории *Математические*).

Пример выполнения второго пункта задания 1 показан в табл. 4.6.

Таблица 4.6

Пример подсчета выручки

A	B	C	D	E	F	G
1	Наименование продукции	Цена за ед., долл.	Продано, шт.	Выручка от продажи, долл.	Выручка от продажи, руб.	Курс долл.
2	Телевизор	300	10	=C3×D3	=$E3×$G$3	27.1
3	Видеомагнитофон	320	5	=C4×D4	=$E4×$G$3	
4	Музыкальный центр	550	6	=C5×D5	=$E5×$G$3	
5	Видеокамера	700	2	=C6×D6	=$E6×$G$3	
6	Видеоплеер	198	7	=C7×D7	=$E7×$G$3	
7	Аудиоплеер	40	4	=C8×D8	=$E8×$G$3	
8	Итого сумма выручки			=СУММ(E3:E8)	=СУММ(F3:F8)	

Задание 2:

1. Изучите методику создания и применение простых формул, используя данные задания 1.

2. Сопоставьте доходность акции по уровню дивидендов за 1999 г. по отдельным эмитентам. Исходные данные задачи представлены в табл. 4.7.

Таблица 4.7

Доходность акций по отдельным эмитентам

Эмитент	Номинал акции, руб.	Цена продажи, руб.	Дивиденды, объявленные в расчете на год		Доходность по дивидендам	
	NA*	CP**	Div***, %	DivR, руб.	К номиналу DN	Фактическая DF
Сибирьгазбанк	10000	17780	400			
Инкомбанк	10000	22900	400			
Сургутнефтегазбанк	5000	5600	320			

Эмитент	Номинал акции, руб.	Цена продажи, руб.	Дивиденды, объявленные в расчете на год		Доходность по дивидендам	
	NA*	CP**	Div***, %	DivR, руб.	К номиналу DN	Фактическая DF
Нефтехимбанк	1000	2015	653			
Сбербанк	1000	2482	736			
КБ Аккобанк	1000	1000	325			
СКБ банк	50000	27050	360			
Промстройбанк	1000	1200	1535			

* NA — номинал акции;
** CP — цена продажи;
*** Div — дивиденды, объявленные в расчете на год.

3. Визуально проанализируйте полученные результаты.

Методика выполнения задания:

1. В соответствующие столбцы введите формулы для расчета выходных показателей:

$$DivR(i) = NA(i) \times Div(i);$$
$$DN(i) = Div(i);$$
$$DF(i) = DivR(i)/CP(i),$$

где i = [1,N]; N — число рассматриваемых эмитентов.

2. На основании исходного документа «Доходность акций по отдельным дивидендам» рассчитайте следующие значения:

— среднюю цену продажи акций по всем эмитентам (выделить столбец *Цена продажи* без заголовка, на панели инструментов *Стандартная* выполнить команду *Мастер функций* → категория *Статистическая* → функция =СРЗНАЧ);

— максимальную цену продажи акций по всем эмитентам (выделить столбец *Цена продажи* без заголовка, на панели инструментов *Стандартная* выполнить команду *Мастер функций* → категория *Статистическая* → функция =МАКС);

— минимальную цену продажи акций (выделить столбец *Цена продажи* без заголовка, на панели инструментов *Стан-*

дартная выполнить команду *Мастер функций* → категория *Статистическая* → функция =МИН);

— максимальную фактическую доходность акций по уровню дивидендов (выделить столбец *Фактическая доходность* без заголовка, выполнить команду *Мастер функций* → категория *Статистическая* → функция =МАКС);

— минимальную фактическую доходность акций по уровню дивидендов (выделить столбец *Фактическая доходность* без заголовка, выполнить команду *Мастер функций* → категория *Статистическая* → функция =МИН).

3. Результаты расчетов оформите в виде табл. 4.8.

Таблица 4.8

Цены и доходность акций

Расчетная величина	Значение
Средняя цена продажи акций	
Максимальная цена продажи акций	
Минимальная цена продажи акций	
Максимальная фактическая доходность акций	
Минимальная фактическая доходность акций	

4. В исходной таблице отсортируйте записи в порядке возрастания фактической доходности по дивидендам (выделить таблицу без заголовков и без строки *Среднее значение,* выполнить команду меню *Сортировка → Данные*).

5. Выполните фильтрацию таблицы, выбрав из нее только тех эмитентов, фактическая доходность которых больше средней по таблице. Алгоритм фильтрации:

— выделить данные таблицы с прилегающей одной строкой заголовка;

— выполнить команду меню *Данные → Фильтр → Автофильтр;*

— в заголовке столбца *Фактическая доходность* щелкнуть на кнопке раскрывающегося списка и выбрать *Условие;*

— в окне пользовательского *Автофильтра* задать условие: *> среднее значение*.

6. Результаты фильтрации поместите на новый рабочий лист, включив в него следующие графы:

— *Эмитент;*
— *Номинал акции;*
— *Цена продажи;*
— *Доходность по дивидендам фактическая.*

7. Постройте на отдельном рабочем листе Excel круговую диаграмму, отражающую фактическую доходность по дивидендам каждого эмитента в виде соответствующего сектора (выделить столбцы *Эмитент* и *Фактическая доходность*, выполнить команду меню *Вставка → Диаграмма*). На графике показать значения доходности, вывести легенду и название графика: *Анализ фактической доходности акций по уровню дивидендов.*

8. Постройте на новом рабочем листе Excel смешанную диаграмму, в которой представьте в виде гистограмм значения номиналов и цены продажи акций каждого эмитента, а их фактическую доходность покажите в виде линейного графика на той же диаграмме. Выведите легенду и название графика: *Анализ доходности акций различных эмитентов.* Алгоритм построения смешанного графика:

— выделить столбцы *Эмитент, Номинал акции* и *Цена продажи;*

— выполнить команду меню *Вставка → Диаграмма →* тип диаграммы *Гистограмма;*

— для добавления линейного графика *Фактическая доходность по дивидендам* правой кнопкой мыши активизировать меню *Диаграмма → Исходные данные →* вкладка *Ряд →* кнопка *Добавить;* в поле *Имя* ввести название ряда: *Доходность,* в поле *Значения* ввести числовой интервал, соответствующий фактической доходности по дивидендам;

— на полученной диаграмме установить курсор мыши на столбец, соответствующий значению *Доходность*, правой кнопкой мыши активизировать контекстное меню, выполнить команду *Тип диаграммы → График.*

9. Подготовьте результаты расчетов и диаграммы к выводу на печать (меню *Файл → Печать*).

Работа № 4. Сортировка данных в списке

Цель работы: выполнение сортировки данных в Excel.

Задание:

1. Выполнить сортировку данных табл. 4.9 по возрастанию кода предмета, даты проведения занятия, номера группы.

Таблица 4.9

Результаты занятий

A	B	C	D	E	F	G	H
1	№ группы	№ зачетной книжки	Код предмета	Таб. № преподавателя	Вид занятия	Дата	оценка
2	133	1	П1	А1	Практика	26.05.07	3
3	134	2	П2	А2	Лекция	26.05.07	4
4	133	1	П1	А1	Лекция	11.06.07	4
5	134	2	П1	А2	Лекция	11.06.07	5
6	135	3	П2	А1	Практика	16.05.07	2
7	133	4	П2	А3	Лекция	20.05.07	3
8	133	4	П1	А1	Лекция	16.05.07	3
9	135	3	П1	А3	Лекция	16.05.07	4
10	133	5	П1	А2	Лекция	26.05.07	4
11	135	5	П2	А1	Лекция	11.06.07	2
12	135	5	П1	А2	Практика	20.05.07	5
13	136	6	П2	А1	Лекция	26.05.07	5
14	136	6	П2	А2	Практика	11.06.07	5
15	135	3	П1	А3	Лекция	20.05.07	4
16	135	3	П1	А1	Практика	16.05.07	3
17	134	2	П2	А2	Лекция	20.05.07	4

2. Выполнить сортировку данных табл. 4.9 по возрастанию, используя сочетания признаков: код предмета и дата проведения занятия; код предмета и номер группы; номер группы и дата проведения занятия, а также сочетание всех трех признаков.

Методика выполнения работы:

1. Создайте новую рабочую книгу (меню *Файл → Создать*) и сохраните ее под именем SORT.XLS в рабочем каталоге (меню *Файл → Сохранить как...*).

2. Сформируйте таблицу результатов занятий.

3. Отформатируйте шапку таблицы следующим образом:

— шрифт — Times New Roman;

— размер шрифта — 12 пт, курсив;

— выравнивание по горизонтали — *По значению*;

— выравнивание по вертикали — *По верхнему краю*;

— установите ключ *Переносить По Словам* (выделите соответствующие ячейки и выполните команду *Формат → Ячейки*).

4. Выполните сортировку по столбцу *Код предмета,* расположив коды предметов по возрастанию. Для этого:

— выделите таблицу с одной строкой заголовка;

— выполните команду меню *Данные → Сортировка;*

— в окне *Сортировка диапазона* в строке *Сортировать по* выберите: *Код предмета*.

5. Результат сортировки скопируйте на *Лист 2:*

— выделите всю таблицу, выполните команду *Правка → Копировать;*

— на *Листе 2* установите курсор в ячейку A1 и выполните команду *Правка → Вставить*.

6. Переименуйте *Лист 2,* присвоив ему имя *Сортировка*. Для этого:

— установите указатель мыши на ярлычке *Лист 2;*

— вызовите правой кнопкой мыши контекстное меню;

— выполните команду *Переименовать*.

7. Выполните сортировку по столбцу *Дата*, расположив данные по возрастанию: установите курсор в любую ячейку поля *Дата* и выполните команду *Сортировка* в меню *Данные*. При этом должны выделиться вся область списка, а в окне *Сортировка Диапазона* в строке *Сортировать по* — столбец G. Если этого не произошло, то предварительно выделите весь список, а затем выполните указанную команду.

8. Выполните сортировку по сочетанию признаков: *Дата, № группы, Код предмета*. Для этого выделите всю таблицу и в диалоговом окне *Сортировка* установите:

— в строке *Сортировать по* — поле *Дата,* по возрастанию;

— в строке *Затем* — поле *№ группы,* по возрастанию;

— в следующей строке *Затем* — поле *Код предмета,* по возрастанию;

— флажок *Строка меток столбцов*.

Результат сортировки скопируйте на *Лист 3* и переименуйте его в *Сортировка 2.*

Работа № 5. Фильтрация записей

Цель работы: ознакомиться со способами фильтрации записей списка и автофильтрации, с работой с формой данных.

Задание: изучите способы фильтрации записей и сформируйте условия отбора записей.

Методика выполнения работы:

1. Создайте новую рабочую книгу с названием *Фильтрация*.
2. Скопируйте в новую рабочую книгу таблицу, созданную в работе № 4 (табл. 4.9).
3. Переименуйте Лист 1, присвоив ему имя *Автофильтр № 1.*
4. Чтобы применить *Автофильтрацию*, установите курсор в область списка и выполните команду *Данные → Фильтр → Автофильтр.*
5. Сформируйте условия отбора: для преподавателя А1 выбрать сведения о сдаче экзамена на положительную оценку, вид занятий: *Лекция*. Для этого:

— в столбце *Таб № преподавателя* щелкните на кнопке *Фильтр*, из списка условий отбора выберите: *А1;*

— в столбце *Оценка* щелкните на кнопке *Фильтр*, из списка условий отбора выберите: *Условие*, в диалоговом окне сформируйте условие отбора: *> 2;*

— в столбце *Вид занятий* щелкните на кнопке *Фильтр,* из списка условий отбора выберите: *Лекция.*

6. Результат фильтрации скопируйте на новый лист, присвоив ему имя *Автофильтр № 2.*
7. На листе *Автофильтр № 1* результат автофильтрации отмените, установив указатель мыши в область списка и выполнив команду *Данные → Фильтр → Автофильтр.*
8. Сформулируйте выборку: для группы № 133 получите сведения о сдаче экзамена по предмету П1 на оценки «3» и «4».
9. Результат сохраните на новом листе, присвоив ему имя *Автофильтр № 3.*
10. Скопируйте исходную таблицу на новый рабочий лист, переименовав его в *Форма данных.*
11. Установите курсор в область списка и выполните команду *Данные → Форма.*

12. В окне *Форма данных* просмотрите записи списка и внесите необходимые изменения по своему усмотрению с помощью кнопок *Предыдущая* и *Следующая*.

13. С помощью кнопки *Создать* добавьте новые записи.

14. В окне *Форма данных* сформируйте условия отбора записей. Для этого щелкните на кнопке *Критерии,* название которой поменяется на название *Правка*. В пустых строках имен полей списка введите критерии:

— в строку *Таб. № преподавателя* введите: *А1;*

— в строку *Вид* занятия введите: *Лекция;*

— в строку *Оценка* введите условие: *> 2.*

15. Просмотрите отобранные записи, щелкнув на кнопке *Предыдущая* или *Следующая*.

16. По аналогии сформулируйте условия отбора записей, указанные в пункте 8.

Работа № 6. Создание базы данных

Цель работы: создание базы данных средствами Excel. Сортировка данных, выборка по различным критериям, поиск записи. Автоматическое подведение итогов.

Задание: создайте базу данных, внесите записи, выполните сортировку данных.

Методика выполнения работы:

1. Создайте таблицу по предложенному образцу (табл. 4.10).

Таблица 4.10

Вид таблицы для создания БД

№ п/п	Фамилия	Имя	Отчество	Дата рождения	Адрес					Телефон	Оклад	Налоги			Сумма к выдаче
					Город	Улица	Дом	Корпус	Квартал			Проф.	Пенсион.	Подоход.	

2. Для ячеек *Дата рождения* установите формат: *Дата (Формат → Ячейка → Число).*

3. Для ячеек *Дом*, *Квартира* установите числовой формат.

4. Для ячеек *Телефон* установите формат: *Номер телефона (Формат → Ячейка → Дополнительный → Номер телефона)*.

5. Для ячеек *Оклад, Налоги, Сумма к выдаче* установите: *Денежный формат*.

6. В ячейку столбца *Налоги профсоюзные* введите формулу для подсчета налогов.

7. В ячейку столбца *Налоги пенсионные* введите соответствующую формулу.

8. В ячейку столбца *Налоги подоходные* введите формулу: *13 % от оклада за вычетом минимальной заработной платы и пенсионного налога*. Минимальную заработную плату принять равной 2000 руб.

9. Внесите первую запись. Начиная со второй записи, заполните таблицу, используя команду меню *Данные → Форма*. Перед выполнением команды выделите первую запись таблицы и прилегающую к ней строку заголовка. Таблица должна содержать не менее 20 записей.

10. Выполните сортировку данных по фамилии *(Данные → Сортировка);* результат сортировки сохраните на *Листе 2.*

11. Отсортируйте исходные данные по возрастанию окладов; результат сохраните на *Листе 3.*

12. Получите список людей, проживающих, например, по улице Мира *(Данные → Фильтр → Автофильтр);* результат сохраните на *Листе 4.*

13. Получите список людей, телефоны которых начинаются с цифры, большей «3»; результат сохраните на *Листе 5.*

Работа № 7. Использование логических функций

Цель работы: ознакомление с использованием логических функций в Excel.

Задание 1:

1. Подсчитайте количество отличных, хороших и т. д. оценок на основании зачетной ведомости, представленной в табл. 4.11.

2. Расчет произведите, используя операцию *Присвоение имени блоку ячеек*.

Таблица 4.11

Таблица рабочей книги в Excel

№ строки / Имя столбца	A	B	C	D	E	F	G	H	I
1	№ п/п	Ф.И.О.	№ зач. книжки	Оценка	Кол-во 5	Кол-во 4	Кол-во 3	Кол-во 2	Неявка
2	1	Демидов М.И.	119	5					
3	2	Иванов И.П.	120	4					
4	3	Кукушкин В.Л.	121	3					
5	4	Орлов А.П.	131	4					
6	5	Петров К.Н.	145	5					
7	6	Сидоров Р.О.	149	2					
8	7	Фролов В.А.	156	н/я					

Методика выполнения задания:

1. На новом листе рабочей книги создайте таблицу по образцу табл. 4.11.

2. Заполните данными первый, второй, третий и четвертый столбцы.

3. В шестой, седьмой, восьмой, девятый и десятый столбцы введите формулы, для этого воспользуйтесь *Мастером функций* на панели инструментов *Стандартная:*

— установите курсор в первую ячейку столбца отличных оценок (D2) и активизируйте *Мастера функций;*

— в первом диалоговом окне выберите категорию и название функции:

Категория: Логические функции
Имя функции: ЕСЛИ

— щелкните на кнопке *Готово;*

— во втором диалоговом окне установите курсор в поле *Логическое выражение* и щелкните в рабочей области Excel на ячейке D2 (оценка «5»);

— с клавиатуры введите: *<= 5>;*

— в поле *Значение_если_истина* введите: *<1>;*

— в поле *Значение_если_ложь* введите: *<0>;*

— щелкните на кнопке *Готово;*

— методом протягивания скопируйте формулу по столбцу *Кол-во 5.*

4. С помощью *Мастера функций* аналогичным способом введите формулы в столбцы *Кол-во 4, Кол-во 3* и т. д., изменяя значение поля *Логическое_выражение* на: D2 = 4, D2 = 3 и т. д.

5. Чтобы подсчитать сумму всех «пятерок», «четверок» и т. д. и представить результаты в виде отдельной таблицы, нужно по каждому столбцу *Кол-во оценок* задать имена блокам соответствующих ячеек. Для этого:

— выделите блок ячеек E2:E8 столбца *Количество 5;*

— выполните команду меню *Вставка → Имя → Присвоить;*

— в диалоговом окне *Присвоение имени* в строке *Имя* введите слово *Отлично* и щелкните на кнопке *Добавить;*

— выделите ячейки F2:F8 столбца *Количество 4* и выполните команду *Вставка → Имя → Присвоить;*

— в диалоговом окне *Присвоение имени* в строке *Имя* введите: *Хорошо;*

— аналогичные действия выполните с остальными столбцами табл. 4.11, создав имена блоков ячеек: *Удовлетворительно, Неудовлетворительно* и *Неявка.*

6. Создайте таблицу *Итоги сессии* (табл. 4.12).

Таблица 4.12

Итоги сессии

Итоги сессии	
Количество отличных оценок	
Количество хороших оценок	
Количество удовлетворительных оценок	
Количество неудовлетворительных оценок	
Неявки	
Итого	

7. Введите формулу подсчета количества полученных оценок определенного вида, используя имена блоков ячеек с помощью *Мастера функций:*

— установите курсор в ячейку подсчета количества отличных оценок;

— щелкните на кнопке *Мастера функций* на панели инструментов *Стандартная;*

— в первом диалоговом окне выберите категорию функции: *Математические,* имя функции: СУММ и щелкните на кнопке *ОК;*

— во втором диалоговом окне установите курсор в строку *Число 1* и выполните команду *Вставка → Имя → Вставить;*

— в диалоговом окне *Вставка имени* выберите имя блока ячеек: *Отлично* и щелкните на кнопке *ОК;*

— повторите аналогичные действия для подсчета количества других оценок.

8. Подсчитайте количество всех полученных оценок, щелкнув на кнопке *Автосумма* на панели инструментов *Стандартная*.

Задание 2: определить, в какой из заданных интервалов попадает зарплата каждого сотрудника НИИ, показанная в табл. 4.13.

Таблица 4.13

Пример выполнения практической работы

№ строки / Имя столбца	A	B	C	D	E	F	G	H
1	№ п/п	Ф.И.О.	Зарплата	1 ин	2 ин	3 ин	4 ин	Проверка
2	1	Кузнецов А.А.	5896	0	0	0	1	1
3	2	Свиридов А.Б.	3990	0	0	1	0	1
4	3	Молотов В.В.	2098	0	1	0	0	1
5	4	Иванов П.П.	1980	1	0	0	0	1
6	5	Петров И.И.	2346	0	1	0	0	1
7	ИТОГО			1	2	1	1	5

ЕСЛИ(И(C2>A10;C2=<B10);1;0)

Методика выполнения задания:

1. Создайте новую рабочую книгу.

2. Создайте таблицу из девяти столбцов, в которой содержатся сведения о пяти сотрудниках НИИ: № п/п, Ф.И.О., ежемесячная зарплата (табл. 4.13).

3. Создайте таблицу, содержащую четыре интервала числовых значений зарплат: 1000—2000, 2000—3000, 3000—4000, 4000—6000 (табл. 4.14).

Таблица 4.14

Интервалы зарплаты

№ п/п	A	B
	Интервалы	
1 ин.	1000	2000
2 ин.	2000	3000
3 ин.	3000	4000
4 ин	4000	6000

4. Чтобы определить, попадает ли значение зарплаты из столбца С в заданный интервал, нужно использовать логическую функцию *Если* с заданием сложного условия *И*. Для этого:

— установите курсор в ячейку D2;

— щелкните на значке *Вставка функции* на панели инструментов *Стандартная*;

— в окне *Мастера функций* выберите категорию функции: *Логические*;

— в окне *Вид функции* выберите функцию *Если*;

— щелкните на кнопке *ОК*;

— в адресной строке рабочего окна в раскрывающемся списке выберите функцию *И*;

— установите курсор в поле *Логическое 1*;

— на рабочем поле Excel щелкните на ячейке С2;

— с клавиатуры введите: >;

— на рабочем поле Excel щелкните на ячейке А10;

— установите курсор в поле *Логическое 2*;

— на рабочем поле Excel щелкните на ячейке С2;

— с клавиатуры введите: <;

— на рабочем поле Excel щелкните на ячейке B10;

— не закрывая окно функции *И*, щелкните на слове *Если* в адресной строке рабочего окна — откроется окно функции *Если*;

— в поле *Значение_если_истина* с клавиатуры введите: < 1 >;

— в поле *Значение_если_ложь* с клавиатуры введите: < 0 >;

— щелкните на кнопке *OK*.

5. Формулу из ячейки D2 операцией *Автозаполнения* скопируйте по столбцу D, ссылки на ячейки A10 и B10 сделайте абсолютными.

6. Аналогичным образом введите формулы в столбцы E, F, G.

7. Для подсчета числа попаданий в каждый интервал выполните следующие действия:

— выделите блок D2:D6;

— щелкните на кнопке *Автосумма* на панели инструментов *Стандартная;*

— повторите это действие для каждого столбца.

8. Значения столбца *Проверка* получите, используя операцию *Автосумма* для значений блоков строк D2:G2, D3:G3 и т. д.

9. Значение ячейки *Итого* столбца *Проверка* должно совпадать с общей суммой зарплаты, выданной сотрудникам.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ТЕМЕ 4.9

1. **Каково назначение электронной таблицы?**
2. **Как называется документ в программе Excel? Из чего он состоит?**
3. **В чем заключаются особенности типового интерфейса табличных процессоров?**
4. **Какие типы данных могут содержать электронные таблицы?**
5. **Какие данные называют зависимыми, а какие независимыми?**
6. **По какому признаку программа определяет, что введенные данные являются не значением, а формулой?**
7. **Что в Excel используется в формулах в качестве операндов?**
8. **Что такое формула в электронной таблице и каковы ее типы? Приведите примеры.**
9. **Что такое функция в электронной таблице и каковы ее типы? Приведите примеры.**
10. **Поясните, для чего используются абсолютные и относительные адреса ячеек?**
11. **Что такое *Автозаполнение?***
12. **В чем состоит приоритет выполнения операций в арифметических формулах Excel?**

13. **Как можно «размножить» содержимое ячейки?**
14. **Как посмотреть и отредактировать формулу, содержащуюся в ячейке?**
15. **Какой тип адресации используется в Excel по умолчанию?**
16. **В чем состоит удобство применения относительной и абсолютной адресации при заполнении формул?**
17. **Что такое диапазон? Как его выделить?**
18. **Как защитить содержимое ячеек электронной таблицы от несанкционированного доступа? Как внести в ячейки изменения?**
19. **Какие вы знаете типы диаграмм, используемых для интерпретации данных электронной таблицы? Поясните, когда следует или не следует использовать каждый из них.**
20. **Какие способы объединения нескольких исходных электронных таблиц в одну вам известны?**
21. **Каковы особенности печати документов в Excel?**
22. **Как использовать электронную таблицу для моделирования решения задачи по типу «что будет, если…».**
23. **Как выделить смежные и несмежные блоки ячеек?**
24. **Какие вы знаете команды для работы с базами данных?**
25. **Что такое консолидация таблиц?**
26. **Что такое макросы и для чего они используются?**
27. **Какие вы знаете форматы данных?**
28. **Какие вы знаете типы аргументов функции?**
29. **Что такое *Мастер функции?***
30. **Что такое *Мастер диаграмм?***
31. **Какие вы знаете методы обработки и анализа данных в Excel?**
32. **Как осуществляется сортировка списков?**
33. **Как осуществляется фильтрация списков?**
34. **В каких случаях используют структурирование и группировку данных?**
35. **Как формируются итоги в списке по заданным критериям?**

Тема 4.10

СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ ACCESS

4.10.1. Общие сведения

База данных — это реализованная с помощью компьютера информационная модель, отражающая состояние объектов и их отношения. **Информационной моделью** (или **структурой данных**)

называют совокупность взаимосвязанных данных. Базы данных (БД) делят на три класса соответственно типам информационных структур: табличные (реляционные), сетевые, иерархические.

С понятием базы данных тесно связано понятие системы управления базой данных (СУБД). СУБД — это комплекс программных средств, предназначенных для создания структуры новой базы, ее наполнения содержимым, редактирования содержимого, отбора отображаемых данных в соответствии с заданным критерием, упорядочения, оформления и последующей выдачи на устройства вывода или передачи по каналам связи.

В мире существует множество СУБД: dBase, Paradox, FoxPro, Clipper, Oracle и т. д. Несмотря на то что они могут по-разному работать с разными объектами и предоставляют пользователю различные функции и средства, большинство СУБД опираются на единый устоявшийся комплекс основных понятий. Это дает возможность рассмотреть одну систему и обобщить ее понятия, приемы и методы на весь класс СУБД. В качестве такого объекта в данном учебнике выбрана СУБД Microsoft Access.

Реляционная база данных. БД, созданная в СУБД Access, является реляционной базой данных, основным объектом которой служат взаимосвязанные двухмерные таблицы, состоящие из однотипных строк-записей. Каждая строка, в свою очередь, составлена из полей и называется **записью**. Если записей в таблице нет, то это значит, что структура БД образована только набором полей. Изменив состав полей базовой таблицы, мы изменяем структуру БД и соответственно получаем новую БД.

Для однозначного определения каждой записи таблица должна иметь уникальный ключ (первичный ключ), который может состоять из одного или нескольких полей. По значению ключа отыскивается единственная запись.

Связи между таблицами БД дают возможность совместно использовать данные из разных таблиц. В нормализованной реляционной БД связи характеризуются отношениями типа «один к одному» (1 : 1) или «один ко многим» (1 : М). Связь каждой пары таблиц обеспечивается одинаковыми полями в них — ключом связи. Ключом связи всегда является уникальный ключ главной таблицы в связи. В подчиненной таблице он называется **внешним ключом**.

Схема данных. В СУБД Access процесс создания реляционной БД включает создание схемы данных. Схема данных наглядно отображает таблицы и связи между ними и обеспечивает использование связей при обработке данных. В схеме данных устанавливаются параметры обеспечения связной целостности в БД.

Поскольку СУБД Access является одним из приложений Windows, входящих в интегрированную систему Office, интерфейс окна программы и его основные компоненты: меню, панели инструментов, справочная система, — а также приемы работы с клавиатурой и мышью используются аналогично другим приложениям (Word, Excel). Общий вид окна программы приведен на рис. 4.25.

Поля БД не просто определяют структуру базы — они еще определяют групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей.

Основные свойства полей таблиц баз данных СУБД Microsoft Access*:*

имя поля — определяет, как следует обращаться к данным этого поля при автоматических операциях с базой (по умолчанию имена полей используются в качестве заголовков столбцов таблиц);

тип поля — определяет тип данных, которые могут содержаться в данном поле;

размер поля — определяет предельную длину (в символах) данных, которые могут размещаться в данном поле;

формат поля — определяет способ форматирования данных в ячейках, принадлежащих полю;

маска ввода — определяет форму, в которой вводятся данные в поле (средство автоматизации ввода данных);

подпись — определяет заголовок столбца таблицы для данного поля (если подпись не указана, то в качестве заголовка столбца используется свойство *Имя поля*);

значение по умолчанию — значение, которое вводится в ячейки поля автоматически (средство автоматизации ввода данных);

условие назначения — ограничение, используемое для проверки правильности ввода данных;

сообщение об ошибке — текстовое сообщение, которое выдается автоматически при попытке ввода в поле ошибочных данных;

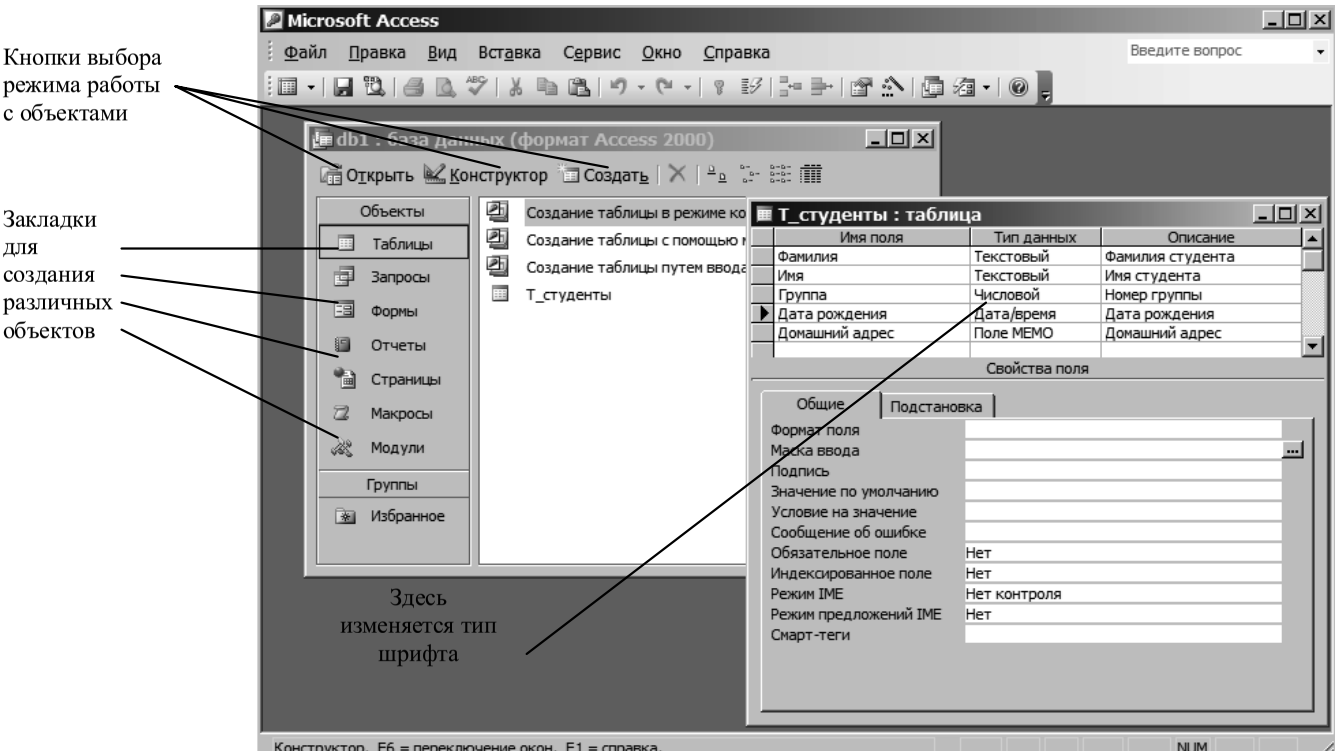


Рис. 4.25. Окно СУБД Microsoft Access

обязательное поле — свойство, определяющее обязательность заполнения данного поля при наполнении базы;

пустые строки — свойство, разрешающее ввод пустых строковых данных (в основном это касается текстовых данных);

индексированное поле — если поле обладает этим свойством, все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. По этому полю также проверяются значения записей на наличие повторов.

Свойства полей различаются в зависимости от типа данных. Базы данных Microsoft Access работают со следующими типами данных:

текстовый — тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов);

поле Мемо — специальный тип для хранения больших объемов текста (до 65 535 символов). Физически в поле хранится лишь указатель на то место БД, в котором хранится непосредственно текст, но для пользователя такое разделение незаметно;

числовой — тип данных для хранения действительных чисел;

дата/время — тип данных для хранения календарной даты и текущего времени;

денежный — тип данных для хранения денежных сумм;

счетчик — специальный тип данных для хранения для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием;

логический — тип для хранения логических данных (могут принимать только два значения, например ДА или НЕТ);

поле объекта OLE — специальный тип данных, предназначенный для хранения объектов *OLE*, например мультимедийных;

гиперссылка — специальное поле для хранения адресов URL Web-объектов Internet (URL — Universal Resource Locator — универсальный местоопределитель ресурсов). При щелчке на ссылке автоматически происходят запуск браузера и воспроизведение объекта в его окне.

Перед созданием БД должна быть определена логическая структура базы — состав таблиц, их структура и межтабличные связи.

Объектами БД, помимо таблиц, являются запросы, формы, отчеты, макросы и модули, создание которых существенно упрощает работу пользователя с массивами данных.

Запросы обеспечивают быстрый и эффективный доступ к данным из одной или нескольких связанных таблиц. Результатом выполнения запроса является таблица, которая может быть использована наряду с другими таблицами БД при обработке данных. Запросы позволяют исключить несанкционированный доступ к конфиденциальной информации, содержащейся в основных таблицах. При работе с запросами данные можно упорядочивать, фильтровать, объединять и производить с ними необходимые итоговые вычисления. Запрос может формироваться с помощью QBE (Query By Example) — запросов по образцу или с помощью инструкции SQL (Structured Query Language) — языка структурированных запросов.

Формы служат для организации внесения информации в таблицы различными пользователями, их применение позволяет легко автоматизировать ввод данных и исключить ошибки ввода. Формы могут также использоваться для просмотра запросов и таблиц на экране.

Отчет формируется для создания бумажного документа, т. е. для распечатки данных.

Для реализации практических задач пользователя разработчику приходится применять средства программирования: язык макрокоманд и язык Visual Basic for Applications (VBA). Макросы и модули, созданные программистом, оперируют с запросами, формами и отчетами и объединяют разрозненные действия в единую задачу пользователя.

В окне БД Access наряду со списком объектов представлены ярлыки для быстрого запуска *Мастеров* или *Конструкторов* создания нового объекта.

Мастера Access позволяют автоматизировать процесс создания таблиц базы данных, форм, запросов, отчетов и страниц доступа к данным.

Размещение базы данных. Все таблицы БД, а также другие объекты Access — формы, запросы, отчеты, макросы и модули, построенные для этой базы, и внедренные объекты — могут размещаться на диске в одном файле БД формата .mdb. Это упрощает технологию ведения базы данных и приложения поль-

зователя. Обеспечиваются высокая компактность размещения всех объектов БД на диске и эффективность обработки данных.

Приложение БД, которое создается программой VBA, может быть скомпилировано и сохранено в файле приложения формата .mde. При этом исходные программы на VBA удаляются, а БД сжимается, что значительно сокращает размер файла. После компиляции объекты БД не могут быть изменены.

При работе с БД коллективного пользования в сети с файловым сервером Access предоставляет возможность записать в отдельный файл от базы данных на сервере те объекты, которые составляют приложение пользователя. Этот файл размещается на всех компьютерах пользователей, работающих с общей базой данных. Приложение можно модифицировать в соответствии с потребностями пользователя.

В Access включены средства разработки проекта-приложения, обеспечивающего работу с БД, размещенной на SQL-сервере. Проект размещается в файле .adp на компьютере пользователя. При создании проекта пользователь может создать БД на SQL-сервере или использовать уже существующую.

ПРАКТИЧЕСКИЕ РАБОТЫ

Работа № 1. Создание объектов базы данных

Цель работы: разработка информационной модели базы данных. Создание объектов базы данных.

Задание: требуется создать базу данных, содержащую сведения о студентах. Из общего списка студентов необходимо выбрать обучающихся в определенной группе. Разработать специальную форму для ввода данных в таблицу. Организовать соответствующий отчет для вывода на печать списка студентов.

Методика выполнения работы:

Определим логическую структуру создаваемой БД.

Поскольку почти все поля БД являются уникальными, создаем одну сводную таблицу, состоящую из записей, в которые входят поля *Фамилия*, *Имя*, *Группа*, *Дата рождения* и *Домашний адрес*.

1. Загрузите Microsoft Access.

2. В появившемся окне Microsoft Access выберите переключатель *Новая база данных*. Щелкните на кнопке *OK*.

3. В раскрывающемся списке *Папка* окна *Файл* новой базы данных установите свой сетевой диск и свой каталог. В наборном поле *Имя файла* введите с клавиатуры имя создаваемой базы данных: STUDENT. Щелкните на кнопке *Создать*.

4. Откроется окно БД. Оно является основным окном базы и позволяет открывать, добавлять и удалять любые объекты базы данных.

Создание структуры новой таблицы

5. В появившемся окне STUDENT: *База данных* выберите объект *Таблицы* и щелкните на кнопке *Создать*.

6. В появившемся окне *Новая таблица* выберите режим работы: *Конструктор*. Щелкните на кнопке *OK*.

7. В первой строке в столбце *Имя поля* введите: *Фамилия*. Нажмите клавишу <→> (*Вправо*).

8. Щелкните на появившейся стрелке раскрывающегося списка ячейки *Тип данных* и выберите пункт: *Текстовый*.

9. В нижней части экрана — в *Свойствах поля* — на вкладке *Общие* в строке *Размер поля* установите: *20*.

10. В строке *Обязательное поле* с помощью кнопки раскрывающегося меню укажите: *Да*.

11. Установите курсор в первой строке в столбце *Описание*. Введите: *Фамилия студента*.

12. Во второй строке в столбце *Имя поля* введите: *Имя*. Установите тип данных: *Текстовый*.

13. В поле *Описание* введите: *Имя студента*. В строке *Размер поля* установите: *10*.

14. В третьей строке в столбце *Имя поля* введите: *Группа*. Установите тип данных: *Числовой*.

15. В раскрывающемся списке *Размер поля* выберите: *Целое*.

16. В поле *Описание* введите: *Номер группы*.

17. В четвертой строке в столбце *Имя поля* введите: *Дата рождения*.

18. Установите тип данных: *Дата/время*. Установите курсор в наборном поле *Формат поля*.

19. В раскрывающемся списке *Формат поля* установите: *Краткий формат даты*.

20. Установите курсор в поле *Маска ввода,* находящееся в нижней части экрана, и щелкните на кнопке с тремя точками. На запрос подтвердите сохранение таблицы под именем *Таблица1* и создание ключевого поля.

21. В появившемся диалоговом окне *Создание масок ввода* выберите: *Краткий формат даты.* Щелкните на кнопке *Далее.*

22. В следующем окне можно выбрать знак заполнителя для отображения в поле. Щелкните на кнопке *Готово.*

23. Обратите внимание: в первой строке Microsoft Access автоматически добавил дополнительное ключевое поле *Код*, которое играет роль уникального идентификатора записей, и установил для него тип данных: *Счетчик*.

24. В шестой строке в столбце *Имя поля* введите: *Домашний адрес*.

25. Установите тип данных: *Поле MEMO.* В поле *Описание* введите: *Домашний адрес.*

26. Закройте текущее окно *Таблица 1: таблица* с сохранением изменений.

Заполнение таблицы

27. В окне STUDENT: *База данных* выберите объект *Таблицы,* установите курсор на название: *Таблица1* и щелкните на кнопке *Открыть.*

28. В поле *Фамилия* введите свою фамилию, в поле *Имя* — свое имя, в поле *Группа* — номер своей группы.

29. Переместите курсор в поле *Дата рождения* и введите дату своего рождения в формате ДД.ММ.ГГ, например: 12.05.90. Вводить следует только числа, а остальное Microsoft Access подставит автоматически по заданной маске.

30. Переместите курсор в поле *Домашний адрес* и введите свой домашний адрес.

31. Подобным же образом введите еще семь записей. В поле *Группа* в любых двух строках введите номер группы: 271, в остальных строках введите: 272. Если потребуется изменить ширину столбца, то это можно сделать с помощью мыши аналогично работе в Excel.

32. Обратите внимание, что в поле *Код* цифры изменяются автоматически по мере ввода новых строк, каждый раз увеличиваясь на единицу.

33. Щелкните на значке закрытия текущего окна *Таблица1: таблица*.

34. В окне STUDENT: *База данных* установите курсор мыши на слово *Таблица1* и правой кнопкой мыши вызовите контекстное меню.

35. Выполните команду *Предварительный просмотр*. Если потребуется, измените масштаб для более удобного просмотра созданной таблицы с помощью пиктограммы с изображением лупы или раскрывающегося списка масштабов. Закройте окно просмотра.

36. В окне STUDENT: *База данных* снова установите курсор на слове *Таблица1* и вызовите контекстное меню.

37. Выберите команду *Переименовать*. Введите новое имя таблицы: *T_Студенты*.

Создание запросов

38. В окне STUDENT: *База данных* выберите объект *Запросы*. Щелкните на кнопке *Создать*.

39. В появившемся окне *Новый запрос* выберите: *Конструктор*. Щелкните на кнопке *OK*.

40. В окне *Добавление таблицы* выберите объект *Таблицы T_Студенты*. Щелкните на кнопках *Добавить* и *Закрыть*.

41. Откроется окно *Конструктор запросов* с заголовком *Запрос1: запрос на выборку*. В его верхней части отображаются списки полей таблицы, к которым обращается запрос. Нижняя область содержит бланк выбора полей таблиц, условий отбора и режимов сортировки. Указывается также название таблицы, которой принадлежит выбранное поле.

42. Поочередно щелкайте два раза левой кнопкой мыши на полях: *Код, Фамилия, Имя, Группа, Дата рождения, Домашний адрес*.

43. Установите курсор в нижней части окна в столбце *Фамилия* в поле *Сортировка*. В раскрывающемся списке этого поля установите: *По возрастанию*.

44. Щелкните на значке закрытия текущего окна *Запрос1: запрос на выборку*. Подтвердите сохранение структуры запроса.

45. В наборном поле *Имя запроса* в окне *Сохранение* введите имя запроса: *Список всех студентов*. Щелкните на кнопке *OK*.

46. Откройте и просмотрите запрос *Список всех студентов*. Обратите внимание: в записях фамилии расположены в алфавитном порядке.

47. Установите курсор на запросе *Список всех студентов*.

48. Нажмите клавишу <*Ctrl*> и, ухватившись за значок запроса, переместите его в сторону и отпустите кнопку мыши. Возникнет новый ярлык: *Копия Список всех студентов*.

49. Установите курсор на новый ярлык и переименуйте его в *Список студентов 271 группы*.

50. Откройте запрос *Список студентов 271 группы*.

51. В меню *Вид* выберите режим работы: *Конструктор*.

52. В столбце *Группа* в строке *Условие отбора* введите: 271.

53. Щелкните на значке закрытия текущего окна, подтвердите сохранение макета *Список студентов 271 группы: запрос на выборку*.

54. Откройте запрос *Список студентов 271 группы*. Просмотрите содержимое запроса. Закройте текущее окно.

55. В исходной таблице в одной из записей измените номер группы на номер *271*. Закройте таблицу.

56. Вновь откройте запрос *Список студентов 271 группы*. Убедитесь, что содержимое запроса изменилось.

Создание формы с помощью *Мастера форм*

57. В окне STUDENT: *База данных* выберите объект *Формы,* щелкните на кнопке *Создать*.

58. В окне *Новая форма* выберите: *Мастер форм,* а в качестве источника данных с помощью кнопки раскрывающегося списка выберите таблицу *T_Студенты*. Щелкните на кнопке *OK*.

59. *Мастер форм* позволяет сберечь время и быстро сконструировать привлекательную форму для записей любой таблицы.

60. В первом окне *Мастера форм,* показанном на рис. 4.26, в списке *Таблицы и запросы* указана выбранная таблица *T_Студенты*.

61. Щелкните на кнопке >, чтобы в список *Выбранные поля* добавить: *все поля таблицы*.

62. Выделите пункт *Код* и щелчком на кнопке < уберите это поле обратно в левый список. Содержимое этого поля генерируется автоматически, а его значение несущественно для пользователя, поэтому не следует включать его в форму.

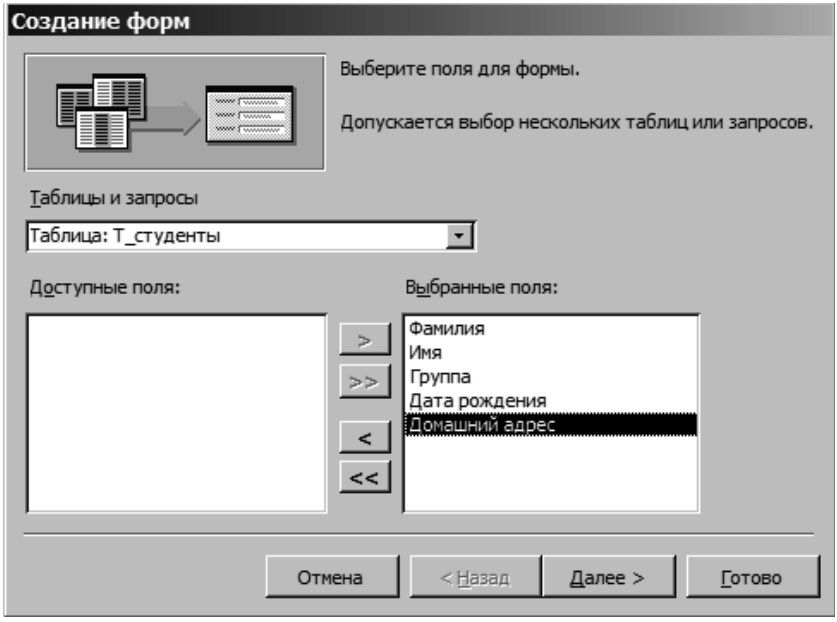


Рис. 4.26. Мастер форм

63. Щелкните на кнопке *Далее*.

64. В следующем окне диалога выберите для формы стиль: *В один столбец* и щелкните на кнопке *Далее*.

65. В списке третьего окна выберите понравившийся вам стиль оформления и снова щелкните на кнопке *Далее*.

66. В последнем окне *Мастера форм* щелкните на кнопке *Готово,* не изменяя никаких параметров. *Мастер форм* сгенерирует форму и откроет ее в режиме просмотра данных.

67. Окно формы содержит названия полей и области отображения данных исходной таблицы. В нижней части формы расположены кнопки перемещения по записям.

68. Щелкните несколько раз на кнопке *Следующая запись,* чтобы добраться до пустой строки, и введите запись еще об одном человеке.

Создание формы с помощью режима *Конструктор форм*

Недостатком форм, создаваемых *Мастером,* является то, что они однообразны и не содержат пояснительных надписей, а также не имеют элементов управления. Чтобы приукрасить

форму, расположить поля более удобным способом, следует воспользоваться режимом *Конструктор форм,* который позволяет создавать новые формы и редактировать имеющиеся.

69. Выберите вкладку *Формы*. Установите режим *Конструктор форм* (команда меню *Вид → Конструктор*). В окне *Конструктор форм* появятся разметочная сетка, вертикальная и горизонтальная линейки, позволяющие позиционировать объекты. Изменение позиции объекта происходит с помощью обычных для Windows методов.

70. Одним щелчком выделите надпись *Фамилия,* установите курсор внутри объекта и измените надпись на *Фамилия студента.*

71. Щелкните на объекте *Фамилия студента* правой кнопкой мыши, в контекстном меню выберите команду *Свойства;* откроется окно свойств *Надпись: Фамилия_надпись* (рис. 4.27). Во вкладке *Макет* установите: цвет фона — голубой, размер шрифта — 12, оформление — *Приподнятое* и другие свойства по своему желанию. Если текст не будет помещаться в рамку, с помощью маркеров измените границы объекта.

72. Измените внешний вид других объектов формы.

Создание элементов управления

При открытии окна *Конструктор форм* на экране должна появиться *Панель элементов*. Если ее нет, щелкните на кнопке *Панель элементов*  панели инструментов. С помощью кнопок *Панели элементов* в форму можно добавлять различные объекты. Элементы управления форм и отчетов сходны между собой, поэтому такая же панель имеется в окне *Конструктор отчетов*.

73. Поместите указатель мыши на угол области формы.

74. Перетащите этот угол вправо вниз, чтобы увеличить форму.

75. С помощью команды *Правка → Выделить все* выделите все элементы формы.

76. Нажмите клавишу <*Ctrl*> и, удерживая ее, нажатием клавиш со стрелками переместите элементы формы вниз и вправо так, чтобы они были отцентрированы относительно новых границ формы.

77. Щелкните на кнопке *Надпись*  панели элементов.

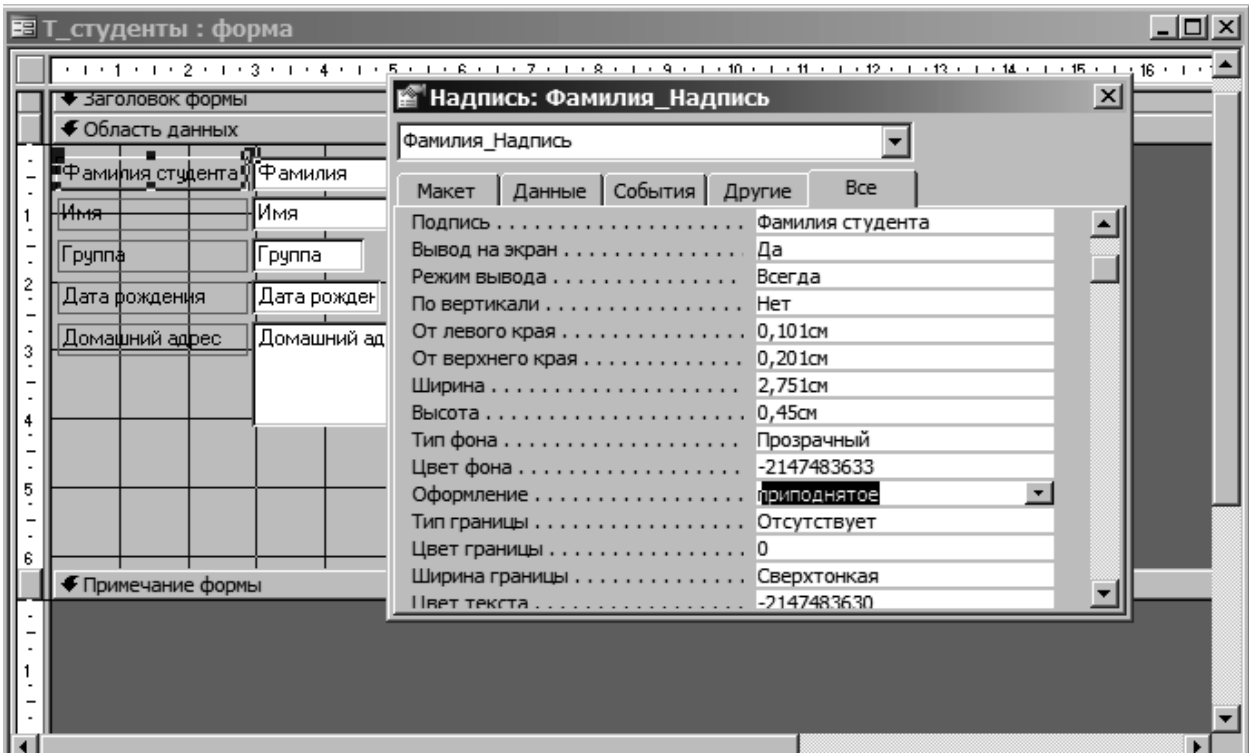


Рис. 4.27. Свойства объектов *Формы*

78. Растяните рамку надписи в верхней части формы на ширину области данных.

79. Введите надпись, которая будет являться заголовком формы: *Список студентов*.

80. Находясь в области заголовка, вызовите контекстное меню и выполните команду *Свойства*.

81. Во вкладке *Макет* установите следующие параметры: ширина границы — 3 пт, цвет фона — розовый, размер шрифта — 14, курсив — *Да,* расстояние от левого края страницы — 3 см.

82. Щелкните на кнопке *Рисунок*  панели элементов.

83. Внизу формы растяните рамку рисунка.

84. В открывшемся окне выбора файла найдите папку с рисунками Windows, выберите любой понравившийся вам рисунок и щелкните на кнопке *OK*. По умолчанию рисунки вставляются по соответствующей форме.

85. Чтобы изменить режим размещения, щелкните на рисунке правой кнопкой мыши и в контекстном меню выполните команду *Свойства*.

86. В списке *Установка размеров* открывшегося окна параметров выберите пункт *Вписать в рамку*.

87. Закройте окно параметров.

88. Щелкните на кнопке *Кнопка*  панели элементов.

89. Перенесите указатель мыши в область формы и щелкните внизу формы.

90. В открывшемся окне *Создание кнопок* (рис. 4.28) выберите категорию действия *Переходы по записям,* в качестве *Действия* выберите: *Следующая запись*. Щелкните на кнопке *Далее*.

91. Во втором окне *Мастера создания кнопок* выберите рисунок на кнопку, например: *Стрелка вправо (синяя)*. Щелкните на кнопке *Далее*.

92. В третьем окне *Мастера создания кнопок* выберите название кнопки: *Следующая запись*. Щелкните на кнопке *Готово*.

93. Создайте кнопки *Предыдущая запись, Найти запись* и *Выход из формы,* применяя действия, описанные в пунктах 82—86.

94. Установите режим работы с формой (команда меню *Вид → Режим формы*).

95. Проверьте действие кнопок.

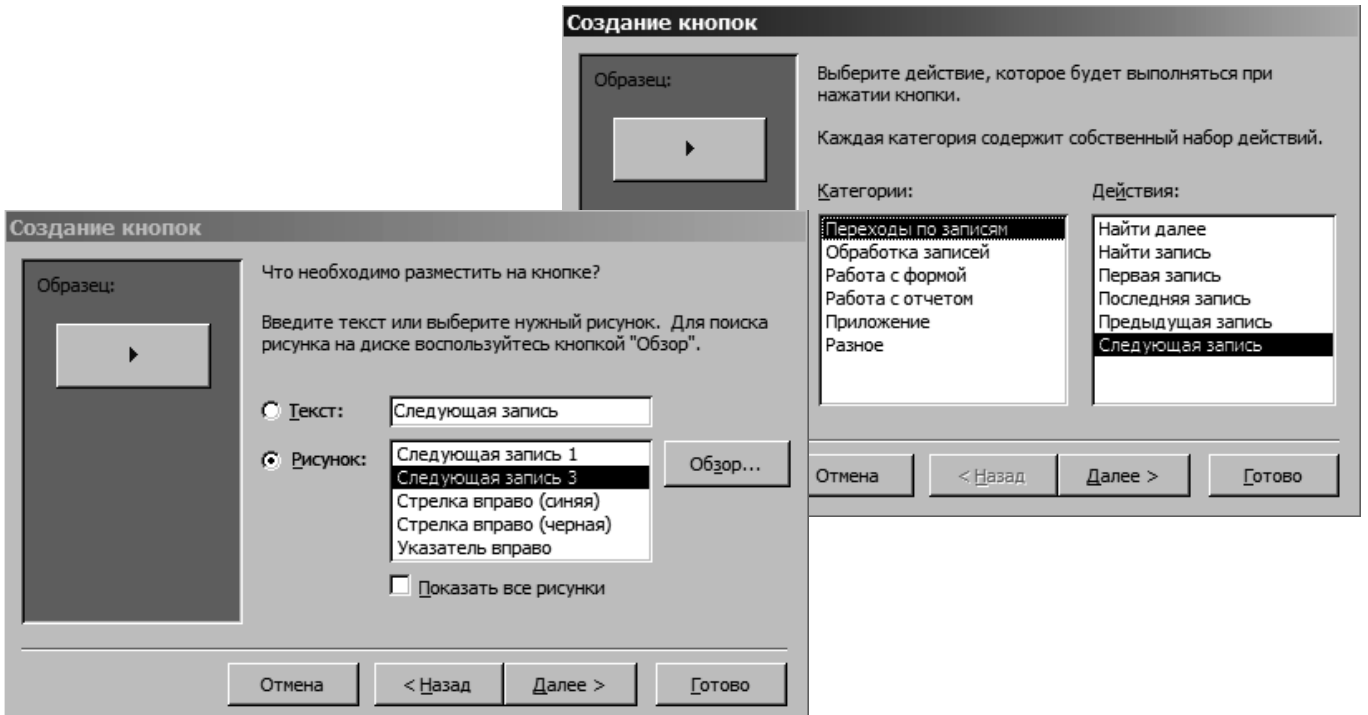


Рис. 4.28. Мастер создания кнопок

Создание отчетов

96. В окне STUDENT: *База данных* выберите объект *Отчеты* и щелкните на кнопке *Создать*.

97. В окне *Новый отчет* выберите режим *Конструктор,* а в качестве источника данных выберите таблицу *Т_Студенты*. Щелкните на кнопке *ОК*.

Окно *Конструктор отчетов* состоит из трех областей: *Верхний колонтитул, Нижний колонтитул* и *Область данных*. Появляется также небольшое окно со списком полей источника записей (таблицы *Т_Студенты*). Если на экране отсутствует панель элементов, выведите ее на экран щелчком на кнопке *Панель элементов* на панели инструментов.

98. На панели элементов щелкните на кнопке *Надпись*  .

99. Щелкните в области *Верхний колонтитул* и введите надпись: *Список студентов*.

100. В области надписи правой кнопкой мыши откройте контекстное меню и выполните команду *Свойства*.

101. В открывшемся окне параметров во вкладке *Макет* выберите размер шрифта: 16. Закройте окно параметров.

102. В области надписи снова откройте контекстное меню, выполните команды: *Размер* — по размеру данных, *Цвет текста* — розовый.

103. Аналогично *Верхнему колонтитулу* оформите в *Нижнем колонтитуле* надпись: *Инженерно-физический факультет*.

104. Щелкните на пиктограмме *Поле*  на панели элементов.

105. Щелкните в первой строке первого столбца *Области данных*. Access создаст элемент управления типа *Поле — Свободный,* а также *Поле№* — для надписи.

106. Щелкните на элементе *Поле№* и удалите этот элемент клавишей <*Delete*>.

107. На элементе *Свободный* откройте контекстное меню, выполните команду *Свойства*, на вкладке *Данные* в строке *Данные* введите: = 1.

108. В строке *Сумма с накоплением* с помощью кнопки выбора установите: *Для всего*. Закройте окно *Свойства*.

109. Поместите указатель мыши на пункт *Фамилия* окна *Т_Студенты*, выполните щелчок и перетащите этот пункт в об-

ласть формы. Access снова создаст элемент управления типа *Поле* справа, а также поле для надписи — слева. Удалите поле для надписи.

110. Проделайте аналогичную операцию для пунктов: *Имя, Группа, Дата рождения.*

111. Оформите новые элементы области данных разными шрифтами, фонами и т. д. по своему выбору.

112. Если поля в *Области данных* не умещаются, увеличьте *Область данных* до нужных размеров. Выравнивание полей по горизонтали или вертикали производится после их выделения левой кнопкой мыши при нажатой клавише *<Shift>* с помощью контекстного меню.

113. Закройте текущее окно *Конструктора*. Дайте название отчету: *О_Список студентов → ОК.*

114. В окне STUDENT: *База данных* выберите объект *Отчеты,* откройте отчет *О_Список студентов.*

115. Если это потребуется, отредактируйте отчет *О_Список студентов,* используя режим *Конструктор* так, чтобы список студентов занимал минимальный размер и все данные были внесены в отчет полностью.

116. Закройте отчет. Закройте базу данных.

Работа № 2. Организация связей между таблицами

Цель работы: разработка организации связей между таблицами в Access.

Задание: к созданной ранее базе данных требуется добавить еще две таблицы. Они должны содержать сведения о преподавателях и предметах, которые изучаются на первом и втором курсах. Один и тот же предмет могут вести несколько преподавателей. Организуем две таблицы, связанные между собой связью типа «один (предмет) ко многим (преподавателям)», чтобы использовать *Список подстановки* для автоматизации ввода данных.

Методика выполнения работы:

1. Откройте базу данных STUDENT.

2. В режиме *Конструктор* создайте в базе данных новую таблицу из двух столбцов: *Название предмета* (тип данных —

Текстовый, размер поля — 20) и *Семестр* (тип данных — *Числовой,* размер поля — один байт).

3. Закройте текущее окно, задав макету имя *Т_Предметы* и подтвердив создание ключевого поля. Access автоматически добавит к созданным вами двум полям еще одно — *Код* с типом данных *Счетчик,* определив это поле как ключевое.

4. Вновь откройте таблицу. Обратите внимание на созданное новое поле со значком  . Заполните таблицу, указав в ней названия трех предметов, изучаемых вами на первых двух курсах. Поле *Код* заполняется автоматически. Закройте таблицу *Т_Предметы*.

5. Создайте еще одну таблицу в режиме *Конструктор*. Назовите ее *Т_Преподаватели*.

6. Создайте поля: *Фамилия* (тип данных — *Текстовый,* размер поля — 20, описание — *Фамилия преподавателя*), *Имя* (тип данных — *Текстовый,* размер поля — 20), *Отчество* (тип данных — *Текстовый,* размер поля — 20), *Код предмета* (тип данных — *Числовой,* размер поля — *Длинное целое,* описание — *Предмет, который ведет преподаватель*).

7. Щелкните на значке закрытия текущего окна *Т_Преподаватели: таблица*. Откажитесь от создания ключевого поля.

8. Откройте таблицу для заполнения.

9. Занесите в таблицу имена, отчества и фамилии пяти преподавателей, ведущих предметы, перечисленные в таблице *Т_Предметы*. Разные преподаватели могут вести один и тот же предмет. В столбце *Код предмета* проставьте цифры, соответствующие кодам предметов из таблицы *Т_Предметы*.

10. Щелкните на значке закрытия текущего окна *Т_Преподаватели: таблица*.

11. Щелкните на кнопке *Схема данных* панели инструментов  .

12. В открывшемся окне *Добавление таблицы* во вкладке *Таблицы* установите курсор на *Т_Предметы* и щелкните на кнопке *Добавить*.

13. Установите курсор на *Т_Преподаватели* и снова щелкните на кнопке *Добавить*. Закройте окно *Добавление таблицы*.

14. В окне *Схема данных* вы видите две небольшие таблицы: *Т_Предметы* и *Т_Преподаватели*. Для создания связи между ними выполните следующие действия:

— поместите указатель мыши на пункт *Код* таблицы *Т_Предметы;*

— щелкните левой кнопкой мыши и перетащите указатель на поле *Код предмета* таблицы *Т_Преподаватели;*

— в открывшемся окне диалога установите флажок *Обеспечение целостности данных* и щелкните на кнопке *Создать*. Между двумя таблицами в окне *Схема данных* появится линия связи типа «один ко многим».

15. Закройте текущее окно, подтвердив сохранение.

Список подстановки

При заполнении таблицы *Т_Преподаватели* в поле *Код предмета* приходится заносить не название предмета, а его код. Это очень неудобно, так как список предметов может быть расширен и трудно будет удержать в памяти все коды. Access позволяет автоматизировать операцию ввода данных с помощью *Списка подстановки,* создание которого обеспечивается наличием связи между двумя таблицами.

16. Откройте таблицу *Т_Преподаватели* в режиме *Конструктор форм*.

17. В столбце *Тип данных* для поля *Код предмета* выберите: *Мастер подстановок*.

18. В первом окне *Мастера форм* оставьте выбранным положение переключателя *Объект «столбец подстановки» будет использовать значение из таблицы или запроса* и щелкните на кнопке *Далее*.

19. Три положения переключателя второго окна *Мастера форм* выводят на экран список таблиц, запросов или объединяют эти два списка. Выберите таблицу *Т_Предметы* на роль источника подстановки. Щелкните на кнопке *Далее*.

20. Третье окно *Мастера форм* (рис. 4.29) предлагает выбрать из таблицы *Т_Предметы* поля, участвующие в подстановке. Этот список должен обязательно включать в себя и то поле, содержание которого будет отображаться вместо численного значения, помещаемого из поля *Код*.

21. Щелкните на пункте *Название предмета*. Щелкните на кнопке *Далее*.

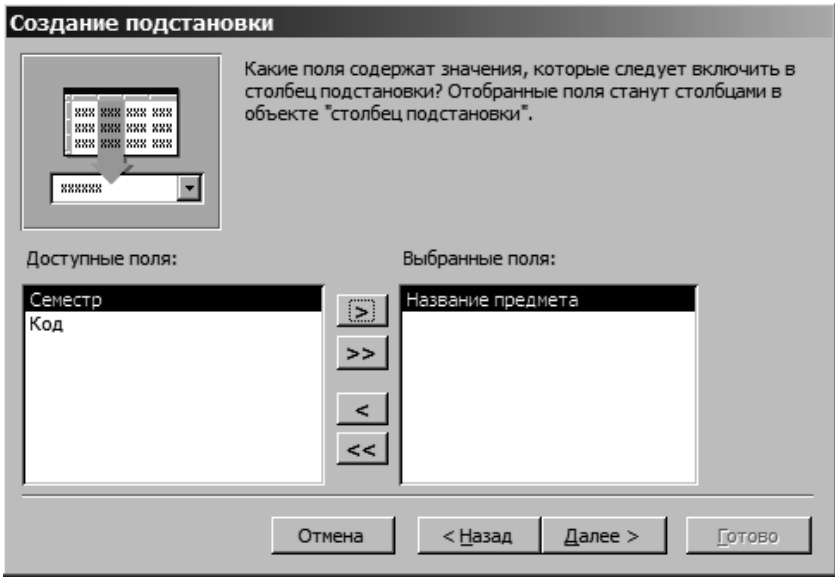


Рис. 4.29. Окно *Мастера* подстановок

22. В следующем окне *Мастера форм* будет продемонстрирован столбец таблицы-источника.

23. В поле последнего окна введите название: *Предмет,* которое заменит имя столбца *Код предмета,* и щелкните на кнопке *Готово*.

24. В появившемся окне диалога щелчком на кнопке *Да* подтвердите необходимость сохранения построенной структуры.

25. Находясь в окне *Т_Преподаватели: таблица,* перейдите в режим таблицы (меню *Вид* → *Режим таблицы*).

Теперь в четвертом столбце вместо чисел стоят названия предметов, соответствующие этим числам. Access автоматически ищет соответствующую запись в таблице *Т_Предметы* и выводит в ячейку таблицы *Т_Преподаватели* текст поля *Название предмета* из выбранного в списке *Мастер подстановок*.

26. Введите еще одну запись в таблицу *Т_Преподаватели*. При вводе данных в столбец *Предмет* используйте список *Мастера подстановок*.

27. Закройте текущее окно.

КОНТРОЛЬНЫЕ ВОПРОСЫ К ТЕМЕ 4.10

1. **Какую базу данных называют реляционной?**
2. **Из каких основных объектов состоит база данных?**
3. **Какую информацию содержит таблица, в которой нет ни одной записи?**
4. **Приведите примеры использования различных типов полей в таблицах.**
5. **Какое поле можно считать уникальным?**
6. **Какой параметр определяет длину поля?**
7. **Как запретить ввод пустых полей?**
8. **Поле какого типа является ключевым в большинстве таблиц?**
9. **Назовите три основных свойства запросов, используемых пользователями при работе с большими базами данных.**
10. **Какие операции закрывают базу данных?**
11. **Как с помощью *Мастера отчетов* сгруппировать записи по дате?**
12. **Как назначить сортировку в алфавитном порядке при создании отчетов по одному полю? по двум полям?**
13. **Для чего создаются межтабличные связи при объединении таблиц и создании схемы данных?**
14. **Какова роль флажков *Обеспечение целостности данных, Каскадное обновление связанных полей* и *Каскадное удаление связанных записей* в диалоговом окне *Связи?***

Раздел 5

МОДЕЛИРОВАНИЕ И ФОРМАЛИЗАЦИЯ

В результате изучения пятого раздела студент должен:

знать:

— этапы информационной технологии решения задач с использованием компьютера;

уметь:

— строить простейшие информационные модели;

— приводить примеры моделирования формализованного описания объектов и процессов.

иметь представление:

— о моделировании и формализации в информационных технологиях.

Методические указания

Моделирование (в широком смысле) — основной метод исследований во всех областях знания и научно обоснованный метод оценок характеристик систем, используемый для принятия решений в различных сферах деятельности. Задача преподавателя — объяснить важность моделирования как мощного метода познания мира. Студенты, со своей стороны, должны усвоить, что существующие и проектируемые системы можно эффективно исследовать с помощью математических моделей (аналитических и имитационных), реализуемых на современных ЭВМ, которые в этом случае выступают в качестве инструмента экспериментатора для работы с моделью системы.

Тема 5.1

МОДЕЛИРОВАНИЕ КАК МЕТОД ПОЗНАНИЯ. МАТЕРИАЛЬНЫЕ И ИНФОРМАЦИОННЫЕ МОДЕЛИ

5.1.1. Моделирование. Формальная и неформальная постановка задачи

В настоящее время нельзя назвать область человеческой деятельности, в которой в той или иной степени не использовались бы методы моделирования. Особенно это относится к сфере управления различными системами, где основными являются процессы принятия решений на основе получаемой информации. Рассмотрим основные положения теории моделирования.

Методологическая основа моделирования. Все то, на что направлена человеческая деятельность, называется **объектом** (от лат. objectum — предмет). Выработка методологии направлена на упорядочение получения и обработки информации об объектах, которые существуют вне нашего сознания и взаимодействуют между собой и внешней средой.

В научных и прикладных исследованиях большую роль играют *гипотезы,* т. е. определенные предсказания, основывающиеся на небольшом количестве опытных данных, наблюдений, догадок. Быстрая и полная проверка выдвигаемых гипотез может быть проведена в ходе специально поставленного эксперимента. В качестве метода суждения при формулировании и проверке правильности гипотез большое значение имеет аналогия.

Аналогией называется суждение о каком-либо частном сходстве двух объектов, причем такое сходство может быть существенным и несущественным. Необходимо отметить, что понятия существенности и несущественности сходства или различия объектов условны и относительны. Существенность сходства (различия) зависит от уровня абстрагирования и в общем случае определяется конечной целью проводимого исследования. Современная научная гипотеза создается, как правило, по аналогии с проверенными на практике научными положениями. Таким образом, аналогия связывает гипотезу с экспериментом.

Гипотезы и аналогии, отражающие реальный, объективно существующий мир, должны обладать наглядностью или сводиться к удобным для исследования логическим схемам. Логические схемы, упрощающие рассуждения и логические построения или позволяющие проводить эксперименты, уточняющие природу явления, называются **моделями**. Другими словами, модель (от лат. modulus — мера) — это объект-заместитель объекта-оригинала, обеспечивающий изучение некоторых свойств оригинала.

Определение моделирования. Замещение одного объекта другим с целью получения информации о важнейших свойствах объекта-оригинала с помощью объекта-модели называется **моделированием**. Таким образом, моделирование может быть определено как представление объекта моделью для получения информации об этом объекте путем проведения экспериментов с его моделью. Теория замещения одних объектов (оригиналов) другими объектами (моделями) и исследования свойств объектов на их моделях называется **теорией моделирования**.

Определяя гносеологическую роль теории моделирования, т. е. ее значение в процессе познания, необходимо прежде всего отвлечься от имеющегося в науке и технике многообразия моделей и выделить то общее, что присуще моделям различных по своей природе объектов реального мира. Это общее заключается в наличии некоторой структуры (статической или динамической, материальной или мысленной), подобной структуре данного объекта. В процессе изучения модель выступает в роли относительно самостоятельного квазиобъекта, позволяющего получить при исследовании некоторые знания о самом объекте.

Если результаты моделирования подтверждаются и могут служить основой для прогнозирования процессов, протекающих в исследуемых объектах, то говорят, что модель адекватна объекту. При этом *адекватность* модели зависит от цели моделирования и принятых критериев.

Обобщенно моделирование можно определить как метод опосредованного познания, так как изучаемый объект (оригинал) находится в некотором соответствии с другим объектом (моделью), причем модель способна в том или ином отноше-

нии замещать оригинал на некоторых стадиях познавательного процесса. Стадии познания, на которых происходит такая замена, а также формы соответствия модели и оригинала могут быть различными:

— моделирование как познавательный процесс, содержащий переработку информации, поступающей из внешней среды, о происходящих в ней явлениях. В результате в сознании появляются образы, соответствующие объектам;

— моделирование, заключающееся в построении некоторой системы-модели (второй системы), связанной определенными соотношениями подобия с системой-оригиналом (первой системой), причем в этом случае отображение одной системы в другую является средством выявления зависимостей между двумя системами, отраженных в соотношениях подобия, а не результатом непосредственного изучения поступающей информации.

Следует отметить, что с точки зрения философии моделирование — эффективное средство познания природы. Процесс моделирования предполагает наличие:

— объекта исследования;

— исследователя, перед которым поставлена конкретная задача;

— модели, создаваемой для получения информации об объекте и необходимой для решения поставленной задачи.

Причем по отношению к модели исследователь является, по сути дела, экспериментатором, только в данном случае эксперимент проводится не с реальным объектом, а с его моделью. Такой эксперимент для исследователя есть инструмент непосредственного решения организационно-технических задач.

Надо иметь в виду, что любой эксперимент может иметь существенное значение в конкретной области только при специальной его обработке и обобщении. Единичный эксперимент никогда не может быть решающим для подтверждения гипотезы, проверки теории. Поэтому исследователи и практики должны быть знакомы с элементами современной методологии теории познания, и в частности не должны забывать основного положения материалистической философии: именно экспериментальное исследование, опыт, практика являются критерием истины.

5.1.2. Основные принципы формализации при моделировании

В последние годы основные достижения в различных областях науки и техники неразрывно связаны с процессом совершенствования ЭВМ. Сфера эксплуатации ЭВМ — бурно развивающаяся отрасль человеческой практики, стимулирующая развитие новых теоретических и прикладных направлений. Ресурсы современной информационно-вычислительной техники дают возможность ставить и решать математические задачи такой сложности, которые в недавнем прошлом казались нереализуемыми, например моделирование больших систем.

Аналитические и имитационные методы. Исторически первым сложился *аналитический подход* к исследованию систем, когда ЭВМ использовалась в качестве вычислителя по аналитическим зависимостям. Анализ характеристик процессов функционирования систем с помощью только аналитических методов исследования наталкивается обычно на значительные трудности, приводящие к необходимости существенного упрощения моделей либо на этапе их построения, либо в процессе работы с моделью, что может привести к получению недостоверных результатов.

Поэтому в настоящее время наряду с построением аналитических моделей большое внимание уделяется задачам оценки характеристик систем на основе *имитационных моделей,* реализованных на современных ЭВМ с высоким быстродействием и большим объемом оперативной памяти. Причем перспективность имитационного моделирования как метода исследования характеристик процесса функционирования систем возрастает с повышением быстродействия и оперативной памяти ЭВМ, развитием математического обеспечения, совершенствованием банков данных и периферийных устройств для организации диалоговых систем моделирования. Это способствует появлению новых — чисто машинных — методов решения задач исследования систем на основе организации имитационных экспериментов с их моделями. Причем ориентация на автоматизированные рабочие места (АРМ) на базе персональных ЭВМ (ПЭВМ) для реализации экспериментов с имитационными моделями систем позволяет не только проводить анализ

их характеристик, но и решать задачи структурного, алгоритмического и параметрического синтеза таких систем при заданных критериях оценки эффективности и ограничениях.

Достигнутые успехи в использовании средств вычислительной техники для целей моделирования часто создают иллюзию, что применение современной ЭВМ гарантирует возможность исследования системы любой сложности. При этом игнорируется тот факт, что в основу любой модели положено трудоемкое по затратам времени и материальных ресурсов предварительное изучение явлений, имеющих место в объекте-оригинале. И от того, насколько детально изучены реальные явления, насколько правильно проведена их формализация и алгоритмизация, зависит в конечном итоге успех моделирования конкретного объекта.

Средства моделирования систем. Расширение возможностей моделирования различных классов систем неразрывно связано с совершенствованием средств вычислительной техники и техники связи. Перспективным направлением является создание для целей моделирования иерархических многомашинных вычислительных систем, включающих центральный вычислительный комплекс из больших ЭВМ и множество связанных с ним каналами связи удаленных локальных вычислительных сетей и ПЭВМ, работающих в режиме телеобработки.

При создании систем их компоненты разрабатываются различными коллективами, которые применяют средства моделирования при анализе и синтезе отдельных подсистем. При этом разработчикам необходимы оперативный доступ к программно-техническим средствам моделирования и оперативный обмен результатами моделирования отдельных взаимодействующих подсистем и элементов. Таким образом, появляется необходимость в создании диалоговых систем моделирования коллективного пользования, которые представляют собой сложные человеко-машинные комплексы.

Цели моделирования систем. Один из наиболее важных аспектов построения систем моделирования — проблема **цели**. Любую модель строят в зависимости от цели, которую ставит перед ней исследователь. Отсюда одна из основных проблем моделирования — это проблема целевого назначения. Подобие процесса, протекающего в модели *M,* реальному процессу яв-

ляется не целью, а условием правильного функционирования модели, и поэтому в качестве цели должна быть поставлена задача изучения какой-либо стороны функционирования объекта.

Для упрощения модели *M* цели делят на подцели и создают более эффективные виды моделей в зависимости от полученных подцелей моделирования. Можно указать целый ряд примеров целей моделирования в области автоматизированных систем обработки информации и управления (АСОИУ). Например, для отраслевых АСУ наиболее существенными целями являются задачи прогноза, потребления и сбыта продукции, размещения предприятий отрасли с учетом всевозможных факторов (наличие сырья, людских ресурсов, энергии и т. д.). Для АСУ предприятием весьма существенно изучение процессов оперативного управления производством, оперативно-календарного и перспективного планирования; здесь также могут быть успешно использованы методы моделирования. Если цель моделирования ясна, то возникает следующая проблема, а именно, проблема построения модели *M*. Построение модели оказывается возможным, если имеется информация или выдвинуты гипотезы относительно структуры, алгоритмов и параметров исследуемого объекта. На основании их изучения осуществляется идентификация объекта. *Идентификация* (от англ. identification) — это процесс отождествления объекта с одним из известных системе объектов.

В зависимости от цели исследования могут рассматриваться разные соотношения между самим объектом *S* и внешней средой *E*. Следовательно, в зависимости от уровня, на котором находится наблюдатель, объект исследования может выделяться по-разному и могут иметь место различные взаимодействия этого объекта с внешней средой.

Таким образом, применительно к вопросам моделирования цель возникает из требуемых задач моделирования, что позволяет подойти к выбору критерия и оценить, какие элементы войдут в создаваемую модель *M*. Поэтому необходимо иметь критерий отбора отдельных элементов в создаваемую модель.

Подходы к исследованию систем. Важным для системного подхода является определение **структуры системы** — совокупности связей между элементами системы, отражающих их взаимодействие. Структура системы может изучаться *извне* — с точ-

ки зрения состава отдельных подсистем и отношений между ними, а также *изнутри* — когда анализируются отдельные свойства, позволяющие системе достигать заданной цели, т. е. когда изучаются функции системы. В соответствии с этим наметился ряд подходов к исследованию структуры системы с ее свойствами, среди которых прежде всего следует отметить структурный и функциональный.

При *структурном подходе* выявляются состав выделенных элементов системы S и связи между ними. Совокупность элементов и связей между ними позволяет судить о структуре системы. Последняя в зависимости от цели исследования может быть описана на разных уровнях рассмотрения. Наиболее общее описание структуры — это топологическое описание, позволяющее определить в самых общих понятиях составные части системы и хорошо формализуемое на базе теории графов. *Топология* (от греч. typos — место, logos — понятие, учение) — это раздел математики, в котором изучаются наиболее общие свойства геометрических фигур, не изменяющихся при любых преобразованиях. *Теория графов* — это раздел математики, особенностью которого является геометрический подход к изучению объектов любой природы.

Менее общим является функциональное описание, когда рассматриваются отдельные функции, т. е. алгоритмы поведения, системы и реализуется *функциональный подход,* оценивающий функции, которые выполняет система. Причем под *функцией* понимается свойство, приводящее к достижению цели. Поскольку функция отображает свойство, а свойство отображает взаимодействие системы S с внешней средой E, то свойства могут быть выражены либо в виде некоторых характеристик элементов $S_{i(j)}$ и подсистем S_i системы, либо в виде системы S в целом.

При наличии некоторого эталона сравнения можно ввести количественные и качественные характеристики систем. Для количественной характеристики вводятся числа, выражающие отношения между данной характеристикой и эталоном. Качественные характеристики системы находятся, например, с помощью метода экспертных оценок.

Проявление функций системы во времени $S(t)$, т. е. *функционирование системы,* означает переход системы из одного со-

стояния в другое, т. е. движение в пространстве состояний Z. При эксплуатации системы S весьма важно качество ее функционирования, определяемое показателем эффективности и являющееся значением критерия оценки эффективности. Существуют различные подходы к выбору критериев оценки эффективности. Система S может оцениваться либо совокупностью частных критериев, либо некоторым общим, интегральным критерием.

Следует отметить, что создаваемая модель M с точки зрения системного подхода также является системой, т. е. $S' = S'(M)$, и может рассматриваться по отношению к внешней среде E. Наиболее простыми по представлению являются модели, в которых сохраняется прямая аналогия явления. Применяются также модели, в которых нет прямой аналогии, а сохраняются лишь общие закономерности поведения элементов системы S. Правильное понимание как взаимосвязей внутри самой модели M, так и ее взаимодействия с внешней средой E в значительной степени определяется тем, на каком уровне находится наблюдатель.

Экспериментальные исследования систем. Одновременно с развитием теоретических методов анализа и синтеза совершенствуются и методы экспериментального изучения реальных объектов, появляются новые средства исследования. Однако эксперимент был и остается одним из основных и существенных инструментов познания. Подобие и моделирование позволяют по-новому описать реальный процесс и упростить его экспериментальное изучение. Совершенствуется и само понятие моделирования. Если раньше оно означало реальный физический эксперимент либо построение макета, имитирующего реальный процесс, то в настоящее время появились новые виды моделирования, в основе которых лежит постановка не только физических, но и математических экспериментов.

Моделирование базируется на некоторой аналогии реального и мысленного экспериментов. Аналогия — основа для объяснения изучаемого явления, однако критерием истины может служить только практика, только опыт. Хотя современные научные гипотезы могут создаваться чисто теоретическим путем, но, по сути, базируются они на широких практических знаниях. Чтобы объяснить реальные процессы, выдвигаются гипоте-

зы, для подтверждения которых ставится эксперимент либо проводятся такие теоретические рассуждения, которые логически подтверждают их правильность. В широком смысле под *экспериментом* можно понимать некоторую процедуру организации и наблюдения каких-то явлений, которые осуществляют в условиях, близких к естественным, либо имитируют их.

Различают *пассивный эксперимент,* когда исследователь наблюдает протекающий процесс, и *активный,* когда наблюдатель вмешивается в процесс и организует его протекание. В последнее время распространен активный эксперимент, поскольку именно на его основе удается выявить критические ситуации, получить наиболее интересные закономерности, обеспечить возможность повторения эксперимента в различных точках и т. д.

В основе любого вида моделирования лежит некоторая модель, имеющая соответствие, базирующееся на некотором общем качестве, характеризующем реальный объект. Объективно реальный объект обладает некоторой формальной структурой, поэтому для любой модели характерно наличие некоторой структуры, соответствующей формальной структуре реального объекта либо изучаемой стороне этого объекта.

В основе моделирования лежат информационные процессы, поскольку само создание модели M базируется на информации о реальном объекте. Получают такую информацию в ходе реализации модели, одновременно в процессе эксперимента с моделью вводят управляющую информацию; существенное место занимает обработка полученных результатов, т. е. информация лежит в основе всего процесса моделирования.

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

Под ***математическим моделированием*** будем понимать процесс установления соответствия данному реальному объекту некоторого математического объекта, называемого **математической моделью**, и исследование этой модели, позволяющее получать характеристики рассматриваемого реального объекта. Вид математической модели зависит как от природы реального объекта, так и от задач исследования объекта и требуе-

мой достоверности и точности решения этой задачи. Любая математическая модель, как и всякая другая, описывает реальный объект лишь с некоторой степенью приближения к действительности. Математическое моделирование для исследования характеристик процесса функционирования систем можно разделить на аналитическое, имитационное и комбинированное.

Для ***аналитического моделирования*** характерно то, что процессы функционирования элементов системы записываются в виде некоторых функциональных соотношений (алгебраических, интегро-дифференциальных, конечностно-разностных и т. п.) или логических условий. Аналитическая модель может быть исследована следующими методами:

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

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

— качественным, когда, не имея решения в явном виде, можно найти некоторые свойства решения (например, оценить его устойчивость).

Наиболее полное исследование процесса функционирования системы можно провести, если известны явные зависимости, связывающие искомые характеристики с начальными условиями, параметрами и переменными системы *S*. Однако такие зависимости удается получить только для сравнительно простых систем. При усложнении систем исследование их аналитическим методом наталкивается на значительные, часто непреодолимые, трудности. Поэтому при использовании аналитического метода приходится идти на существенное упрощение первоначальной модели, чтобы иметь возможность изучить хотя бы общие свойства системы. Такое исследование на упрощенной модели аналитическим методом помогает получить ориентировочные результаты для определения более точных оценок другими методами.

Численный метод, по сравнению с аналитическим, позволяет исследовать более широкий класс систем, но при этом полученные решения носят частный характер. Численный метод особенно эффективен при использовании ЭВМ.

В отдельных случаях исследователя системы могут удовлетворить и те выводы, которые можно сделать при применении качественного метода анализа математической модели. Такие качественные методы широко используются, например, в теории автоматического управления для оценки эффективности различных вариантов систем управления.

В настоящее время распространены методы машинной реализации исследования характеристик процесса функционирования больших систем. Для реализации математической модели на ЭВМ необходимо построить соответствующий моделирующий алгоритм.

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

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

Другие виды моделирования. При ***реальном моделировании*** используется возможность исследования различных характеристик либо на реальном объекте целиком, либо на его части. Такие исследования могут проводиться как на объектах, работающих в нормальных режимах, так и при организации специальных режимов для оценки интересующих исследователя характеристик (при других значениях переменных и параметров, в другом масштабе времени и т. д.). Реальное моделиро-

вание является наиболее адекватным, но его возможности — с учетом особенностей реальных объектов — ограничены. Например, проведение реального моделирования АСУ предприятием потребует, во-первых, создания такой АСУ, а во-вторых — проведения экспериментов с управляемым объектом, т. е. предприятием, что в большинстве случаев невозможно. Рассмотрим разновидности реального моделирования.

Натурное моделирование — это проведение исследования на реальном объекте с последующей обработкой результатов эксперимента на основе теории подобия. При функционировании объекта в соответствии с поставленной целью удается выявить закономерности протекания реального процесса. Надо отметить, что такие разновидности натурного эксперимента, как *производственный эксперимент* и *комплексные испытания,* обладают высокой степенью достоверности.

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

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

5.1.3. Математические схемы моделирования систем

Наибольшие затруднения и наиболее серьезные ошибки при моделировании возникают при переходе от содержательного к формальному описанию объектов исследования, что объясняется участием в этом творческом процессе и специалистов в области систем, которые требуется моделировать (заказчиков), и специалистов в области машинного моделирования (исполнителей). Эффективным средством, обеспечиваю-

щим взаимопонимание между этими группами специалистов, является язык математических схем, позволяющий во главу угла поставить вопрос об адекватности перехода от содержательного описания системы к ее математической схеме и лишь затем решать вопрос о конкретном методе получения результатов с использованием ЭВМ: аналитическом или имитационном, а возможно, и комбинированном, т. е. аналитико-имитационном. Что касается конкретного объекта, то разработчику модели должны помочь тоже конкретные, уже прошедшие апробацию для данного класса систем математические схемы, показавшие свою эффективность в прикладных исследованиях на ЭВМ и получившие название **типовых математических схем**.

Исходной информацией при построении математических моделей процессов функционирования систем служат данные о назначении и условиях работы исследуемой (проектируемой) системы *S*. Эта информация определяет основную цель моделирования системы *S* и позволяет сформулировать требования к разрабатываемой математической модели *M*. Причем уровень абстрагирования зависит от круга тех вопросов, на которые исследователь системы хочет получить ответ с помощью модели, и в какой-то степени определяет выбор математической схемы.

Введение понятия «математическая схема» позволяет рассматривать математику не как метод расчета, а как метод мышления, средство формулирования понятий, что является наиболее важным при переходе от словесного описания системы к формальному представлению процесса ее функционирования в виде некоторой математической модели (аналитической или имитационной). При использовании математической схемы исследователя системы *S* в первую очередь должен интересовать вопрос об адекватности отображения в исследуемой системе реальных процессов в виде конкретных схем, а не возможность получения ответа (результата решения) на конкретный вопрос исследования.

Математическую схему можно определить как звено при переходе от содержательного к формальному описанию процесса функционирования системы с учетом воздействия внешней среды, т. е. имеет место цепочка: описательная модель — математическая схема — математическая (аналитическая или/и имитационная) модель.

Каждая конкретная система S характеризуется набором свойств, под которыми понимаются величины, отражающие поведение моделируемого объекта (реальной системы) и учитывающие условия ее функционирования во взаимодействии с внешней средой (системой) E. При построении математической модели системы необходимо решить вопрос о ее полноте. Полнота модели регулируется в основном выбором границы «система S — среда E». Также должна быть решена задача упрощения модели, которая помогает выделить основные свойства системы, отбросив второстепенные. Причем отнесение свойств системы к основным или второстепенным существенно зависит от цели моделирования системы (например, анализ вероятностно-временных характеристик процесса функционирования системы, синтез ее структуры и т. д.).

Формальная модель объекта. Формализм — это направление (схема), в котором высказывания располагаются и связываются друг с другом по чисто формальному признаку, чтобы затем перейти к логическому пониманию. В формализме значительно большее внимание уделяется понятийно-рациональному аспекту задачи, в противоположность конкретной содержательной стороне сущности. Формализм получил название от латинского слова forma, означающего прежде всего «внешнее очертание, наружный вид объекта, внешнее выражение какого-либо содержания». Модель объекта моделирования, т. е. системы S, можно представить в виде множества величин, описывающих процесс функционирования реальной системы и образующих в общем случае следующие подмножества:

— совокупность *входных воздействий* на систему:

$$x_i \in \vec{X},\ i = \overline{1,\ n_{\mathrm{X}}};$$

— совокупность *воздействий внешней среды:*

$$v_l \in \vec{V},\ l = \overline{1,\ n_{\mathrm{V}}};$$

совокупность *внутренних (собственных) параметров* системы:

$$h_k \in \vec{H},\ k = \overline{1,\ n_{\mathrm{H}}};$$

совокупность *выходных характеристик* системы:

$$y_j \in \vec{Y},\ j = \overline{1,\ n_{\mathrm{Y}}}.$$

При этом в перечисленных подмножествах можно выделить управляемые и неуправляемые переменные. В общем случае $\mathbf{x}_i$, $\mathbf{v}_l$, $\mathbf{h}_k$, $\mathbf{y}_j$ являются элементами непересекающихся подмножеств и содержат как детерминированные, так и вероятностные составляющие.

При моделировании системы S входные воздействия, воздействия внешней среды E и внутренние параметры системы являются независимыми переменными, которые в векторной форме соответственно имеют вид: $\vec{X}(t) = \mathbf{x}_1(t), \mathbf{x}_2(t), ..., \mathbf{x}_{n\mathrm{X}}(t)$; $\vec{V}(t) = \mathbf{v}_1(t), \mathbf{v}_2(t), ..., \mathbf{v}_{n\mathrm{V}}(t)$; $\vec{H}(t) = \mathbf{h}_1(t), \mathbf{h}_2(t), ..., \mathbf{h}_{n\mathrm{H}}(t)$, а выходные характеристики системы являются *зависимыми переменными* и в векторной форме имеют вид: $\vec{Y}(t) = \mathbf{y}_1(t), \mathbf{y}_2(t), ..., \mathbf{y}_{n\mathrm{Y}}(t)$.

Процесс функционирования системы S описывается во времени оператором F_s, который в общем случае преобразует переменные в соответствии с соотношениями вида

$$\vec{Y}(t) = F_s(\vec{X},\ \vec{V},\ \vec{H},\ t).$$

Тема 5.2

ОСНОВНЫЕ ТИПЫ ИНФОРМАЦИОННЫХ МОДЕЛЕЙ

5.2.1. Понятие об информационной технологии решения задач

Технология информационных процессов в любом типе ЭВМ имеет сходную структуру и состоит из операций и этапов. *Операция* — это совокупность выполняемых на одном рабочем месте элементарных действий, которая приводит к реализации определенной обработки данных. Под операцией понимается любой процесс, связанный с обработкой данных. *Этап* — это совокупность взаимосвязанных операций, которые реализуют определенную законченную функцию обработки данных.

В технологии информационных процессов выделяют следующие этапы: первичный, предварительный, основной и заключительный.

Первичный этап состоит из сбора, регистрации и передачи информации на обработку.

На ***предварительном этапе*** осуществляются прием, визуальный контроль данных, регистрация, кодирование, комплектование, перенос на машинный носитель, заполнение и формирование первичного документа.

При *визуальном контроле* проверяются четкость заполнения, отсутствие пропусков реквизитов, исправление ошибок.

Для сокращения объема вводимой в ЭВМ информации применяется *операция кодирования,* т. е. присвоения кодов одному или нескольким реквизитам. Обычно кодируются наименования, для чего разработаны специальные справочники и классификаторы.

Комплектование данных — операция вынужденная. При вводе больших объемов данных их разбивают на комплекты (пачки). Каждой пачке присваивается свой номер, который также вводится в ЭВМ. Комплектование облегчает поиск информации и исправление ошибок, обеспечивает контроль полноты вводимых данных, позволяет прерывать процесс ввода или подготовки данных на машинном носителе.

Операция *переноса на машинный носитель* выполнялась на больших ЭВМ. Основными носителями были перфокарты, перфоленты и магнитные ленты. В настоящее время эта операция часто совмещается с непосредственным вводом информации в ЭВМ с клавиатуры и со специальных устройств, считывающих данные с документов и штрих-кодов, а также с получением данных по сети или по запросу из БД.

Первичный и предварительный этапы технологии информационных процессов были выделены при обработке данных на больших ЭВМ, так как они выполнялись на разных рабочих местах в условиях пооперационной технологии. При обработке информации на ПК эти два этапа чаще всего объединяются в один *домашинный этап,* на котором все операции практически выполняются вручную.

Основной этап технологии информационных процессов содержит следующие операции: ввод данных в ЭВМ, контроль

безопасности данных и систем, сортировка, корректировка, группировка, анализ, расчет, формирование отчета и вывод данных. Так как все эти операции выполняются ПК, то этот этап называется **внутримашинным**.

Операция *ввода данных* — одна из основных и сложных операций технологии информационных процессов. Данные могут быть представлены в виде бумажного документа, в образе электронного документа, электронной таблицы, штрих-кодов; могут быть запрошены из БД, получены по сети, вводиться с клавиатуры, а в перспективе будет осуществляться речевой ввод. Поэтому ввод данных в ЭВМ обязательно сопровождается операцией контроля, так как неверные данные нет смысла обрабатывать.

Сами данные могут быть любого типа: текстовые, табличные, физические, в виде знаний, объектов реального мира и т. д. При этом подсистемы информационных систем (ИС) обычно имеют дело с разнородными данными, приходящими из различных источников. После ввода и контроля данные могут быть записаны в файл, показаны на дисплее, переданы в БД в режиме ее актуализации, переданы по сети. Чаще всего данные записываются в файл или БД.

Контроль безопасности данных и систем подразделяют на контроль достоверности данных, безопасности данных и компьютерных систем. *Контроль достоверности данных* выполняется программно во время их ввода и обработки. *Средства безопасности данных и программ* защищают их от копирования, искажения и несанкционированного доступа. *Средства безопасности компьютерных систем* обеспечивают защиту от кражи, вирусов, неправильной работы пользователей, несанкционированного доступа.

Сортировка данных используется для упорядочения записей файла по одному или нескольким ключам. *Запись* — это минимальная единица обмена между программой и внешней памятью. *Файл* — это совокупность записей. Обычно одна запись содержит информацию одного документа или его законченной части. Структура записи и файла определяется пользователем при проектировании. *Ключ* — это реквизит или группа реквизитов, служащих для идентификации записей. Сортировка упрощает дальнейшую обработку данных. Она присутствует во всех файловых системах.

Корректировка — это операция актуализации файла или БД. При этом обычно выполняются операции просмотра, замены, удаления, добавления нового. Эти операции применяются к отдельным реквизитам записи, группе записей, файлу, БД. *Группировка* — это операция соединения записей, сходных по одному или нескольких ключам, в относительно самостоятельные новые объекты — группы.

Анализ — это операция, реализующая метод научного исследования, основанный на расчленении целого на составные части. Для проведения анализа используются экономико-математические и статистические методы, методы выявления тенденций, прогнозирования, моделирования, построения графиков и диаграмм, экспертные методы.

Расчет — это операция, позволяющая выполнить требуемые вычисления для получения результатов или промежуточных данных.

Операция *формирования отчетов* заключается в оформлении результатов расчета для вывода и передачи данных потребителю в привычном для него виде.

Последняя операция — *вывод* — служит для вывода результатов обработки данных на печать, в БД, файл, на дисплей, по сети ЭВМ.

Заключительный этап технологии информационного процесса включает такие операции, как визуальный контроль результатов, размножение и передача потребителю. Этот этап также называется **послемашинным**. При установке ПК на рабочем месте пользователя заключительный этап может содержать только операции контроля: четкость вывода, непротиворечивость результатов и т. д. Все остальные операции могут выполняться на машинном этапе. Так как существует система электронной подписи, а потребителем является сам пользователь, то результаты обработки данных либо передаются по сети, либо записываются в БД.

5.2.2. Этапы решения задач на компьютере

При решении любых задач на компьютере предполагается выполнение следующих действий: постановка задачи, построение модели, разработка алгоритма и программы, ее отладка, исполнение программы, анализ результатов.

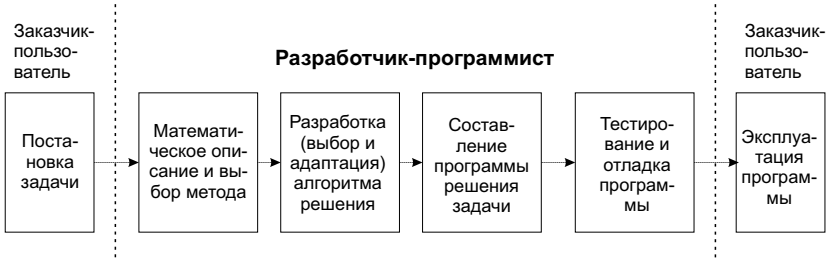


Рис. 5.1. Принципиальная схема технологического процесса разработки программных средств решения задачи

Решение задачи на ЭВМ представляет собой процесс получения информации на основе обработки исходной задачи с помощью программы, составленной из команд системы управления работой отдельных устройств вычислительной машины. Создание такой программы решения задачи предполагает выполнение ряда последовательных этапов технологического процесса, принципиальная схема которого представлена на рис. 5.1.

В зависимости от специфических особенностей конкретной задачи (ее вычислительной и логической сложности, состава и структуры исходной, промежуточной и результатной информации и т. п.), от профессионального уровня подготовки специалистов и ряда других факторов некоторые этапы технологического процесса, представленные на общей схеме (рис. 5.1), могут быть объединены в более крупные.

На **первом этапе** осуществляется постановка задачи, т. е. раскрывается ее организационно-экономическая сущность: формулируется цель и указывается периодичность ее решения; определяется взаимосвязь с другими задачами; раскрываются состав и форма представления входной, промежуточной и результатной информации; характеризуются формы и методы контроля достоверности информации на ключевых этапах решения задачи; специфицируются формы взаимодействия пользователя с ЭВМ в ходе ее решения и т. п.

Особое внимание в процессе постановки задачи уделяется детальному описанию входной, выходной (результатной) и промежуточной информации. При этом характеризуются:

— форма представления отдельных данных (цифровая, символьная и т. д.);

— количество знаков (разрядов), выделяемых для записи данных, исходя из их максимальной значимости;

— вид данных в процессе решения (первичный, расчетный, нормативный, справочный и т. п.);

— источник (документ) возникновения данных.

Кроме того, для цифровой информации указываются характер величин данных — целочисленный или дробный (для дробных величин дополнительно указывается количество десятичных знаков — разрядов, выделяемых для записи дробной части числа) и допустимый диапазон изменения величин (т. е. максимальное и минимальное значения данных).

Для расчетных данных дается соответствующее описание расчетов и особо выделяются те данные, которые используются при последующих решениях задачи, так как эта информация должна сохраниться в памяти ЭВМ.

Завершается постановка задачи описанием контрольного примера, демонстрирующего порядок ее решения традиционным способом. Основное требование к контрольному примеру: он должен отражать все многообразие возможных форм существования исходных данных. Контрольный пример сопровождается перечислением нештатных ситуаций, которые могут возникнуть при решении задачи, и описанием действий пользователя в каждой конкретной ситуации.

Особенность реализации этого этапа технологического процесса заключается в том, что конечный пользователь разрабатываемой программы, хорошо знающий ее проблемную сторону, обычно слабо представляет специфику и возможности ЭВМ для ее решения. В свою очередь, предметная область пользователя часто незнакома разработчику программ, хотя он знает возможности и ограничения на применение ЭВМ. Именно это противоречие является основной причиной возникновения ошибок при реализации данного этапа технологического процесса разработки программ. По данным экспертов, на этот этап приходится более 50 % общего числа ошибок, обнаруживаемых в процессе разработки программ решения задач организационно-экономического характера, а затраты на исправление таких ошибок составляют в среднем 80 % всех усилий разработчиков на поиск и устранение ошибок в программе.

Отсюда вся важность и ответственность этого этапа, необходимость осуществления корректной и полной постановки задачи, а также однозначность ее понимания как разработчиком программы, так и ее пользователем, в качестве которого обычно выступает постановщик задачи.

Второй этап технологического процесса разработки программы — математическое описание задачи и выбор метода ее решения. Выделение этого этапа обусловливается рядом причин, одна из которых вытекает из свойства неоднозначности естественного языка, на котором осуществляется описание постановки задачи. В связи с этим на втором этапе технологического процесса разработки программы выполняется формализованное описание задачи, т. е. устанавливаются и формулируются средствами языка математики логико-математические зависимости между исходными и результатными данными.

Математическое описание задачи обеспечивает ее однозначное понимание постановщиком (пользователем) задачи и разработчиком программы, реализующей эту задачу.

В процессе подготовки математического описания (модели) задачи могут использоваться различные разделы математики, особенно прикладной. Так, при решении экономических задач наиболее часто используются следующие классы моделей для формализованного описания их постановок:

— аналитические (вычислительные);

— матричные (балансовые);

— графические (частным видом которых являются сетевые).

Выбор класса модели, а иногда и конкретной формы ее представления внутри одного и того же класса в ряде случаев позволяет не только облегчить и ускорить процесс решения задачи, но иногда и повысить точность получаемых результатов. Хотя математическая запись постановки задачи, как правило, отличается высокой точностью отображения ее сущности, лаконичностью записи, а главное, однозначностью понимания, далеко не для всех задач она может быть выполнена. Кроме того, математическое описание задачи в большинстве случаев трудно однозначно перевести на язык ЭВМ. Для задач, допускающих возможность математического описания, необходимо выбрать численный метод решения, а для нечисловых задач — принципиальную схему решения в виде однозначно понимаемой по-

следовательности выполнения элементарных математических и логических функций (операций).

При выборе метода решения задачи предпочтение отдается методу, который наиболее полно удовлетворяет следующим основным требованиям:

— обеспечивает необходимую точность получаемых результатов;

— не обладает свойством вырождения, т. е. бесконечного зацикливания на каком-либо участке решения задачи при определенных исходных данных;

— позволяет использовать уже готовые стандартные программы для решения задачи или ее отдельных фрагментов;

— ориентирован на минимальный объем исходной информации;

— обеспечивает наиболее быстрое получение искомых результатов решения.

Сложность и ответственность этапа подготовки математического описания задачи и выбора (разработки) соответствующего метода ее решения часто требует привлечения квалифицированных специалистов в области прикладной математики, обладающих знанием таких дисциплин, как исследование операций, математическая статистика, численный анализ, вычислительная математика и т. д.

Третий этап технологического процесса подготовки решения задачи на ЭВМ представляет собой алгоритмизацию ее решения, т. е. разработку оригинального или адаптацию (уточнение и корректировку) уже известного алгоритма.

Алгоритмизация — сложный процесс, носящий в значительной степени творческий характер. По оценкам специалистов, постановка задачи и ее алгоритмизация нередко составляют 20—30 % общего времени на разработку программных средств ее решения. Сложность и ответственность реализации данного этапа объясняется тем, что для решения одной и той же задачи, как правило, существует множество различных алгоритмов, отличающихся друг от друга уровнем сложности, объемами вычислительных работ, составом необходимой исходной и промежуточной информации и другими факторами, которые оказывают существенное влияние на эффективность выбранного способа решения.

Четвертый (завершающий) **этап** технологического процесса разработки программных средств — составление, кодирование, тестирование и отладка программы. Он предшествует началу непосредственно машинной реализации алгоритма решения задачи. Процесс *кодирования* заключается в переводе описания алгоритма на один из доступных (понятных) для ЭВМ языков программирования. В процессе *составления программы* для ЭВМ конкретизируются тип и структура используемых данных, а последовательность действий, реализующих алгоритм, отражается посредством языка программирования.

Тестирование и отладка — заключительные этапы разработки программы решения задач. Эти процессы функционально связаны между собой, хотя их цели несколько отличаются друг от друга. *Тестирование* представляет собой совокупность действий, предназначенных для демонстрации правильности работы программы в заданных диапазонах изменений внешних условий и режимов эксплуатации программы. Цель тестирования заключается в демонстрации отсутствия (или выявлении) ошибок в разработанных программах на наборе заранее подготовленных контрольных примеров.

Тестированию сопутствует процесс *отладки,* т. е. выполнения совокупности действий, направленных на устранение ошибок в программах, начиная с момента обнаружения фактов ошибочной работы программы и заканчивая устранением причин их возникновения.

По своему характеру (природе возникновения) ошибки в программах делятся на синтаксические и логические.

Синтаксические ошибки в программе представляют собой некорректную запись отдельных языковых конструкций с точки зрения правил их представления на выбранном языке программирования. Эти ошибки выявляются автоматически при *трансляции* исходной программы (т. е. в процессе перевода с исходного языка программирования во внутренние коды машины) для ее выполнения. После устранения синтаксических ошибок проверяется *логика работы программы* на заданных исходных данных. При этом возможны следующие основные виды проявления логических ошибок:

— в какой-то момент программа не может продолжать работу (возникает программное прерывание, обычно сопровождающееся указанием оператора, на котором оно произошло);

— программа работает, но не выдает всех запланированных результатов и не останавливается (происходит ее «зацикливание»);

— программа выдает результаты и завершает свою работу, но эти результаты полностью или частично не совпадают с контрольными. После выявления логических ошибок и установления причин их возникновения в программу вносятся соответствующие исправления, и ее отладка продолжается.

Программа считается отлаженной, если она безошибочно выполняется на достаточно представительном наборе тестовых данных, обеспечивающих проверку всех ее ветвей.

Процесс тестирования и отладки программ носит итерационный характер и считается одним из наиболее трудоемких этапов разработки программ. По оценкам специалистов, он может составлять от 30 до 50 % (а иногда и больше) от общего времени, затрачиваемого на разработку проекта, и зависит от объема и логической сложности разрабатываемых программных комплексов.

Для сокращения затрат на проведение тестирования и отладки в настоящее время широко используются специальные программные средства тестирования (например, генераторы тестовых данных) и приемы отладки (например, метод трассировки программ).

5.2.3. Эксплуатация программных средств

После завершения процесса тестирования и отладки программные средства вместе с сопроводительной документацией передаются пользователю для *эксплуатации*. Основное назначение сопроводительной документации — обеспечить пользователя необходимыми инструктивными материалами для работы с программными средствами. Как правило, это документы, регламентирующие работу пользователя при эксплуатации программных средств, а также содержащие информацию о программе, необходимую для изменений и дополнений в ней, которые могут потребоваться при дальнейшей ее модернизации. Сопроводительная документация призвана также облегчить процесс выявления причин возникновения тех или иных ошибок в работе программ, которые могут быть обнаружены уже в ходе их эксплуатации пользователем.

Состав сопроводительной документации обычно оговаривается заказчиком (пользователем) и разработчиком технического задания на программное средство.

Для передачи пользователю разработанных программных средств обычно создается специальная комиссия, включающая в свой состав представителей как разработчиков, так и заказчиков (пользователей). Комиссия в соответствии с заранее составленным и утвержденным обеими сторонами планом проводит работы по приемке-передаче программных средств и сопроводительной документации. По завершении работы комиссии оформляется акт приемки-передачи.

В процессе внедрения и эксплуатации программных средств могут быть выявлены различного рода ошибки, не обнаруженные разработчиком при их тестировании и отладке. Поэтому при реализации достаточно сложных и ответственных программных комплексов этап эксплуатации программных средств может быть разбит на подэтапы *экспериментальной (опытной)* и *промышленной эксплуатации*. Смысл экспериментальной эксплуатации заключается во внедрении разработанных программных средств на объекте заказчика (иногда параллельно с традиционными методами решения задач) с целью проверки их работоспособности при решении реальных задач в течение достаточно большого периода времени (обычно не менее года). Только после завершения периода экспериментальной эксплуатации и устранения выявленных при этом ошибок программное средство передается в промышленную эксплуатацию.

Для повышения качества работ, оперативности исправления ошибок, выявляемых в процессе эксплуатации программных средств, а также выполнения различного рода модификаций разработчик (поставщик) может по договоренности с заказчиком (пользователем) осуществлять сопровождение программных средств. Целесообразность привлечения высококвалифицированных специалистов для сопровождения программных средств у пользователей объясняется тем, что затраты на сопровождение программ в большинстве случаев значительно превосходят первоначальные затраты на их разработку (приобретение).

Следует принимать во внимание, что по своему характеру и последовательности выполняемых действий внесение различного рода изменений в уже функционирующие программы

представляет собой в значительной мере повторение рассмотренных выше этапов, начиная с постановки задачи и заканчивая внесением изменений в сопроводительную документацию.

Описанная схема технологического процесса разработки и эксплуатации программных средств отражает их жизненный цикл.

Длительность и высокая стоимость разработки программных средств обусловливает целесообразность применения индустриальных методов их разработки, тиражирования и распространения. По мнению специалистов, в ближайшее время примерно 20—30 % финансовых средств пользователей, выделяемых на программное обеспечение систем обработки информации, будет затрачиваться на приобретение готовых программных средств.

КОНТРОЛЬНЫЕ ВОПРОСЫ К РАЗДЕЛУ 5

1. **Раскройте понятие «гипотеза».**
2. **Что такое аналогия?**
3. **В чем состоит содержание понятия «модель»?**
4. **В каком соотношении находятся понятия «гипотеза», «аналог», «модель»?**
5. **Дайте определение научного метода «моделирование».**
6. **Каково соотношение эксперимента и моделирования?**
7. **Как вы понимаете два подхода — аналитический и имитационный?**
8. **Какие средства моделирования объектов вам известны?**
9. **Что такое структурный и функциональный подходы к описанию объектов?**
10. **Расскажите о математическом моделировании.**
11. **Что такое формализм?**
12. **Объясните формальную модель объекта.**
13. **Что такое информационная модель объекта?**
14. **Перечислите основные этапы решения задачи на компьютере.**

Раздел 6

АЛГОРИТМИЗАЦИЯ

В результате изучения материала шестого раздела студент должен:

знать:

— основные алгоритмические конструкции;

— способы записи алгоритмов;

— назначение подпрограмм;

уметь:

— разрабатывать простейшие алгоритмы и записывать их в графическом представлении;

— использовать графические представления для построения алгоритмов;

— приводить примеры алгоритмов;

— перечислять свойства алгоритмов.

Методические указания

Целью изучения материала шестого раздела является выработка у студентов основ алгоритмического мышления. Преподаватель должен четко объяснить им, что алгоритм — это организованная последовательность действий, допустимых в определенных случаях, а программа — это алгоритм, записанный на языке, понятном программисту и ЭВМ. При этом студенты должны уметь распознавать, подходит ли данный алгоритм для решения задач данного класса. Важным условием усвоения основ алгоритмизации является четкое понимание сути ветвления, циклов и рекурсии: ветвление в алгоритмах появляется тогда, когда необходимо сделать выбор одного из нескольких действий в зависимости от некоторого условия; любой выбор

можно свести к одному или нескольким ветвлениям; при записи ветвлений необходим указатель конца ветвления, отделяющий его от остальной части алгоритма.

Тема 6.1

ПОНЯТИЕ АЛГОРИТМА. СВОЙСТВА АЛГОРИТМА. СПОСОБЫ ЗАПИСИ АЛГОРИТМОВ

6.1.1. Понятие алгоритма

Можно утверждать, что каждый читающий эти строки знаком с термином «алгоритм». Его применяют весьма широко и не только в области вычислительной техники и программирования. Несомненно и то, что у читателя сформировалось свое (пусть даже большей частью интуитивное) понимание смысла этого термина.

В основу процесса алгоритмизации положено фундаментальное понятие математики и программирования — «алгоритм». Слово «алгоритм» происходит от латинизированного воспроизведения арабского имени узбекского математика Аль-Хорезми, жившего в конце VIII — начале IX в., который первым сформулировал правила, позволяющие систематически составлять и решать квадратные уравнения.

Европейцы, начавшие осваивать современную десятичную систему счисления в XII в., знакомились с трудами арабских ученых, и труд упомянутого выше жителя Хорезма, посвященный правилам счета в десятичной системе счисления, был широко известен. Поэтому и наполнение термина «алгоритм» было следующим: операции над числами.

Со временем прежнее понимание этого термина утратилось, и его стали применять по отношению к одному-единственному алгоритму — алгоритму Евклида, который был предназначен для нахождения наибольшего общего делителя пары натуральных чисел (m, n).

Алгоритм Евклида:

1. {Нахождение остатка} r:= m mod n.
2. {Замена} m:=n; n:=r.
3. {Остановка?} Если n¯0, то переход к п. 1.
4. {Остановка процесса} m — искомое число.

Представленное описание алгоритма — это последовательность шагов, направленных на достижение некоторого результата (наибольшего общего делителя).

Теория алгоритмов — это раздел математики, изучающий общие свойства алгоритмов. Современное понятие «алгоритм» сформировалось в математике в 1920-х гг. Началом систематической разработки теории алгоритмов можно считать 1936 год, и связывают это с публикацией работы А.А. Черча.

Под алгоритмом всегда (и до возникновения строгой теории) понималась процедура, которая позволяла путем выполнения последовательности элементарных шагов получать однозначный результат (не зависящий от того, кто именно выполнял эти шаги) или за конечное число шагов прийти к выводу о том, что решения не существует.

Конечно же это не строгое определение понятия алгоритма, и именно попытки сформулировать такое понятие привели к возникновению теории алгоритмов. Причиной развития этой теории были внутренние проблемы математики, и лишь с возникновением и развитием вычислительной техники и смежных наук выяснилось, что в их основе должна лежать теория алгоритмов. Так стало очевидным прикладное значение новой науки.

В основе формализации понятия «алгоритм» лежит идея построения алгоритмической модели. Составляющими такой модели должны быть конкретный набор элементарных шагов, способы определения следующего шага и т. д. От модели также требуются простота и универсальность. Требование простоты важно для того, чтобы выделить действительно необходимые элементы и свойства алгоритма и облегчить доказательства общих утверждений об этих свойствах. Универсальность необходима для того, чтобы модель позволяла описать любой алгоритм.

Алгоритм — это точное предписание, которое задает алгоритмический процесс, начинающийся с произвольного исходного данного (из некоторой совокупности возможных для конкретного алгоритма исходных данных) и направленный на получение полностью определенного этим исходным данным результата.

Алгоритмический процесс (алгоритмизация) — это процесс последовательного преобразования конструктивных объектов

(слов, чисел, пар слов, пар чисел, предложений и т. п.), происходящий дискретными шагами. Каждый шаг состоит в смене одного конструктивного объекта другим.

Поскольку алгоритмы могут применяться к весьма произвольным объектам (числам, буквам, словам, графам, логическим выражениям и т. д.), в определении алгоритма используется специальный термин — «конструктивный объект», объединяющий в себе все эти возможные случаи. Так, в алгоритме Евклида под конструктивными объектами можно понимать пары чисел.

Процесс алгоритмизации решения задачи в общем случае реализуется по следующей схеме:

— выделение автономных этапов процесса решения задачи (как правило, с одним входом и одним выходом);

— формализованное описание содержания работ, выполняемых на каждом выделенном этапе;

— проверка правильности реализации выбранного алгоритма на различных примерах решения задач.

Развитие ЭВМ сделало понятие алгоритма одним из центральных в прикладной математике, так как возникла острая потребность в определении общих способов формирования и единообразного решения целых классов задач управления на основе разработки комплексов универсальных алгоритмов.

Наряду с трактовкой алгоритма в соответствии с ГОСТ 19.004—80 («алгоритм — это точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату»), существует и иное определение: алгоритм — это конечный набор правил, однозначно раскрывающих содержание и последовательность выполнения операций для систематического решения определенного класса задач за конечное число шагов.

Таким образом, алгоритм представляет собой заранее определенное, точное предписание, которое задает дискретный (пошаговый) процесс, начинающийся определенным образом и приводящий к результату за конечное число шагов. Алгоритм относится к исходным математическим понятиям, которые не могут быть определены через другие, более простые понятия. Иногда такое или подобное определение называют **интуитивным**, т. е. понятным из опыта.

Каждый алгоритм должен задаваться:

— множеством допустимых исходных данных;

— начальным состоянием;

— множеством допустимых промежуточных состояний;

— правилами перехода из одного состояния в другое;

— множеством конечных результатов;

— конечным состоянием.

В зависимости от конкретного задания этих параметров определяются классы алгоритмов, например алгоритмы линейные, циклические, сортировки и т. д.

Уточнение понятия алгоритма в интуитивном смысле представляется в виде машины Тьюринга, машины Поста, нормального алгоритма Маркова и пр.

Машина Тьюринга — это математическое построение, предназначенное для уточнения понятия алгоритма и называемое машиной потому, что при построении используются некоторые понятия реальных машин — «память», «команда» и пр. Машина получила имя английского математика А. Тьюринга и решает следующую проблему: если для решения задачи можно построить машину Тьюринга, она алгоритмически разрешима.

Машина Тьюринга состоит из неограниченной в обе стороны ленты, разделенной на ячейки, последовательно пронумерованные целыми числами, как положительными, так и отрицательными. В каждой ячейке ленты может стоять любой символ из заданного алфавита, в котором выделен «пустой» символ — признак того, что ячейка пустая.

Машина имеет конечное множество внутренних состояний: начальное (с него начинается работа машины) и конечное, попав в которое машина прекращает работу. Кроме ленты, имеется головка чтения/записи, которая умеет двигаться вперед, назад и стоять на месте, читать содержимое, стирать и записывать символы из данного алфавита.

Машина управляется программой. Программа — это таблица, в каждой клетке которой записана команда. Каждая клетка определяется двумя параметрами — символом алфавита и состоянием машины. Команда — это указание, куда передвинуть головку чтения/записи из текущего состояния, какой символ записать в текущую ячейку и в какое состояние перейдет машина.

Можно ли любой алгоритм представить в форме машины Тьюринга? Ответ на этот вопрос дается в виде так называемого тезиса Тьюринга: всякий алгоритм представим в форме машины Тьюринга. Это тезис потому, что его невозможно доказать, так как в нем фигурируют, с одной стороны, интуитивное понятие «всякий алгоритм», а с другой — точное понятие «машина Тьюринга».

Машина Поста — это математическое построение для уточнения понятия алгоритма. Машиной называется потому, что при построении используются некоторые понятия реальных машин — «память», «команда» и пр.

Машина получила имя американского математика Э. Поста и решает следующую проблему: если для решения задачи можно построить машину Поста, то она алгоритмически разрешима.

Машина Поста состоит из неограниченной в обе стороны ленты, разделенной на ячейки, последовательно пронумерованные целыми числами, как положительными, так и отрицательными. В каждой ячейке ленты стоит либо признак того, что в ячейке записана метка, либо признак того, что ячейка пустая. Состояние ленты — это данные, какие ячейки заняты, а какие пусты. Кроме ленты имеется головка чтения/записи, которая умеет двигаться вперед и назад, стоять на месте, читать содержимое, стирать и записывать метку.

Машина управляется программой, в которую могут входить в любой комбинации и в любом количестве шесть команд: 1) вправо; 2) влево; 3) поставить метку; 4) стереть метку; 5) передать управление на один номер команды в программе, если в текущей ячейке есть метка; если метки нет, то передать управление на другой номер команды; 6) прекратить работу машины.

Состояние машины — это состояние ленты и положение головки чтения/записи.

Несмотря на внешнюю простоту, машина Поста может производить различные вычисления, для чего надо задать начальное состояние машины и программу, которая эти вычисления сделает.

Машина Поста и машина Тьюринга эквивалентны по своим возможностям; разработаны практически в одно и то же время (в 1936 г.) независимо друг от друга.

Класс нормальных алгоритмов Маркова и класс алгоритмов, представленных в форме машин Тьюринга и Поста, совпадают.

Нормальный алгоритм Маркова — это математическое построение для уточнения понятия «алгоритм», задается алфавитом A и нормальной схемой подстановок. Алгоритм назван по имени автора — русского математика А.А. Маркова (1856—1922).

Алфавит — конечное, непустое множество элементов, называемых буквами. Различные сочетания букв образуют слова.

Нормальная схема подстановок — это конечный набор, состоящий из пар слов, где левое слово переходит в правое (но не наоборот).

Нормальным алгоритмом в алфавите A называют следующий алгоритм построения последовательности слов: в качестве начального слова берется само слово P и к нему применяют по порядку каждую пару из схемы подстановки. Если подстановка возможна, то ее осуществляют и начинают подстановки сначала. Если процесс обрывается (нет ни одной допустимой подстановки) на слове Q или приходит в конечную подстановку, то данный нормальный алгоритм преобразовал P в Q.

Если есть задача: от P перейти к Q, и доказано, что нельзя построить нормальную схему, то имеет место алгоритмически неразрешимая задача.

Можно ли любой алгоритм представить в виде нормального алгоритма Маркова? На этот вопрос дается ответ в виде так называемого тезиса (гипотезы) Маркова: всякий алгоритм в алфавите A представим в виде нормального алгоритма в этом же алфавите. Это тезис потому, что его невозможно доказать, так как в нем фигурируют, с одной стороны, интуитивное расплывчатое понятие «всякий алгоритм», а с другой — точное понятие «нормальный алгоритм».

6.1.2. Свойства и способы записи алгоритмов

Любой алгоритм обладает следующими свойствами: детерминированностью, массовостью, результативностью, дискретностью и конечностью.

Детерминированность (определенность, обусловленность) означает, что набор указаний алгоритма должен быть однозначно и точно понят любым исполнителем. Это свойство определяет однозначность результата работы алгоритма при заданных исходных данных.

Массовость алгоритма предполагает возможность варьирования исходных данных в определенных пределах. Свойство массовости определяет пригодность использования алгоритма для решения множества задач данного класса и является основным фактором, обеспечивающим экономическую эффективность решения задач на ЭВМ. Из сказанною следует, что для задач, решение которых осуществляется один раз, целесообразность использования ЭВМ, как правило, диктуется внеэкономическими категориями.

Результативность алгоритма предполагает, что для любых допустимых исходных данных он должен через конечное число шагов (или итераций) завершить свою работу.

Дискретность алгоритма допускает разбиение определенного алгоритмического процесса на отдельные элементарные этапы, возможность реализации которых человеком или ЭВМ не вызывает сомнения, а результат выполнения каждого элементарного этапа вполне определен и понятен.

Конечность алгоритма означает, что он должен выполняться за конечное время.

Таким образом, алгоритм дает возможность чисто механически решать любую конкретную задачу из некоторого класса однотипных задач и предполагает наличие определенных свойств:

— алгоритм состоит из отдельных шагов;

— каждый шаг выполняет обработку входных данных (аргументов), получая выходные данные (результаты);

— результат каждого шага однозначно определяется аргументами;

— количество шагов конечно;

— шаги алгоритма выполняются последовательно, в порядке написания (естественный порядок выполнения);

— существуют способы изменения естественного порядка выполнения (управляющие конструкции).

С понятием алгоритма тесно связано понятие «данные». В алгоритмическом аспекте данные — это информация, несущая полезную смысловую нагрузку, представленная в формализованном виде, позволяющем собирать, передавать, вводить и обрабатывать эту информацию с помощью заданных алгоритмов. Реализация алгоритма на конкретных исходных данных решаемой задачи называется **алгоритмическим процессом**.

Существует несколько способов описания алгоритмов: словесный, формульно-словесный, графический, средствами специального языка операторных схем, с помощью таблиц решений и др. Помимо требования обеспечения наглядности, выбор конкретного способа диктуется рядом факторов, из которых определяющими являются степень необходимой детализации представления алгоритма, степень его формализации, уровень логической сложности задачи и т. п.

Словесный способ описания алгоритма основан на записи содержания выполняемых действий средствами естественного языка. К достоинствам этого способа следует отнести его общедоступность, возможность описывать алгоритм с любой степенью детализации, а к главным недостаткам — довольно громоздкое описание (и как следствие — относительно низкую наглядность), отсутствие строгой формализации в силу неоднозначности восприятия естественного языка, вытекающего из свойств синонимии, омонимии, полисемии.

Формульно-словесный способ описания алгоритма основан на записи содержания выполняемых действий с использованием изобразительных возможностей языка математики, дополненного — с целью указания необходимых пояснений — средствами естественного языка. Обладая всеми достоинствами словесного способа, формульно-словесный способ вместе с тем более лаконичен, а значит, и более нагляден, имеет бóльшую формализацию, однако также не является строго формальным.

Графический способ описания алгоритма — это такое изображение логико-математической структуры алгоритма, при котором все этапы процесса обработки данных представляются с помощью определенного набора геометрических фигур (блоков), имеющих строго определенную конфигурацию в соответствии с характером выполняемых действий.

Для облегчения процесса разработки и восприятия графического изображения алгоритмов их составление осуществляется в соответствии с требованиями ГОСТ 19701—90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения» и ГОСТ 19.003—80 «Схемы алгоритмов и программ. Обозначения условные графические».

По назначению и характеру отображаемых функций условные графические изображения, называемые **символами**, делятся на основные и вспомогательные. Основные символы используются для представления операций, раскрывающих характер обработки данных в процессе решения задачи (рис. 6.1), вспомогательные — для пояснения отдельных элементов схемы алгоритма, а также обозначения связей между ними.

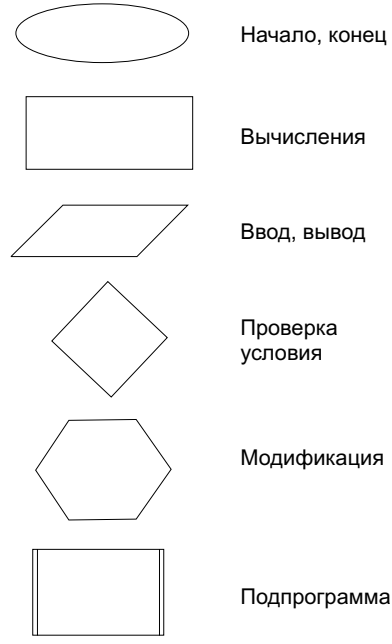


Рис. 6.1. Основные графические обозначения блоков алгоритмов и программ

Изображение схем алгоритмов осуществляется по определенным правилам, призванным повысить их наглядность и однозначность восприятия, что облегчает обнаружение логических ошибок в программах. В соответствии с этими правилами каждая схема должна начинаться и заканчиваться соответствующими символами, обозначающими начало и окончание алгоритма.

Все блоки в схеме располагаются в последовательности сверху вниз и слева направо, объединяясь между собой линиями потока. Нормальным направлением линий потока, т. е. следования этапов процесса решения задачи, принято направление сверху вниз и слева направо. В этом случае направление линий потока не идентифицируется с помощью стрелок, в отличие от других направлений (рис. 6.2).

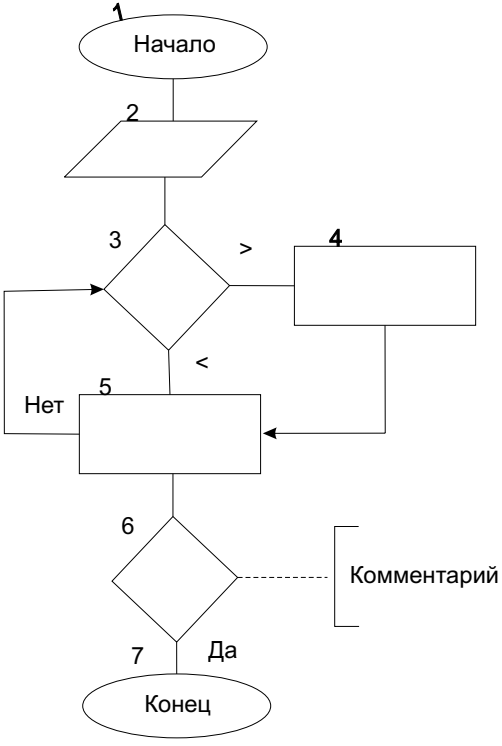


Рис. 6.2. Пример записи схемы алгоритма

Необходимым условием корректности схем алгоритмов является недопустимость более чем одного выхода из символов, обозначающих обрабатывающие блоки, и не менее двух выходов из символов, обозначающих логические операции по проверке выполнения условий.

С целью повышения наглядности графические схемы алгоритмов могут сопровождаться кратким формульно-словесным описанием внутри условного изображения блоков, раскрывающим содержание выполняемой операции. В случае необходимости представления более развернутых пояснений к блокам или линиям потока используются комментарии, идентифицируемые в схемах с помощью специальных изобразительных средств. Для обозначения условий передачи управления от блоков логических операций над соответствующими линиями потока могут записываться специальные знаки операций отношения («>», «<», «=», «=>», «<=») или слова ДА либо НЕТ (рис. 6.2).

Для удобства работы со схемами алгоритмов все блоки на схеме идентифицируются с помощью цифр или их сочетаний с буквами. Идентификаторы блоков размещаются в верхней левой части условного изображения в разрыве их контуров (рис. 6.2, 6.3).

Для отражения связи между удаленными друг от друга блоками соответствующие им линии потока можно разрывать.

В этом случае в конце и начале обрыва линии потока изображаются специальные символы:

— при отражении связей между блоками, расположенными на одной и той же странице, — внутристраничные соединители с идентификаторами, как правило, совпадающими с идентификаторами соответствующих блоков перехода (рис. 6.3);

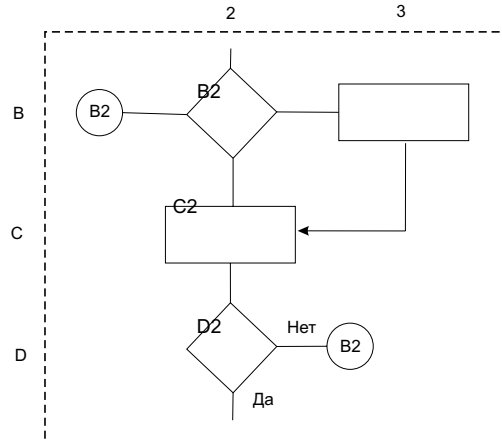


Рис. 6.3. Пример координатной идентификации блоков алгоритма

— при отражении связей между блоками, расположенными на разных страницах, — межстраничные соединители, в которых, помимо идентификации блоков, отражаются также номера соответствующих им страниц.

Операторный способ записи алгоритма представляет собой изображение последовательности операций процесса обработки данных с помощью заданного набора символов, обозначающих ту или иную типовую операцию (например, А — вычисление, В — ввод, Р — проверка условия и т. п.). Последовательность выполнения операций алгоритма определяется расположением операторов в схеме (при чтении слева направо в соответствии с цифровой индексацией). Передача управления от оператора к оператору осуществляется в порядке следования в символической записи алгоритма. В случае отсутствия передачи управления от очередного оператора к последующему оператору в записи между ними ставится признак завершения ветви алгоритма — символ «точка с запятой». Нарушение естественного порядка выполнения операторов отражается с помощью символов передачи управления (стрелок), которые используются:

— для указания перехода от условного оператора (Р) при разветвлении алгоритма;

— в случае отражения безусловного перехода от последнего оператора, завершающего одну из ветвей алгоритма.

Использование операторного способа представления алгоритма значительно упрощает процесс его записи, так как каждому оператору схемы обычно соответствует определенная совокупность достаточно простых операций обработки информации. Однако из-за малой наглядности отображения процесса обработки информации использование языка операторных схем не нашло практического применения для разработки и отражения алгоритмов решения задач экономического характера.

Перечисленные способы описания алгоритмов имеют существенный недостаток, так как не обеспечивают наглядности представления многовариантных вычислительных процессов, что характерно для алгоритмов решения сложных задач с разветвленной логикой.

Тема 6.2

ОСНОВНЫЕ АЛГОРИТМИЧЕСКИЕ КОНСТРУКЦИИ. ВСПОМОГАТЕЛЬНЫЕ АЛГОРИТМЫ

6.2.1. Основные типы алгоритмов

Рассмотрим три основных типа алгоритмов.

Линейный алгоритм — это простейший тип алгоритма, все шаги которого выполняются однократно и строго последовательно (рис. 6.4). Часто употребляется синоним понятия «линейный алгоритм» — «последовательное выполнение» — как одна из трех основных структур, применяемых при составлении алгоритмов, которая предполагает выполнение шагов алгоритма по порядку следования. При этом каждый шаг может быть простым (элементарным) и составным, который выполняется подпрограммой. **Подпрограммой** называется часть программы, оформленная специальным образом. К подпрограмме можно обращаться из других программ по мере необходимости. Она используется для замены часто повторяющихся фрагментов программы и для достижения большей ее компетентности. Во многих языках программирования понятие «подпрограмма» совпадает с понятием «процедура».

Разветвляющийся алгоритм (синонимы: ветвящийся алгоритм, ветвление) — это алгоритм, который позволяет в зависи-

мости от условий выполнять различные ветви (рис. 6.5). В некоторых литературных источниках данная структура называется **выбором**. К ветвлению в языках программирования высокого уровня относятся такие команды, как **if**, **then**, **else**, **case**, **of**.

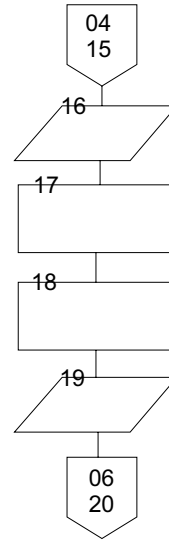


Рис. 6.4. Пример линейной структуры алгоритма

Циклический алгоритм — это:

— схема выполнения части алгоритма, в которой некоторые действия повторяются;

— путь в графе, начинающийся от одного из узлов и заканчивающийся в нем;

— оператор в процедурно-ориентированном языке программирования, обеспечивает выполнение части программы некоторое количество раз.

Существуют три типа операторов цикла (на примере языка Паскаль):

— с известным заранее (до цикла) количеством повторений, например цикл **for**;

— с проверкой условия выполнения цикла перед выполнением тела цикла, так называемый цикл с предусловием, например цикл **while**;

— с проверкой условия выполнения цикла после выполнения тела цикла, так называемый цикл с постусловием, например цикл **repeat**.

В одних языках программирования в цикле for проверка условий окончания цикла производится перед выполнением тела цикла, в других — после.

В языках программирования декларативного типа циклические конструкции выполняются с помощью рекурсивных построений.

Рекурсивный алгоритм — это алгоритм, который в своей структуре использует обращение к самому себе. Он имеет в своей основе рекурсивные функции. **Рекурсивной** называют функцию, если она в своем определении содержит вызов самой себя. Например, общеизвестное использование рекурсии — определение факториала (n!) произвольного числа n>=0: 0!=1, n!=n*(n−1)!.

Процедура — это часть программы для выполнения некоторых стандартных действий, зависящих в общем случае от входных параметров. Процедуру можно считать подпрограммой.

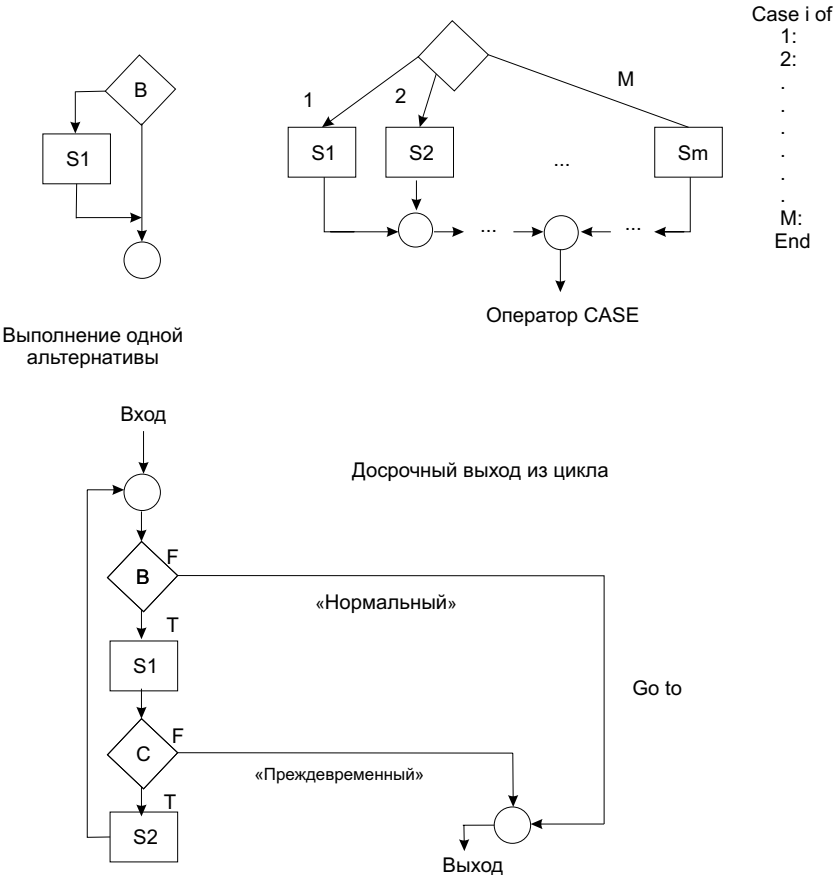


Рис. 6.5. Дополнительные (вспомогательные) структуры управления

Процедуры разделяются на стандартные и пользовательские, внешние и внутренние относительно программы.

Стандартная процедура входит в состав языка программирования и поставляется в составе соответствующей *библиотеки*, а *пользовательскую процедуру* создает каждый разработчик самостоятельно.

Внешняя процедура хранится независимо от программы, в которой имеется обращение к этой процедуре, а *внутренняя* входит в состав программы.

Как правило, для задания алгоритма можно выделить семь характеризующих его независимых параметров:

— совокупность возможных исходных данных;
— совокупность возможных промежуточных результатов;
— совокупность результатов;
— правило начала;
— правило непосредственной переработки;
— правило окончания;
— правило извлечения результата.

На практике в программировании очень часто используется задание алгоритмов в виде блок-схем.

Блок-схема — это ориентированный граф, вершины которого могут быть одного из трех типов (рис. 6.5).

Функциональная вершина используется для представления функции $f: X \to Y$.

Предикатная вершина применяется для представления функции (или предиката) $p: X \to (T, F)$, т. е. логического выражения, передающего управление по одной из двух возможных ветвей.

Объединяющая вершина представляет передачу управления от одной из двух входящих ветвей к одной выходящей ветви.

Важной особенностью приведенных структур является то, что они имеют один вход и один выход.

Структурное программирование — это процесс разработки алгоритмов с помощью структурных блок-схем. ***Структурная блок-схема*** — это блок-схема, которая может быть выражена как композиция из четырех элементарных блок-схем (рис. 6.6).

Любая программа для машины может быть представлена структурной блок-схемой.

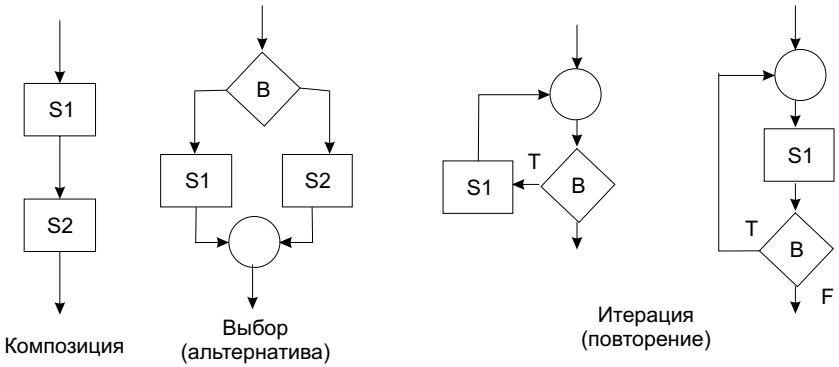


Рис. 6.6. Структуры управления программой

В более широком плане структурное программирование допускает большее разнообразие элементарных структур управления, чем предложенные четыре. Причиной для расширения множества структур является требование удобства и естественности.

6.2.2. Методы разработки алгоритма

Существует весьма большое количество всевозможных приемов и методов разработки алгоритмов. Однако среди них можно выделить небольшой набор основных методов, применяемых часто и лежащих в основе многих процедур и алгоритмов. Можно утверждать, что знание приведенных ниже методов необходимо для любого программиста.

Программирование сверху вниз. Это процесс пошагового разбиения алгоритма на все более мелкие части с целью получения таких элементов, для которых можно написать конкретные команды. Программирование сверху вниз ограничивается использованием структурных блок-схем.

Метод частных целей. Этот метод имеет весьма общую формулировку: «Необходимо свести трудную задачу к последовательности более простых задач». Приведенная рекомендация выглядит столь естественной и разумной, что вряд ли вызовет у кого-нибудь возражения. Более того, любой человек очень часто использует метод частных целей для решения стоящих перед ним задач, при этом даже не догадываясь (или не отдавая себе отчета) о его названии. С другой стороны, нередко трудно указать способ разбиения конкретной сложной задачи на набор более простых задач. Здесь большое значение имеют опыт и искусство специалиста. Тем не менее данный метод лежит в основе решения многих задач и по своей сути составляет базу алгоритмизации и программирования. Именно с вопроса, можно ли данную задачу разбить на последовательность (набор) более простых, и нужно начинать разработку простого алгоритма.

Метод подъема. Этот метод, как и предыдущий, можно отнести к одному из общих «рецептов» разработки алгоритмов. Его суть заключается в следующей процедуре. Алгоритм начинается с принятия начального предположения или построения начального решения задачи. Затем начинается (насколько возможно) быстрое движение «вверх» от начального уровня по

направлению к лучшим решениям. Когда алгоритм достигает точки, из которой больше невозможно двигаться «вверх», он останавливается.

Программирование с отходом назад. Основой программ искусственного интеллекта (независимо от того, к чему он прилагается — к программированию игр, выбору решений, распознаванию образов и т. п.) является программирование перебора вариантов. Это сложная задача, так как алгоритмы перебора ищут решения не по заданным правилам вычислений, а путем проб и ошибок, и схема не укладывается в схемы циклов, имеющихся в языках программирования. Ситуация зачастую осложняется тем, что прямыми методами перебор всех возможных вариантов невозможно осуществить из-за их огромного количества.

Метод программирования с отходом назад позволяет осуществить организованный исчерпывающий поиск требуемого решения задачи. При этом часто удается избежать перебора всех возможных вариантов.

Алгоритмы ветвей и границ. Такие алгоритмы, как и большинство описанных выше, применяются для решения переборных задач. Как и алгоритм с отходом назад, они исследуют древовидную модель пространства решений и ориентированы на поиск в некотором смысле оптимального решения (из конечного множества возможных решений-вариантов).

КОНТРОЛЬНЫЕ ВОПРОСЫ К РАЗДЕЛУ 6

1. **Дайте определение термина «алгоритм».**
2. **Что такое алгоритмический процесс?**
3. **Какими свойствами должен обладать любой алгоритм?**
4. **Расскажите о способах записи алгоритма.**
5. **Перечислите основные типы алгоритмов.**
6. **В чем отличие линейного алгоритма от разветвляющегося?**
7. **Раскройте особенности циклического и рекурсивного алгоритмов.**
8. **Какие вы знаете методы разработки алгоритма?**
9. **Сформулируйте метод частных целей разработки алгоритма.**
10. **Что такое программирование методом подъема и с отходом назад?**
11. **В чем преимущества метода ветвей и границ как метода разработки алгоритма?**

Раздел 7

ПРОГРАММИРОВАНИЕ

В результате изучения материала седьмого раздела студент должен:

знать:

— основные типы данных и формы их представления для обработки на компьютере;

— операторы ввода, вывода, присваивания, условные и циклические операторы языка программирования;

уметь:

— упорядочивать массив;

— искать минимальный и максимальный элементы массива с указанием местоположения;

— определять количество одинаковых и разных букв в тексте, количество слов в тексте;

— создавать движущиеся объекты на экране дисплея; производить численные расчеты на компьютере с использованием стандартных функций;

— использовать стандартные алгоритмы для решения учебных задач;

— записывать на языке программирования алгоритм решения учебной задачи и отлаживать его;

иметь представление:

— о современных языках и средствах программирования;

— о жизненном цикле программного обеспечения.

Методические указания

Студент должен уметь составлять «протоколы» выполнения разветвляющихся и циклических алгоритмов, мысленно совершая действия алгоритма и комментируя их; записывать разветв-

ляющиеся и циклические алгоритмы, не допуская двусмысленности записи (из записи алгоритма должно быть понятно, где начинаются и где кончаются ветвление и цикл); изменять ветвления и циклы при решении задач (при переходе от модели к алгоритму); использовать простейшие приемы отладки разветвляющихся и циклических программ; проводить вычислительные эксперименты по программам, написанным на языке программирования.

В результате изучения языка Паскаль (или Бейсик) студент должен уяснить, что язык программирования — это одно из средств общения с ЭВМ; знать правила перевода алгоритмов на язык программирования и перевода программ с одного языка на другой; усвоить понятия «слово», «символьная переменная», «кодирование и декодирование текстов»; освоить основные действия со словами; получить четкое представление о том, что:

— изучить язык программирования — значит знать, как в нем называются те или иные допустимые действия;

— при решении задач на ЭВМ можно пользоваться разными методами;

— одни методы могут быть эффективнее других (например, метод деления пополам обычно эффективнее метода простого перебора);

— алфавит — это произвольный набор символов, а слово — это любая последовательность символов некоторого алфавита;

— существуют различные способы кодирования текстов.

Тема 7.1

ЗНАКОМСТВО С ЯЗЫКОМ ПРОГРАММИРОВАНИЯ ПАСКАЛЬ

7.1.1. Введение в язык программирования Паскаль

В данном учебном пособии для изучения программирования выбран язык Паскаль. Он разработан известным швейцарским ученым Никлаусом Виртом в конце 1960-х гг. и назван в честь знаменитого французского физика Блеза Паскаля (1623—1662). Первоначально задуманный как структурный язык для обучения программированию, он быстро перерос в очень мощный, универсальный язык программирования высокого уровня,

процедурно-ориентированный. Язык Паскаль предназначен для решения широкого класса задач. Существует много версий этого языка, последние из них представляют объектно-ориентированные версии в среде Windows.

Рассмотрим основные базовые понятия, которые будут использоваться в дальнейшем изложении.

После описания, чтобы реализовать алгоритм, необходимо его представить в виде последовательности команд, которые могут быть выполнены на некоторой ЭВМ. Алгоритм, выраженный в терминах команд, называется **программой**. Таким образом, любое применение ЭВМ предполагает ее комплектацию определенным набором программ.

Программа должна описывать некоторый процесс, состоящий в считывании исходных данных, реализации набора действий по преобразованию информации (состав и порядок этих действий определяется исходными данными) и получении некоторого результата. Процесс, реализуемый с помощью программы, может быть достаточно сложным, поэтому должен быть описан с максимальной точностью. Для однозначного описания алгоритма перед реализацией его на ЭВМ используется система правил и обозначений, которая называется **алгоритмическим языком** или **языком программирования**.

Машинный язык описывает возможности, предоставляемые аппаратурой ЭВМ; как правило, это ограниченный набор команд и обрабатываемых структур данных. Основными компонентами ЭВМ являются процессорное устройство и память. Память представляет собой массив адресуемых двоичных данных. Работа процессорного устройства заключается в выборке данных из памяти, модификации их и записи обратно в память. Программирование на машинном языке требует точного знания того, как те или иные данные представлены в виде последовательности бит и какие машинные команды должны применяться для реализации требуемых операций. Язык большинства ЭВМ достаточно беден и состоит из небольшого набора команд-указаний типа «сложить числа», «прочитать число, размещенное в определенном месте памяти», «запомнить число в определенном месте памяти», «перейти к команде, расположенной в определенном месте» и т. д. Чем проще машинные команды и чем меньше предусматривается команд, тем более эффектив-

но они могут быть реализованы аппаратурой. Синтаксически машинные команды — не более чем последовательности двоичных цифр, плохо запоминаемые человеком.

В то же время желательно, чтобы используемый разработчиком программы язык максимально отражал его потребности, давал возможность реализовывать средства управления и работы со структурами данных, максимально соответствующие решаемой задаче. Такой язык программирования носит название **языка высокого уровня**. Ссылка на объекты в языке высокого уровня производится с помощью определенных пользователем имен, а не адресов памяти. Объекты данных связаны с типом, определяющим множество значений, которые могут приниматься объектами этого типа, и множеством операций, которые могут применяться к объектам этого типа. Для манипуляции данными используется множество синтаксических конструкций, называемых **операторами**. Можно привести следующие примеры операторов таких языков:

— операторы присваивания — по сути, это формулы, значения которых запоминаются в выделенных для этой цели переменных;

— операторы цикла — обеспечивают многократное повторение групп операторов;

— условные операторы — управляют изменением порядка вычислений в зависимости от значений данных;

— операторы ввода-вывода — необходимы для организации обмена программы с внешним миром;

— более или менее мощные операторы описания данных, обрабатываемых программой.

Чем разнообразнее конструкции, которые можно определить с помощью языка программирования, тем более сложная задача может быть реализована на данном языке.

Для перевода программы, написанной на языке высокого уровня, в машинную программу, используется программа, или система, называемая **компилятором**. Чтобы получить возможность выполнить программу, написанную на языке высокого уровня, ее необходимо откомпилировать. Процесс создания программы заключается в разработке алгоритма, его записи на некотором языке программирования и компиляции подготовленного таким образом текста для получения работающей ма-

шинной программы. Кроме того, производятся проверка правильности составленного алгоритма и его реализации в виде программы, соответствия получаемых с помощью программы результатов изначально поставленной задаче, а также поиск и исправление выявленных ошибок, т. е. *тестирование* и *отладка* программы.

Основные понятия и определения. Рассмотрим процесс чтения и восприятия человеком обычного текста, например книги. При чтении человек сканирует последовательность символов, с помощью которых кодируется текст; выделяет в тексте слова и знаки препинания; составляет из них отдельные предложения, интерпретируя таким образом смысл текста. Каждое предложение состоит из подлежащего, сказуемого и других членов предложения. Возможная структура предложений и правила связывания слов в предложения определяются грамматикой данного языка.

Программа на языке программирования есть не что иное, как текст, т. е. набор символов. Множество символов, используемых в языке программирования, называется **алфавитом языка**. Группы символов текста программы составляют **лексемы** — отдельные смысловые единицы текста. Можно выделить следующие классы лексем: идентификаторы, константы, операции, разделители, комментарии. Лексемы, в свою очередь, объединяются в предложения, называемые **операторами языка**. Язык программирования предоставляет набор правил, которые описывают способы комбинирования отдельных лексем для получения правильных операторов языка. Правила формирования работающих программ из набора существующих символов называются **синтаксисом языка**.

Программа на процедурно-ориентированном языке программирования содержит выполняемые и невыполняемые (описательные) операторы. ***Выполняемые*** операторы описывают некоторые действия по преобразованию данных или определяют порядок выполнения других исполняемых операторов. ***Невыполняемые*** операторы, или ***директивы,*** служат для описательных целей, используются для обозначения программных объектов, описания типов, переменных или констант, способов взаимодействия с другими программами, управления распределением памяти, ходом трансляции программы и др.

Идентификатор представляет собой строку символов, предназначенную для идентификации (или наименования) некоторого элемента программы. Идентификатор, или имя, обозначает объект независимо от его физического месторасположения или адреса и в языке программирования обычно представляет собой последовательность букв и цифр, начинающуюся с буквы. Состав букв может различаться, чаще всего допустимо использовать только буквы латинского алфавита. Идентификатор может содержать некоторые специальные символы, например, символ «_» (подчеркивание), причем он считается буквой. Пробелы и другие символы, не являющиеся буквенно-цифровыми, внутри идентификаторов недопустимы. Так, являются *допустимыми* следующие идентификаторы:

J7; Fanta; Program; Znachenie_Sinusa_Pi_Popolam.

Примеры *неправильных* идентификаторов:

7Up	(*начинается с цифры*);
Mild Seven	(*содержит внутри пробел*);
Bus_№_6	(*включает недопустимые символы*);
High-Level	(*содержит дефис*).

Синтаксис идентификатора в различных языках программирования имеет незначительные отличия: могут существовать ограничения на допустимую длину идентификатора, на количество символов, являющихся значимыми. В языке Паскаль допустимая длина идентификаторов не ограничена, но только первые 255 символов идентификатора считаются значимыми.

Чтобы однозначно именовать объекты, все идентификаторы должны быть уникальны в пределах одного программного контекста. Иногда одинаковые идентификаторы могут быть объявлены в составе различных структур, модулей или объектов в программе. Такие идентификаторы требуют уточнения (квалификации) с помощью другого идентификатора — имени структуры, модуля или объекта, в котором они объявлены. Пример общего вида квалифицированного идентификатора: Identifierl. Identifier2; здесь Identifierl квалифицирует (уточняет) Identifier2.

Вид элемента, именуемого с помощью идентификатора, зависит от языка программирования; в качестве таких элементов могут выступать переменная, структура данных, тип, процедура,

оператор или сама программа. Идентификаторы делятся на *служебные* (или зарезервированные) *слова* и *определяемые пользователем*.

Набор служебных (зарезервированных) слов служит для организации программы и отдельных операторов; например, идентификатор **Program** в языке Паскаль служит для обозначения программы, идентификатор **Function** — для обозначения функции.

Константы могут быть численными, символьными и строчными. *Численные* константы служат для описания целых и дробных чисел. Значение константы, описывающей целое число, может быть представлено цифрами от 0 до 9; кроме того, в ней может содержаться знак «+» или «—».

Примеры *целочисленных* констант:

—1524; 10; 1; 1234567890 (*целые числа*).

Константы с плавающей точкой дополнительно могут включать в себя точку, отделяющую целую и дробную часть числа, и знак порядка. Любой из этих элементов — но не оба сразу! — может отсутствовать.

Примеры *численных* констант, описывающих *дробные числа:*

—3.1415;
2.718281828459045;
1.0E-3 = 0.001;
1.5E3 = 1500;
1E6 = 1000000;
1234E5 = 123400000.

Символьные константы представляют собой литеры, заключенные в одинарные кавычки (апострофы), например: ‘x’ или ‘?’. Значением символьной константы является код символа в машинном наборе символов (алфавите).

Строковые константы представляют собой произвольную последовательность литер, заключенную в одинарные кавычки.

Примеры *строк* в языке Паскаль:

‘String constant’
‘Пример строковой константы в языке Паскаль’

Константы называются **самоопределенными**, поскольку их значение полностью определяется их описанием.

Операции, предусмотренные в языке программирования, обычно делятся на *бинарные*, в результате которых из двух входных переменных (операндов) формируется одна выходная переменная (результат), и *унарные*, служащие для преобразования одной переменной. В большинстве языков предусмотрен базовый набор арифметических операций (сложение, вычитание, умножение, деление), логических операций (И, ИЛИ, НЕ), операций сравнения («>», «<», «=») и сдвига.

Для обозначения операций могут использоваться специальные литеры или служебные (зарезервированные) слова. Зарезервированные слова, используемые для обозначения операций, не могут использоваться в качестве идентификаторов.

Примеры использования *специальных литер для обозначения операций:*

Литера	Операция
+	Сложение
*	Умножение
:=	Присваивание в языке Паскаль
—	Вычитание
/	Деление
|	Побитовое ИЛИ

Примеры использования *служебных идентификаторов для обозначения операций:*

Литера	Операция
SHR	Сдвиг вправо
AND	Логическая операция И
MOD	Взятие остатка от деления

Разделители служат для отделения лексем, знаков операций и операторов языка друг от друга. В качестве разделителей в большинстве языков используются пробел, символ табуляции, символ перехода на новую строку (фактически в конце строки стоят два символа — «перевод строки» и «возврат каретки»). Несколько пробелов, табуляций или символов перевода строки, идущих

подряд, трактуются как один разделитель. Символы-разделители обязательны в тех случаях, когда их отсутствие приведет к слиянию двух стоящих рядом лексем в одну; например, в строке

if y = z then

обязательны пробелы перед y и после z, а пробелы вокруг знака равенства необязательны — они служат только для удобочитаемости и могут быть опущены.

В качестве разделителя элементов различных списков чаще всего используется запятая. Алголоподобные языки отделяют отдельные операторы друг от друга с помощью знака «;» (точка с запятой).

Комментарий представляет собой часть текста программы, которая облегчает чтение программы человеком. Компилятором комментарии игнорируются. В каждом языке имеется свой собственный синтаксис комментариев. Обычно комментарии заключаются в скобки, например:

{комментарий} или (*комментарий*) в языке Паскаль.

Текст программы на языке Паскаль должен содержаться в дисковом файле стандартной для MS-DOS структуры. Он может быть сформирован любым текстовым редактором, работающим с ASCII-кодами, и представляет собой последовательность предложений, состоящих из символов, образующих алфавит языка. Максимальная длина предложения — 126 символов, рекомендуемая длина — 70 символов.

Алфавит языка Паскаль включает следующие символы:

— латинские прописные и строчные буквы (A — Z; a — z) и символ «подчерк» (код ASCII 95) — для формирования идентификаторов и служебных слов;

— арабские цифры (0—9) — для записи чисел и идентификаторов;

— 22 специальных символа:

математические: + | – | * | / | = | > | < |(|)|,

пунктуации: | . | — в конце программы; | , | : | ; | — после каждого оператора,

прочие: | [] | — квадратные скобки для обозначения массивов и множеств; | { } | или (* *) — комментарии; | ' ' | — для записи констант символьного и текстового типов; | $ | — для записи шестнадцатеричного числа и записи директив компилятору; | # | — для записи символа, не имеющего графического пред-

ставления в ASCII-кодах (по его коду); | @ | — обозначение адреса переменной, типизированной константы, подпрограммы, метода; | ^ | — обозначение типа-указателя, значения величины по ее указателю или управляющего символа (коды от 0 до 31).

Лексическая структура языка Паскаль. Программа на языке Паскаль состоит из последовательности лексем — минимальных лексических единиц языка, имеющих самостоятельный смысл. Лексемы условно делятся на несколько классов:

Ключевые (служебные, зарезервированные) ***слова***. Всего 51 слово, в редакторе интегрированной среды Borland Паскаль изображаются белым цветом: AND — логическое умножение (И); ARRAY — массив; BEGIN — начало; CASE — вариант; CONST — константа; DIV — деление нацело с отбрасыванием остатка; DO — исполнять; DOWNTO — уменьшать на единицу параметр цикла; ELSE — иначе; END — конец; FILE — файл; For — для; FUNCTION — функция; GOTO — переход к метке; IF — если; IN — принадлежность к множеству; LABEL — метка; MOD — нахождение остатка от деления нацело; NOT — логическое отрицание (НЕ); OF — из; OR — логическое сложение (ИЛИ); PACKED — упакованный; PROCEDURE — процедура; PROGRAM — первое слово программы; RECORD — запись; REPEAT — повторить; SET — множество; THEN — то; TO — увеличение на единицу параметра цикла; TYPE — тип; UNTIL — до; VAR — переменная; WHILE — пока; WITH — с и др.

Идентификаторы. Изображаются желтым цветом, могут быть двух разновидностей:

— *имена*, которые программист присваивает какой-либо переменной, константе, типу, метке, процедуре или функции (нельзя использовать ключевые слова)

— *стандартные идентификаторы,* которые являются именами встроенных в язык процедур и функций. Компилятор воспринимает 63 символа. Идентификатор должен начинаться с буквы, может содержать цифры и знак подчеркивания.

Знаки операций*:* := | <| >| + | – | * | / |;

Изображения. Эта группа лексем обозначает:

— *десятичные числа,* которые записываются чаще всего в традиционном формате с фиксированной точкой (дробную часть числа от целой отделяет точка, а не запятая):

<Вещ_фикс>::=<целое>.<целое> — 12.56.

Кроме того, применяется так называемый экспоненциальный формат:

<Вещ эксп>::=<Вещ_фикс>E<порядок>, где <порядок>::= [+|–] <целое>,

7.4E–2 Ж 7.4 * 10^{-2}; или 0.074;

— *строки* — последовательность любых символов из расширенного набора ASCII, заключенная в апострофы;

— *комментарии* (изображаются серым цветом) — любая комбинация произвольных символов, заключенная либо в фигурные скобки — {}, либо в комбинированные — (* *).

Если за скобкой следует знак $, комментарий интерпретируется как директива компилятора. Между лексемами разрешено вставлять один или несколько разделителей: пробелов, комментариев, символа «конец строки» (код 13) и других управляющих символов (коды от 0 до 31). Разделителями являются знаки математических операций, круглые и квадратные скобки, запятые.

7.1.2. Структура Паскаль-программы

Как уже было сказано, Паскаль является *алгоритмическим* языком в традиционном понимании этого термина. Иными словами, правильная программа на этом языке представляет собой формальную запись некоторого алгоритма, т. е. конечной последовательности действий, приводящих к решению некоторой задачи.

В соответствии с этим принципом программа на языке Паскаль всегда состоит из двух основных частей: описания *последовательности действий,* которые необходимо выполнить, и описания *данных,* с которыми оперируют действия. Действия представляются операторами языка, данные вводятся посредством описаний и определений. Кроме того, программа может быть снабжена заголовком, который задает имя программы и ее параметры. В авторском варианте языка заголовок реализует связь программы с вычислительной средой. Язык Турбо Паскаль рассматривает заголовок как комментарий. Текст программы должен завершаться символом «.» (точка).

Описания данных текстуально предшествуют описанию действий и должны содержать упоминания всех объектов, используемых в действиях (операторах). Совокупность описаний и определений и следующая за ней последовательность опера-

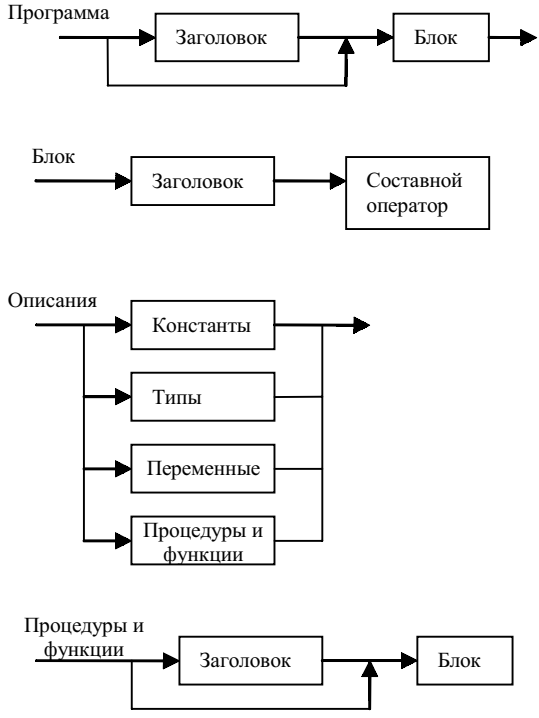


Рис. 7.1. Общая структура программы

торов называется **блоком**. Структура программы показана на рис. 7.1.

Объекты, вводимые посредством описаний и определений, имеют различную природу и могут быть разбиты на пять классов:

— метки;
— константы;
— типы;
— переменные;
— процедуры и функции.

Описание *меток* вводит совокупность идентификаторов и/или целых чисел, предназначенных для организации последовательности вычислений. Посредством меток можно отметить (указать) те операторы, на которые будет передано управление из других точек программы. Передача управления на помеченный оператор реализуется с помощью специального оператора перехода.

Определение *констант* задает в программе идентификаторы, являющиеся синонимами (представителями) некоторых значений.

Определение *типов* предназначено для задания конкретных множеств значений. Указанные множества обозначаются именами (идентификаторами) и в дальнейшем могут служить для описания переменных.

Описание *переменных* вводит совокупность данных, с которыми производятся действия. Переменная обозначается идентификатором; с каждой переменной связывается ее тип, определяющий множество допустимых значений этой переменной и набор допустимых операций.

Наконец, описание *процедуры* или *функции* определяет часть программы как отдельную синтаксическую единицу и сопоставляет с ней имя. Впоследствии действия, сосредоточенные в процедуре (функции), могут быть выполнены («вызваны») посредством указания ее имени. Кроме действий, описание процедуры (функции) может содержать совокупность описаний *локальных* объектов, образующих собственный контекст имен. Процедуры и функции являются основным средством структурирования программы.

В авторской версии языка Паскаль описания всех перечисленных объектов должны быть сосредоточены в соответствующих *разделах,* а порядок следования разделов фиксирован. Однако современные реализации, в частности Турбо Паскаль, не содержат такого ограничения. Иными словами, описания и определения всех объектов в блоке могут следовать в произвольном порядке. Соответствующая синтаксическая диаграмма представлена на рис. 7.2.

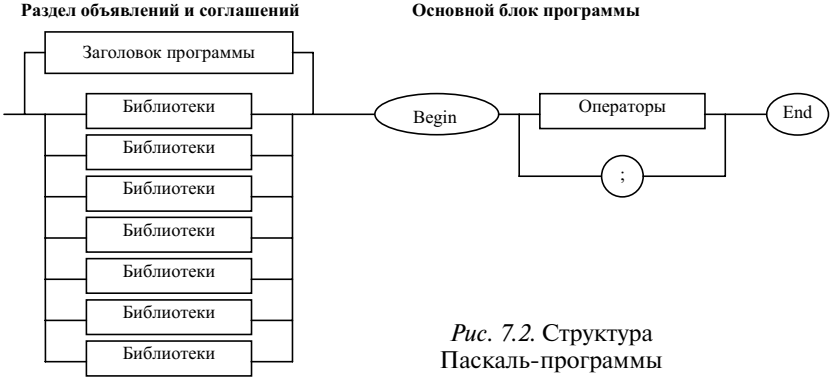


Рис. 7.2. Структура Паскаль-программы

Более подробно все введенные выше конструкции и объекты будут рассмотрены далее, а изложение данной подтемы завершает рассмотрение простой программы на языке Паскаль.

```
Program Example;
Var
n : integer;
R : real;
Begin
    Readln (n);
    R := 0;
    while n>0 do
    Begin
        R := R + 1/n;
        n := n – 1
    End;
    Writeln(R)
End.
```

Приведенная программа предназначена для вычисления ряда вида R = 1 + 1/2 + 1/3 + ... + 1/n. Первая строка программы представляет собой заголовок, который в данном случае состоит из служебного слова **program**, имени программы и символа, завершающего заголовок.

Далее следует собственно программа, или блок, который состоит из описания двух переменных и совокупности операторов. Описание переменных начинается со служебного слова **var** и вводит два объекта данных, поименованных идентификаторами **n** и **R**. Переменная **n** определяется как имеющая целый тип (**integer**) и предназначена для задания количества элементов ряда. Переменная **R** служит для вычисления результата (суммы ряда) и имеет вещественный тип (т. е. может принимать любое значение из некоторого стандартного подмножества целых чисел, обозначенного идентификатором **real**).

Действия (операторы) реализуют алгоритм вычисления суммы. Совокупность операторов заключена между служебными словами **begin** и **end**; друг от друга операторы отделяются символом <**;**> (точка с запятой). Первые два оператора устанавливают начальные значения переменных. Значение **n** определяется посредством чтения его из стандартного входного файла (в данном случае таким «файлом» является клавиатура ЭВМ). Чтение

реализуется вызовом стандартной (предописанной) процедуры **Readln**. Начальное значение переменной **R** устанавливается непосредственным заданием конкретного значения с помощью оператора присваивания.

Следующий оператор организует собственно процесс вычислений, определяя повторяющуюся последовательность действий (цикл). Строка **while n>0 do** является заголовком цикла и задает условие его повторения (в данном случае — неотрицательность текущего значения **n**). Два оператора присваивания, заключенные между служебными словами **begin** и **end**, составляют тело цикла. Первый из этих операторов добавляет к текущему значению суммы очередной элемент ряда, второй изменяет значение **n**, переходя тем самым к следующему элементу ряда.

Наконец, последний оператор программы осуществляет передачу вовне программы результата вычислений. Вызов стандартной процедуры **Writeln** выводит значение **R** в стандартный выводной файл (по умолчанию таким «файлом» является экран дисплея).

Процесс программирования на универсальном языке высокого уровня Паскаль состоит из следующих действий: ввода и редактирования текста программы, трансляции и отладки. Для повышения качества и скорости разработки программ была создана интегрированная система программирования Турбо Паскаль.

Процесс обработки программы на языке Паскаль может быть проиллюстрирован схемой на рис. 7.3.

Для выполнения каждого этапа применяются специальные средства интегрированной среды программирования: редактор

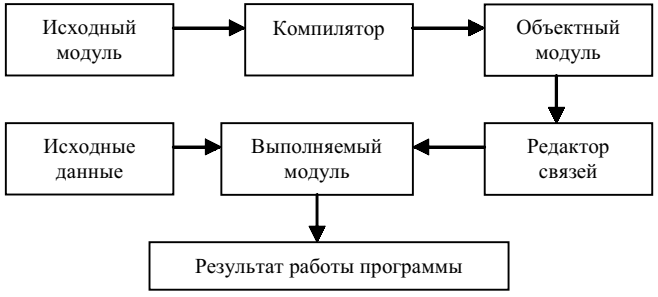


Рис. 7.3. Этапы процесса обработки программы на языке Паскаль

текстов (**editor**), компилятор (**compiler**), компоновщик (**linker**), отладчик (**debugger**). Краткая справка текстового редактора Турбо Паскаль и основные приемы работы с программами приведены в табл. 7.1.

Таблица 7.1

Справка текстового редактора Турбо Паскаль

ПЕРЕМЕЩЕНИЕ КУРСОРА
В начало/конец строки <Home>/<End>
На первую/последнюю строку экрана <Ctrl> + <Home>/<Ctrl> + <End>
В начало/конец текста программы <Ctrl> + <PgUp>/<Ctrl> + <PgDn
УНИЧТОЖИТЬ И ВОССТАНОВИТЬ
Вставить строку. Курсор перед или после строки <Enter>
Разделить строку на две части <Enter>
Соединить две строки — удалить перенос <BS>, <DEL>
Удалить строку <Ctrl> + Y
ОПЕРАЦИИ С ФАЙЛАМИ
Запись на диск под старым именем <F2>
Запись на диск под новым именем <F10>, File, Save as
Прочитать текст из дискового файла <F3>
Выход в главное меню <F10>
ОПЕРАЦИИ С БЛОКАМИ
Пометка блока (начало/конец) <Ctrl> + K, B <Ctrl> + K, K
Отмена пометки блока <Ctrl> + K, H
Копирование блока внутри файла <Ctrl> + K, C
Перемещение блока внутри файла <Ctrl> + K, V
Удаление блока <Ctrl> + K, Y
Записать блок в дисковый файл <Ctrl> + K, W
СПРАВОЧНАЯ СЛУЖБА
Помощь (информация об активном окне) <F1>
Вызов оглавления справочной информации <Shift> + <F1>
Вызов контекстной справки по языку <Ctrl> + <F1>
ОТЛАДОЧНЫЙ РЕЖИМ
Продолжить исполнение программы до курсора <F4>
Распахнуть активное окно на весь экран <F5>
Сделать активным следующее окно <F6>
Выполнить следующую строку программы <F7>
Выполнить процедуру или функцию <F8>
Компилировать программу <Alt> + <F9>
Выполнить прогон программы <Ctrl> + <F9>
Сменить окно редактора на окно результатов <Alt> + <F5>
Выйти из Турбо Паскаль <Alt> + <X>

7.1.3. Основные типы данных

Для характеристики данных в программах вводится понятие «тип». Каждая переменная должна быть отнесена к некоторому типу. Указание типа информационного объекта определяет его структуру, а также набор значений, которые может принимать элемент данных; размер памяти, отводимый для хранения объекта, и набор действий, которые можно осуществлять над данным объектом. Способ указания типа определяется синтаксисом языка программирования. Обычно это делается в начальных разделах программы (описаниях), в которых перечисляются переменные, встречающиеся в ней, и указываются их типы.

Тип переменной определяется при ее описании и не может быть изменен. Под *типом данных* понимается множество допустимых значений этих данных, а также совокупность операций над ними. На рис. 7.4 приведена классификация стандартных типов данных.

Базовые (простые) типы данных. Переменные типа данных «целое» принимают только целочисленные значения. Над целыми числами допустимы основные арифметические операции: сложение («+»), вычитание («–»), умножение («*»). Некоторой особенностью языка Паскаль являются обозначения целочисленных операций выделения частного и остатка от деления, для этого используются ключевые слова **div** и **mod** соответственно:

```
5 div 2;    // результат 2
5 mod 2     // результат 1
```

При реализации целых чисел обычно предполагается, что они непосредственно отображаются на машинное оборудование. Для достижения максимальной эффективности реализации размер поля, отводимого для хранения величины типа данных «целое», соответствует размеру слова, непосредственно обрабатываемого процессором; представление целого числа, таким образом, является машинно-зависимым. Для обозначения типа данных «целое» в языке Паскаль используется ключевое слово **integer**. Максимально возможное значение целого числа зависит от количества двоичных разрядов, используемых в конкретной ЭВМ для представления этого типа. Диапазон

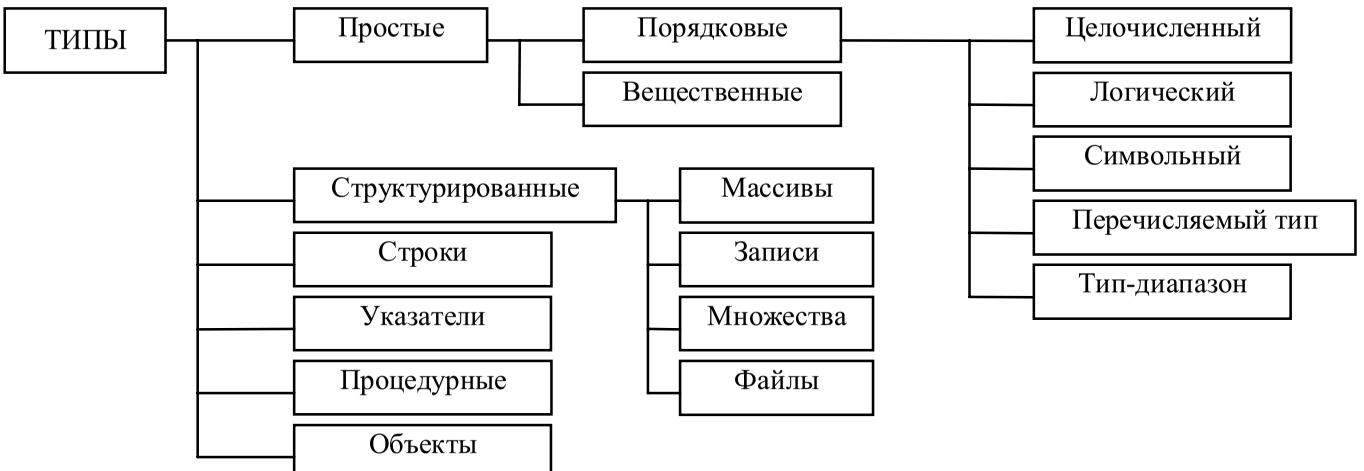


Рис. 7.4. Классификация стандартных типов данных

значений величины i типа данных «целое» при длине машинного слова N бит определяется соотношением

$$-2^{N-1} <= i < 2^{N-1}$$

при условии использования обычного двоичного дополнительного представления чисел. Для 32-битовой ЭВМ диапазон целых чисел ограничен константами –2 147 483 648 и +2 147 483 647.

Помимо целых чисел, допускается использование беззнаковых натуральных чисел. Диапазон таких чисел при длине машинного слова N бит определяется соотношением

$$0 <= i < 2^{N} - 1.$$

Для обозначения беззнаковых целых чисел в языке Паскаль используется ключевое слово **cardinal**; значения переменных этого типа на 32-битовой ЭВМ лежат в диапазоне от 0 до 4 294 967 295.

С целью преодоления излишней машинной зависимости в языке Паскаль определены дополнительные базовые целочисленные типы: **Shortint** — знаковое число длиной 8 бит; **Smallint** — знаковое число, 16 бит; **Longint** — знаковое число, 32 бита; **Byte** — беззнаковое, 8 бит; **Word** — беззнаковое, 16 бит; **Longword** — беззнаковое, 32 бита.

Тип элемента данных может быть *зарезервированным* (базовым) или *определяемым пользователем.* Вся информация о базовых типах данных встроена в язык программирования и не нуждается в дополнительном уточнении и описании в программе. Наиболее употребительный набор базовых типов состоит из целого, вещественного (с плавающей точкой), символьного, логического (булева), которые встречаются почти во всех языках программирования. Это означает, что если, например, для переменной с именем *x* указан тип integer, то описывать множество ее значений и набор допустимых операций не нужно, транслятор сделает это сам. Если объект логического типа, то по структуре он, видимо, очень простой и может принимать только логические значения «истина» (True) и «ложь» (False), а разрешенными для него операциями могут быть только логические операции типа И (AND), ИЛИ (OR) и НЕ (NOT).

Тип данных называется **простым**, если он характеризует элементарные данные, не имеющие внутренней структуры, и **сложным** (производным) — если данные этого типа состоят из нескольких данных, которые, в свою очередь, могут быть простыми или сложными.

Язык программирования обычно имеет набор базовых (предопределенных) простых типов. Например, в языке Паскаль существуют следующие базовые типы:

— целый;

— вещественный;

— логический;

— символьный.

Переменные ***вещественного*** типа принимают действительные значения. Машинное представление вещественного числа состоит из двоичной мантиссы длиной N бит, имеющей два различных значения, и характеристики, описывающей положение точки в числе. Следовательно, в каждой ЭВМ можно представить только конечное множество действительных чисел. Поэтому следует считать, что операции с вещественными числами представляют собой некоторую модель вычислений, дающую приблизительный результат. Точность представления результатов зависит от числа двоичных разрядов, выделенных для записи вещественных чисел в данной ЭВМ. Большинство языков позволяет манипулировать двумя вещественными типами — одинарной и двойной точности. Числа *одинарной точности* занимают меньше места; длина чисел *двойной точности* больше, следовательно, они обеспечивают бóльшую точность представления чисел.

Над вещественными числами допустимы обычные арифметические операции: сложение, вычитание, умножение, деление; в некоторых языках определена операция возведения числа в степень.

Результат арифметической операции над числами зависит не только от значений переменных, но и от их типа. Например, для $x = 12$ и $y = 5$ результат операции деления x на у равен 2, если x и y имеют целый тип и равен (возможно, приблизительно) 2.4, если x и y имеют вещественный тип. Точнее, деление целых и вещественных чисел — это две различные операции.

Для описания вещественных чисел в языке Паскаль используется ключевое слово **Real**. В современных реализациях языка Паскаль, помимо аппаратно-независимого типа Real, предусматриваются аппаратно-поддерживаемые типы **Single (**одинарной точности) и **Double** (двойной точности).

Вещественные типы:

Single: $+1.5*10^{-45}...+3.4*1038$ — 4 байта;

Real: $+2.9*10^{-39}...+1.7*10^{38}$ — 6 байт;

Double: $+5.0*10^{-324}...+1.7*10^{308}$ — 8 байт;

Extended: $+3.4*10^{-4932}...+1.1*10^{+4932}$ —10 байт;

COMP (целое в формате вещественного): $-9.2*10^{18}$... ...$+9.2*10^{18}$.

Булев (логический) тип Boolean имеет два значения: true (истинно) и false (ложно). Над значениями допустимы операции сравнения, причем считается, что false < true. Значения булева типа занимают один байт памяти. В версии Турбо Паскаль 7.0 добавлены еще три булева типа: ByteBool, WordBool, LongBool — для обеспечения совместимости с Windows.

Символьный тип (Char). Его значениями являются символы из множества ASCII (Американского стандартного кода для обмена информацией) — это 256 различных символов, упорядоченных определенным образом. Для русскоязычных пользователей часть символов и элементов псевдографики заменена русскими буквами (Модифицированная альтернативная кодировка ГОСТа). Если символьное значение имеет графическое представление, то оно изображается соответствующим знаком, заключенным в одинарные кавычки — апострофы (например: '*', 'x'). Если символ не имеет графического представления, то можно воспользоваться эквивалентной формой записи, состоящей из символа «#» (решетка) и целочисленного кода символа от 0 до 31.

Значениями переменных символьного типа являются произвольные символы: буквы, цифры, знаки препинания, знаки операций, скобки и т. д. Состав возможных символов зависит от языка программирования и его машинной реализации. Символы представляют собой упорядоченные величины. Способ их упорядочения зависит от машинной реализации языка, однако

всегда можно предполагать, что цифры упорядочены по возрастанию, а буквы — по алфавиту. Над упорядоченными значениями допустимы операции поиска значений, предшествующих данному значению и следующих за ним (соответственно **pred** и **succ** в языке Паскаль):

pred('B') = 'A';
succ ('B') = 'C';
pred('1') = '0';
succ ('1') = '2'.

Кроме того, над символьными типами допустимы операции сравнения. Результат сравнений зависит от порядка символов в используемой символьной таблице. Над символами может быть определена операция, результатом которой является порядковый номер символа в алфавите, используемом в данной ЭВМ (операция **ord** в языке Паскаль). Если **ch** — символьная переменная, значением которой является цифра, численное значение цифры **ch** можно получить как ord(ch) — ord('0'), например:

ord('9') – ord('0') = 9.

Арифметические операции над переменными этого типа обычно недопустимы.

В языке Паскаль для описания типов, определяемых пользователем, служит секция программы, обозначаемая ключевым словом **type.** Каждый элемент этой секции описывает структуру некоторого типа и определяет идентификатор, который позднее используется для обозначения этого типа. Структура определения типа:

type
имя_типа = тип;

Здесь *имя типа* — идентификатор, вводимый для обозначения нового типа, определяемого пользователем, а *тип* описывает новый тип данных.

Определяемые пользователем типы необязательно могут быть составными. Примером простого типа, определяемого пользователем, может служить перечисляемый тип. Так,

в программе на языке Паскаль может встретиться описание вида:

```
type
    Season = (Winter, Spring, Summer, Autumn);
```

Это описание вводит новый простой тип — *сезон*, переменные этого типа могут принимать одно из перечисленных значений — зима, весна, лето, осень.

Имена элементов перечисления в языке Паскаль должны быть различны и уникальны, т. е. отличны от имен других программных объектов. Перечисляемый тип в языке Паскаль является порядковым, т. е. к переменным этого типа применимы операции **pred** (предыдущий) и **succ** (последующий), а также операция **ord** (порядковый номер). Первому из членов перечисления присваивается порядковый номер 0, номера остальных последовательно возрастают на единицу слева направо.

Символьный, целый и булев типы относятся к порядковым типам: каждый элемент множества допустимых значений имеет свой порядковый номер. Переменные типа COMP могут иметь только целое значение, но, в отличие от других, этот тип не является порядковым.

Пользовательские типы данных — новые типы данных, определяемые программистом и требующие предварительного объявления.

Перечисляемый тип — задается перечислением тех значений, которые могут принимать переменные: **Color = (red, white, blue);**

Тип-диапазон — это подмножество своего базового типа, в качестве которого может выступать любой порядковый тип, кроме типа-диапазона. Задается границами своих значений внутри базового типа: **Date = 1..31; Month = (1..12);**

7.1.4. Описание меток, констант, типов и переменных

Описание меток:

Label M1, Met2;

Описание констант используется в программе для задания значений, которые не изменяются в процессе выполнения действий:

```
Const
name = 'Таня'; r2 = 4.15;
st = '********************';
```

Pi = 3.1416, Maxint = 32767, true и false — стандартные константы языка Паскаль.

Описание типов:

```
Type
Dip = 1..20;  //тип-диапазон.
Let = 'a' .. 'z';  //тип-диапазон.
Month=(jan,feb,mar);  //перечисляемый тип.
Ms=Array[1..4] of real;  //тип-массив.
```

Описание переменных. Переменные используются для записи значений, изменяющихся в программе. Выбор имен осуществляется так, чтобы были понятны смысл и назначение:

```
Var a, b, c: Integer;
s, name: Char;k,s:Let;
a12: Boolean; x1,x2: real;
Mes:Month;
Massiv1:Ms;
```

7.1.5. Операторы языка Паскаль

По функциональному назначению операторы подразделяются на группы: присваивания, ввода-вывода, управления, определения функций и процедур.

Операторы присваивания вызывают выполнение выражений и присваивание значения имени результата:

```
< Имя переменной>: = < выражение>;
C:= A/B;   D:= ( 32*S-(18+c)*F);  S:=S + Un;
```

Операторы ввода-вывода. В языке Паскаль нет специальных операторов ввода-вывода. Для обмена информацией с окружающим миром в программах используются стандартные (встроенные) процедуры. **Процедура** — это некоторая последовательность операторов языка Паскаль, к которой можно обратиться по имени. Стандартная процедура не нуждается

в предварительном описании, она доступна любой программе, в которой содержится обращение к ней. Название процедуры не является зарезервированным словом.

Для *ввода данных* используются следующие операторы обращения к встроенной стандартной процедуре ввода данных:

Read(A, B, C), где A, B, C — имена переменных, значения которых подлежат вводу для запоминания в оперативной памяти.

Readln(A,B,C) — после окончания ввода курсор перемещается к началу новой строки.

Readln — означает ожидание нажатия клавиши <*Enter*>. Обычно вставляется в текст программы перед последним <*End*> для сохранения на экране содержимого окна результатов выполнения программы.

Для *вывода данных* на экран монитора используется оператор обращения к стандартной процедуре вывода данных:

Write('a=', a:7:3, ' b=', b:6:3) — число после первого двоеточия означает количество позиций, выделяемых для вывода значения численной переменной, включая знак, целую часть, десятичную точку и цифры после запятой; цифра после второго двоеточия означает количество цифр после десятичной запятой. Лишние позиции будут заменены пробелами перед целой частью числа и нулями после дробной части. Например, в приведенном операторе ввода при **a=3,14744** на экран выведется: **a= 3.147**, при **b=–3,4** на экране выведется: **b=–3.400**.

Writeln ('Значение числа ПИ =' , pi) — запись значения числа **pi** на кране будет осуществлена в показательной форме: Значение числа Пи =3.14159265358979E+0000.

Writeln — означает пропуск одной строки и переход к началу новой строки.

Оператор безусловного перехода. Оператор **Goto** позволяет изменить стандартный последовательный порядок выполнения операторов и перейти к выполнению программы, начиная с оператора, помеченного меткой. Эта же метка должна быть указана в операторе Goto и в разделе описания меток. Метки могут быть как целым числом, так и обычным идентификатором. Метка отделяется от помеченного оператора двоеточием.

Составной оператор — это последовательность операторов программы, заключенная в операторные скобки: зарезервированные слова **Begin .....End**.

Условный оператор позволяет проверить некоторое условие и в зависимости от результатов проверки выполнить то или иное действие. Условный оператор — это средство ветвления вычислительного процесса.

Структура условного оператора:

If <условие> Then <оператор 1> Else <оператор 2>, где If, Then, Else — зарезервированные слова (*если*, *то*, *иначе*); <условие> — произвольное выражение логического типа; <оператор 1>, <оператор 2> — любые операторы языка Паскаль.

Следует помнить, что любой из операторов 1 или 2 может быть также условным, причем необязательно в нем будет присутствовать часть Else <оператор 2>. Подобная неоднозначность решается следующим образом: любая встретившаяся часть Else соответствует ближайшей к ней «сверху» части Then условного оператора.

Если требуется выполнить несколько операторов в условном операторе, то их необходимо заключить в операторные скобки, в противном случае в условном операторе выполняется только первый после ключевого слова Then или Else оператор.

Операторы цикла (операторы повторений). В языке Паскаль имеется три различных оператора, с помощью которых можно запрограммировать повторяющиеся фрагменты программ:

— ***арифметический оператор цикла***:

For <параметр цикла>:= <начальное значение> To <конечное значение> Do <оператор>, где параметр цикла — переменная типа Integer. При выполнении оператора For циклически повторяется:

проверка условия **параметр цикла <= конечного значения**; если условие не выполнено, оператор For завершает свою работу;

выполнение оператора тела цикла;

наращивание параметра цикла на единицу.

Существует другая форма оператора For:

For <параметр цикла>:= <начальное значение> Downto <конечное значение> Do <оператор> (шаг наращивания параметра цикла при этом равен (–1));

— ***оператор цикла While с предпроверкой условия***. Структура оператора цикла While:

While <условие> Do <тело цикла> (пока условие истинно, выполнять тело цикла). Если тело цикла состоит из нескольких операторов, их необходимо заключать в операторные скобки;

— ***оператор цикла Repeat....Until с постпроверкой условия***. Структура оператора цикла:

Repeat <тело цикла> Until <условие> (повторять выполнение операторов, входящих в тело цикла, до тех пор, пока условие не станет истинным);

— ***оператор выбора***. Позволяет выбрать одно из нескольких возможных продолжений программы. Параметром, по которому осуществляется выбор, служит ключ выбора: выражение любого порядкового типа. Структура оператора выбора:

Case <ключ выбора> Of <список выбора>
Else <оператор> End

(<список выбора> — конструкции вида <константа выбора>: <оператор>; константа выбора — константа того же типа, что и выражение <ключ выбора>).

Оператор выбора работает следующим образом. Вначале вычисляется значение выражения <ключ выбора>, а затем в последовательности операторов <список выбора> отыскивается такой, которому предшествует константа, равная вычисленному значению. Найденный оператор выполняется, оператор выбора завершает свою работу. Если в списке выбора не найдена константа, соответствующая вычисленному значению ключа выбора, управление передается оператору, стоящему за словом Else. Часть Else может отсутствовать.

7.1.6. Операции и выражения

Основной блок программы состоит из последовательности операторов. С помощью операторов описываются действия над данными, которые необходимо выполнить для нахождения результата решения поставленной задачи.

Для определения действий, которые в математике обычно описываются формулами, в программировании служат выражения.

Выражения — это конструкции, которые могут включать в себя константы, переменные, стандартные функции, пользовательские функции и числа, соединенные между собой знаками операций и парами круглых скобок. Выражения состоят из

операндов и операций, записываются в одну строку (ленточная запись) и всегда имеют конечное значение определенного типа (табл. 7.2). Все переменные, входящие в выражение, должны иметь конкретное значение.

Таблица 7.2

Арифметические операции

Операция	Действие	Тип операнда	Тип результата
+	Сложение	integer, real	integer, real
—	Вычитание	integer, real	integer, real
*	Умножение	integer, real	integer, real
/	Деление	integer, real	real
Div	Деление нацело	integer	integer
Mod	Остаток от деления	integer	integer

По количеству операндов операции делятся на унарные (единичные) и бинарные (двойные).

Пример.

C:= A Mod B; при A:= 34 и B:= 9 C:= 7;
C:= A Div B; при A:= 34 и B:= 9 C:= 3;

Булевы, или **логические**, **выражения** включают в себя переменные и простые логические операции: =, >=, <=, <, >, < > (например: A<=B).

Простые булевы выражения могут объединяться в сложные с помощью следующих логических операций: логическое сложение — Or, логическое умножение — And, отрицание — Not и исключающее ИЛИ — Xor. (табл. 7.3 и 7.4)

Таблица 7.3

Логическая операция отрицания

a	«не» Not a
False	True
True	False

Таблица 7.4

Логические операции

a	b	И a And b	ИЛИ a Or b	Исключающее ИЛИ a Xor b
True	True	True	True	False
False	True	False	True	True
False	False	False	False	False

Пример.

(X=0) And (Y=0) при X=0 и Y=4 (выражение имеет значение False).

Фрагмент программы:

```
Var I: Integer;
    R: Boolean;
Begin
    I:= 10 Div 5;
    R:= I=2;
```

В результате выполнения этого фрагмента программы переменной **R** присваивается значение True.

Таблица 7.5

Приоритетность операций

Приоритет и описание	Операции в языке Паскаль
1. Вычисления в круглых скобках	( )
2. Вычисление значений функций	Функции
3. Отрицание и получение адреса объекта	Not , @
4. Умножение, деление, логическое умножение, сдвиги операндов	*, /, Div, Mod, And, Shl, Shr
5. Сложение, вычитание, логическое сложение, исключающее ИЛИ	+, —, Or, Xor
6. Простые булевы операции, принадлежность к множеству	=, < >, <, >, <=, >=, In

7.1.7. Процедуры, функции и рекурсии языка Паскаль

При создании программы для решения сложной задачи выполняется разделение (декомпозиция) этой задачи на подзадачи, а подзадач — на еще меньшие подзадачи и т. д.

Турбо Паскаль имеет различные средства для деления программы на части. На верхнем уровне деления (больших задач) — это модули, на нижнем уровне (элементарных подзадач) — это подпрограммы, которые в Турбо Паскаль могут быть двух видов: процедуры и функции.

Подпрограммой в языке Турбо Паскаль называется особым образом оформленный фрагмент программы, имеющий собственное имя. Упоминание этого имени приводит к активизации подпрограммы и называется ее **вызовом**. Сразу после активизации подпрограммы начинают выполняться входящие в нее операторы. После выполнения последнего из них управление возвращается обратно в основную программу, и выполняются операторы, стоящие непосредственно за оператором вызова подпрограммы. Язык Паскаль имеет богатую библиотеку программных заготовок (модулей), существенно облегчающих разработку прикладных программ. Такими модулями являются SYSTEM, DOS, CRT, PRINTER, GRAPH, OVERLAY, ТУРБО3 и GRAPH3. Модули GRAPH, ТУРБО3 и GRAPH3 выделены в отдельные TPU-файлы, а остальные входят в состав библиотечного файла ТУРБО.TPL. Лишь один модуль — SYSTEM — подключается к любой программе автоматически, все остальные становятся доступны только после указания их имен в списке, следующем за словом **USES.**

Процедуры и функции — важнейший элемент любого языка программирования. Обычно эти термины используются в контексте языков программирования высокого уровня, в языке Ассемблер и машинном языке обычно применяется общий термин «подпрограмма», в языке Фортран — термины «подпрограмма» и «подпрограмма-функция».

Первоначально процедуры вводились в качестве средства сокращенной записи повторяющихся фрагментов программы. Однако в настоящее время им придается не меньшая важность в качестве средства структурирования программы. Механизм процедур позволяет при разработке программы методом после-

довательного приближения использовать метод «черного ящика». Каждое действие может быть описано в виде обращения к отдельной процедуре. При этом можно игнорировать, как именно определяются действия, реализуемые в процедуре, следует только определить, какие именно данные нужно предоставить ей для выполнения возложенных на нее действий и какие результаты от нее необходимо получить. Совокупность этих данных и результатов называется **параметрами** процедуры. Последовательное уточнение действий состоит в определении процедур, которые вызывались на предыдущем этапе.

При определении процедуры указываются ее имя, список необходимых ей параметров и блок, содержащий описание действий, реализуемых процедурой. Список параметров, приведенных в заголовке, специфицирует интерфейс процедуры с внешней средой. В заголовке указывается строгое описание количества и типа параметров, поэтому элементы этого списка называются **формальными параметрами**. Само описание процедуры помещается в описательной части программы. Тело процедуры может содержать локальные переменные, используемые в процессе вычислений.

Обращение к процедуре содержит ссылку на ее имя, снабженную списком констант и переменных, которые подлежат обработке при данном вызове, или *фактические* параметры. Количество и типы фактических параметров должны соответствовать количеству и типам формальных параметров. Выполнение процедуры происходит «вне очереди», т. е. управление при вызове передается на первый выполняемый оператор процедуры, а по ее окончании возвращается на ту команду вызывающей программы, которая следует за оператором вызова.

Функция — это специальная форма процедуры, предназначенная для вычисления по заданным параметрам одного-единственного значения. Для вызова функции, в отличие от процедуры, ее имя нужно записать в виде компоненты выражения, например: $y:= F(x)$, где F — имя функции; x — фактический параметр. Вычисленное значение функции связывается с именем функции, что позволяет обойтись без промежуточных параметров, передающих вычисленное значение. С операционной точки зрения функция в языке программирования аналогична функции в математике.

Описание процедуры на языке Паскаль:

procedure Имя (Список формальных параметров);блок;

оператор обращения к процедуре:

Имя (Список фактических параметров);

описание функции:

function Имя (Список формальных параметров) : тип; блок;

обращение к функции — компонента выражения;

Имя (Список фактических параметров).

В теле функции должен содержаться один или несколько операторов присваивания значения идентификатору функции:

Имя:= exp;

Пример:

```
Var a:real;
    procedure sum(x,y : real; var z : real);
Begin
    z: = x+y {var нужно, чтобы возвратить значение}
End;
Begin
    sum (3.1, 2.2, s);
    writeln(s);
End.
```

Тот же результат достигается с помощью функции:

```
function sum (x,y: real):real;
Begin
    writeln(sum(3.1,2.2);
End.
```

Стандартные функции и процедуры языка Паскаль:

Abs(X) — вычисляется абсолютное значение X.

Exp(X) — E возводится в степень X.

Ln(X) — вычисляется натуральный логарифм X.

Sqr(X) — X возводится в квадрат.

Sqrt(X) — вычисляется квадратный корень из X.

Sin(X), Cos(X), Arctan(X) — тригонометрические функции (аргумент задается всегда в радианах).

Trunc(X) — определяет целую часть числа, тип результата Longint.

Round(X) — округляет до целого.

Chr(I) (I — целое число) — определяет символ, порядковый номер которого равен I.

Ord(X) (X — порядковый) — определяет порядковый номер символа в наборе символов.

Pred(X) (X — порядковый) — находит предшествующий элемент.

Succ(X) (X — порядковый) — находит последующий элемент.

Odd(X) (X — целого типа) — определяет четность числа: если X — нечетный, то результат принимает значение True, если четный — False.

Eoln(X) (X — файловая переменная) — результат принимает значение True, если при чтении текстового файла достигнут конец текущей строки. В остальных случаях результат равен False.

Eof(X) (X — файловая переменная) — результат принимает значение True, если при чтении текстового файла достигнут конец файла. В остальных случаях результат равен False.

Dec(X[,i]) — уменьшает значение X на i, при отсутствии i — на 1.

Inc(X[,i]) — увеличивает значение X на i, при отсутствии i — на 1.

Frac(X) — определяет дробную часть аргумента.

Int(X) — определяет целую часть аргумента. Тип результата — Real.

Random (X) — равномерное псевдослучайное число 0 <= I < X, при отсутствии X — интервал чисел от 0 до 1.

Математические функции, не представленные в языке Паскаль в явном виде:

десятичный логарифм: **Lg (*X*) = Ln(*X*)/Ln(10)**;

возведение в степень: $Y = M^n \rightarrow$ **Y := Exp(*n**Ln(*M*))**;

$A = 16^5$ **A := Exp(5*Ln(16))**;

тангенс угла: **Tg (*X*) = Sin (*X*)/Cos (*X*)**;

котангенс угла: **Ctg (*X*) = Cos (*X*)/Sin (*X*)**;

секанс угла: **Sc (*X*) = 1/Cos (*X*)**;

косеканс угла: **Csc (*X*) = 1/Sin (*X*)**;

арксинус числа: **Arcsin (*X*) = ArcTan (*X*/Sqrt(1 – *X**X))**;

арккотангенс числа: **Arcctg(*X*) = Pi/*2* — ArcTan (*X*)**;

арккосинус числа: **Arccos (*X*) = Pi/*2* — ArcTan (*X*/Sqrt(1 – *X**X))**.

Тригонометрические функции. Параметр тригонометрических функций всегда задается в радианах. Для перевода из градусов в радианы и наоборот используются соотношения:

1 рад = 180°/Pi = 57°17′45″; 1° = Pi/180 рад = 0,0174 рад.

Нестандартные процедуры и функции. В большинстве случаев некоторые специфичные для данной прикладной программы действия не находят прямых аналогов в библиотеках языка Турбо Паскаль, и тогда программисту приходится разрабатывать свои, нестандартные процедуры и функции.

Нестандартные процедуры и функции необходимо описать, чтобы компилятор мог установить связь между оператором вызова и теми действиями, которые предусмотрены в процедуре (функции). Описание подпрограммы помещается в разделе описаний и внешне выглядит как программа, но вместо заголовка программы фигурирует заголовок процедуры или функции.

Функции представляют собой группу операторов, в результате выполнения которых вычисляется одно значение, присваиваемое имени функции. В заголовке функции за ключевым словом **Function** указывается ее имя, а в скобках — список параметров со своими описаниями. В заголовке определяется тип значения, присваиваемый функции. Как правило, окончательный результат присваивается функции в конце тела функции. Общая структура записи функции имеет вид:

Function F(q_1:T_1,q_2:T_2, ...):T;

<Раздел описания локальных меток, констант, типов и переменных>

<Раздел описания внутренних процедур и функций>

Begin

S1;... S2; <Операторы>

F:= <Обязательный оператор, который присваивает имени функции значение результата>

End;

где F — имя функции; q_i — имена формальных параметров; T_i — типы формальных параметров; T — тип результата; S_i — операторы тела функции.

Обращение к функции осуществляется в правой части оператора присваивания, при этом в выражении записываются имя функции и фактические параметры в виде:

X:= F(b_1, b_2,...),

где F — имя функции; b_i — фактические параметры. После выполнения функции вычисленное значение присваивается имени функции и передается в выражение.

Рекурсия. Если процедура или функция в ходе выполнения вызывает саму себя, то она является *рекурсивной*. Использование рекурсии позволяет легко запрограммировать вычисления по рекуррентным формулам. Например, для вычисления факториала существует рекуррентная формула: **0!=1**; для всех **n>0 n!=n*(n-1)!**

В случае использования рекурсии функция вычисления факториала имеет следующий вид:

```
Function Fact(k:byte):longint;
Begin
    if k=1 then Fact:=1
    else
    Fact:=k*Fact(k-1);
End.
```

Процедуры используются в тех случаях, когда необходимо в подпрограмме получить несколько результатов или выполнить действие над параметрами. Описание процедуры включает в себя заголовок процедуры, разделы описаний, тело процедуры. В заголовке после ключевого слова **Procedure** указывается имя процедуры, в скобках — список формальных параметров со своими описаниями. Эти параметры используются только в теле подпрограммы и локальны по отношению к ней. Общая структура записи процедуры имеет следующий вид:

```
Procedure P(r1,r2:T1;var q1:T2;q2:T3;...);
```

<Разделы определений и описаний локальных параметров и подпрограмм)

```
Begin
    S1;.... S2;
End;
```

где P — имя процедуры; r_i и q_i — имена формальных параметров, причем r_i — параметры-значения, q_i — параметры-пере-

менные; T_i — типы формальных параметров; S_i — операторы процедуры.

Обращение к процедуре осуществляется оператором процедуры, в котором записываются имя процедуры и ее фактические параметры $P(b_1, b_2, ...)$, где b_i — фактические параметры, которые соответствуют формальным по количеству, типу и месту расположения.

Формальные параметры можно указывать в любом порядке, однако при обращении к подпрограмме фактические параметры необходимо записывать в той же последовательности, что и формальные.

В тех случаях, когда в процедуре и главной программе используются одни и те же имена параметров (процедура связана с главной программой посредством глобальных переменных), процедуру можно организовать без параметров.

7.1.8. Структурированные типы данных

Массивы. В языке Турбо Паскаль могут использоваться также объекты, содержащие множество однотипных элементов. Это *массивы* — формальное объединение нескольких однотипных элементов (чисел, символов, строк и т. п.), рассматриваемых как единое целое. Например, результаты многократных замеров температуры воздуха в течение одного года удобно рассматривать как совокупность вещественных чисел, объединенных в один сложный объект — массив измерений.

При описании массива за ключевым словом **Array** в квадратных скобках указывается тип-диапазон, а за ключевым словом **of** следует тип элементов. Тип-диапазон задается левой и правой границами изменения индекса массива.

Var a: Array[1..10,1..5] of Real; {Двухмерный массив, состоящий из 50 элементов (10 строк, 5 столбцов) вещественного типа};

b: Array[1..50] of Char;{Одномерный массив из 50 символов};

c: Array [-3..4] of Boolean;{Одномерный массив из 8 элементов с порядковыми номерами от –3 до 4, тип элементов — логический}.

При компиляции программы в оперативной памяти резервируется объем памяти, необходимый для размещения всего

объявленного массива. Если количество элементов массива заранее не известно или может изменяться, то необходимо зарезервировать объем памяти для размещения максимально необходимого количества элементов.

Обращение к определенному элементу массива в программе осуществляется с помощью индекса — целого числа в квадратных скобках, следующего за именем массива:

```
b[17]:= 'F'; C[-2]:= a[1,1] > a[2,2];
```

Массивы могут быть заданы в разделе объявления констант. Двухмерные и многомерные массивы объявляются обычно с использованием раздела Type:

```
Const n= 10; m= 15; p= 20;
    MA:Array[1..2, 1..3] of Integer= ((3,5,6), (2,1,7));
    {Значения постоянного массива}
Type ms= Array[1..n, 1..m] of Real;
Var a1: Array[1..p] of ms;
    a2,a3: ms;
```

В качестве примера рассмотрим некоторые операции с массивами:

1. Ввод с клавиатуры двухмерного массива A[N,M]:

```
For i:= 1 to n do
For j:= 1 to m do
    Begin
Writeln('Введите  a[',i,',',j,']')
Readln (a[i,j]);
    End;
```

2. Вывод двухмерного массива A[N,M] на печать:

```
For i:= 1 to n do
    Begin
For j:= 1 to m do
Write (a[i,j]:6:3);
Writeln;
    End;
```

3. Формирование двух массивов с помощью датчика случайных чисел:

```
Randomize;
For i:= 1 to k do
    Begin
Y[i]:= Random; {Для задания массива Y 0 < Yi < 1}
X[i]:= Random(100); {Для задания массива X 0 < Xi < 100}
    End.
```

Символы и строки символов. Каждый из символов имеет свой уникальный номер — *код*. Соответствие кода и внешнего вида символа называется **кодовой таблицей**, оно зависит от страны (языка), от операционной системы, от устройства, на которое символ выводится. Для преобразования кода в символ в языке Паскаль существует функция **Chr(код),** для преобразования символа в код — функция **Ord(символ**). Например, в результате выполнения оператора **WriteLn(Chr(68));** на экране появится символ D, так как его код — 68, а при выполнении **WriteLn(Ord('D'));** — число 68, так как именно ему соответствует символ D.

Переменная для хранения одного символа имеет тип **char**. Символьную константу в программе можно указать двумя способами: 'D' или #68. Первый способ удобнее использовать, когда символ легко ввести с клавиатуры.

Обрабатывать текстовую информацию удобнее крупными частями**.** Строка символов в языке Паскаль — это последовательность символов длиной от 0 до 255, ее тип имеет название **String**. Фактически это массив, нулевой элемент которого содержит логическую длину строки. Если там находится #0, то считается, что строка пуста, если #20, значит, в строке 20 символов (тип всех, в том числе и нулевого, элементов — char, поэтому перед 0 и 20 стоит знак #). Физическая же длина строки задается при написании программы (по умолчанию — 255 байт под содержимое плюс 1 байт, отвечающий за длину), например:

```
var
    s1: String;
    s2: String[10];
```

переменная s1 будет занимать в памяти 256 байт, а s2 — 11.

Доступ к каждому элементу аналогичен доступу к элементам массива: **имя_строки[значение_индекса]**.

Процедуры для работы со строками (вызываются как отдельные операторы**):**

Delete(<Строковая_переменная>, <Позиция>, <Количество_символов>) — удаляет из строки указанное количество символов, начиная с указанной позиции.

Insert(<Исходная_строка>, <Строковая_переменная>, <Позиция>) — вставляет последовательность символов в строковую переменную.

Str(<Число>, <Строковая_переменная>) — преобразует число в строку. После числа может стоять спецификация формата. Например, после выполнения оператора **Str(f:7:3,s);** при f = –1,8 в строке s будет находиться: ' —1.800'. **(Val(<Строка>, <Численная_переменная>, <Код_результата>)** — если строка содержит число, оно будет помещено в численную переменную и переменная «Код результата» будет равна 0. Если же при преобразовании произойдет ошибка (будет обнаружен недопустимый в числе символ, например буква), позиция ошибочного символа помещается в переменную «Код результата». Похожие действия автоматически выполняются при вводе чисел с клавиатуры (ведь вводятся символы, а результатом должно стать число), но тогда при обнаружении ошибочного символа программа аварийно завершается.

Функции для работы со строками (должны быть частью выражений соответствующих типов):

Length(<строка>) — возвращает целое число — логическую длину строки.

Copy(<Строка>, <Позиция>, <Размер>) — возвращает подстроку из указанной строки.

Pos(<Искомая_подстрока>, <Строка>) — возвращает число — позицию первого вхождения подстроки в строку — или 0, если строка не содержит такой последовательности символов где-то внутри себя.

Запись (структура данных). Тип-запись включает ряд компонент, называемых полями, которые могут быть разных типов.

Формат объявления типа-записи:

```
Type
    <имя типа> = record
        < поле 1>: тип 1;
        ..............
        <поле  N>: тип M
    End;
Type
    Complex =  record
        Re, Im: real;
    End;
    Data = record
        Year: Integer;
```

```
        Month: 1..12;
        Day: 1..31
    End;
Var
    X,Y,Z: complex;
    Spisok: array [1..100] of Data;
Const
    Birthday: Data= (Year:1971; Month:12;
    Day:9);
```

Доступ к полям записи осуществляется указанием имени переменной (константы) и имени поля через точку, например:

X.Re, Birthday.Day, Spisok[99].Year и т. д.

Чтобы упростить доступ к полям записи, используется оператор присоединения **With:**

With <переменная>DO<оператор>

Множества представляют собой ограниченный набор однотипных логически связанных друг с другом объектов. Характер связей между объектами лишь подразумевается программистом и никак не контролируется языком Турбо Паскаль. Количество элементов, входящих в множество, может меняться в пределах от 0 до 256 (возможно пустое множество). Именно непостоянство количества элементов отличает множества от массивов и записей. Элементами множества могут быть значения скалярных типов **byte** и **char**.

Описание типа **множество** имеет следующий вид:

<имя типа>= SET OF <баз. тип>, где <баз. тип> — базовый тип элементов множества, в качестве которого может использоваться также порядковый тип.

Пример определения и задания множеств:

```
Type
    dg1 = Set of '0'.. '9';
    mn2=Set of  Byte;
Var
    s1: dg1;
    s2:mn2;
Begin
    s1:= ['1', '2', '3'];
    s2:= [0..3,6];
End.
```

Два множества считаются эквивалентными, когда все их элементы одинаковы, причем порядок следования элементов безразличен. S1 и S2 — эквивалентны, S3 включено в S2, но не эквивалентно ему.

Мощностью множества называется количество неповторяющихся элементов, входящих в него. Над множествами определены следующие операции:

*	Пересечение множеств (рис.7.5, *а*). Результат содержит элементы, общие для обоих множеств. Математическое обозначение: S1∩S2, логическое умножение
+	Объединение множеств (рис. 7.5, *б*). Результат содержит все элементы первого множества, дополненные недостающими элементами из второго множества. Математическое обозначение: S1∪S2, логическое сложение
–	Разность множеств (рис. 7.5, *в*). Математическое обозначение: S3=S2\S1, дополнение
=	Проверка эквивалентности. Результат True, если множества эквивалентны
<>	Проверка неэквивалентности
<= >=	Проверка вхождения. Математическое обозначение: S1 ⊂ S2, S1содержится в S2
IN	Проверка принадлежности. Математическое обозначение: P ∈ S1, P является элементом множества S1

Ввод элементов множества с клавиатуры:

```
x1:=[ ]; {Задание пустого множества}
For i:=1 to 10 do
Begin
    Readln (a);
    x1:= x1+[a];
End;
```

Вывод множества на экран:

```
For i:= 1 to N do
If i in x1 then Write(i:4);
```

Файлы — это средство связи с внешними источниками, приемниками и носителями информации. Традиционно под

Пересечение
а

Объединение
б

Разность
в

Рис. 7.5. Графическая интерпретация операций над множествами

файлом понимается поименованная совокупность данных на внешнем носителе, однако в языке Турбо Паскаль файлом считается также любое внешнее устройство (называемое логическим), по своему назначению являющееся источником или приемником информации, например клавиатура, дисплей, принтер и т. д.

С двумя файлами INPUT (текстовый файл, вводимый с клавиатуры) и OUTPUT (текстовый файл, выводимый на экран монитора и содержащий результаты работы программы) мы уже знакомы. Результаты работы программы можно сохранить и отправить в другие файлы, записанные на диски. В качестве источника данных также могут использоваться файлы, записанные заранее на диски. Каждый такой файл должен иметь имя, а его тип должен быть объявлен в разделе VAR. Одновременно могут быть открыты несколько файлов; в ходе выполнения одной программы один и тот же файл может быть открыт для записи и впоследствии установлен на чтение. **До начала операции ввода-вывода конкретному внешнему файлу должна быть поставлена в соответствие переменная файлового типа**. Затем файл необходимо открыть для чтения информации, для записи информации или для чтения и записи совместно.

В языке Турбо Паскаль определены три типа файлов: типизированные, нетипизированные и текстовые. В общем случае переменные типа File могут объявляться следующим образом:

Var <Имя файловой переменной >**: File [of** < Тип данных элемента>**];**

Если зарезервированное слово **of** и параметр *тип* опущены, объявляемый файл является нетипизированным. Типизирован-

ные и нетипизированные файлы могут эксплуатироваться в режиме как последовательного, так и произвольного (когда допускается выборочное обращение к конкретным записям, которые задаются их именами) доступа.

Формат объявления текстовых файлов, используемых только в режиме последовательного доступа:

Var <Имя файловой переменной >: Text;

Примеры:

```
Type
    FF = Record
        Name: string[10];
        Tele: word;
    End;
Var
    Txtfile: text;
    Spisok: File of FF;
    Ss1: File;
```

Стандартные средства обработки файлов. Процедура **Assign(F1, 'ttt.pas')** служит для связи файловой переменной F1 с некоторым файлом ttt.pas, расположенным в текущем каталоге. В общем случае имя типа должно быть написано в соответствии с правилами MS DOS, может включать путь и не должно превышать 79 символов.

Процедура **Reset(F1)** открывает существующий файл данных, имя которого перед этим было связано с помощью процедуры **Assign** с некоторой файловой переменной, указанной в процедуре Reset как параметр.

Если возможности открыть файл ttt.pas нет, то возникает ошибочная ситуация, подавить которую при выполнении можно, блокировав директивой компилятора **{$I–}** проверку ошибок ввода-вывода.

Процедура **Rewrite(F1)** создает новый пустой файл, присваивает ему имя, заданное процедурой **Assign**, и открывает его для записи или чтения. Если файл существует, его содержимое стирается, а сам файл открывается заново.

Процедура **Append(F1)**, где F1 — имя файловой переменной, позволяет добавлять новые записи: строки в файлы, объявленные в программе как текстовые.

Процедура **Close(F1)** закрывает открытый ранее файл, связанный с указанной в качестве параметра файловой переменной.

Процедуры **Rename(F1, New)** и **Erase(F1)** предназначены для того, чтобы переименовывать или стирать существующие файлы с диска.

Функция **Eof(F1)** позволяет в процессе считывания информации проверять, достигнут ли конец файла, т. е. находится ли указатель файла за последним элементом или нет. Например, цикл **While not Eof(F1) Do Read(F1, X)** будет выполнять считывание порций данных из файла, связанного файловой переменной F1, до тех пор пока файловый указатель не достигнет конца открытого логического файла. Такая операция необходима при дополнении содержимого типизированных файлов, так как в противном случае данные, которые должны быть приписаны к концу файла, запишутся поверх уже существующих данных.

Функция **SeekEof(F1)** принимает значение True, если указатель установлен на признак конца файла; во всех остальных случаях возвращается значение False.

Функция **IoResult** предназначена для поиска ошибок, возникающих при работе с файлами. В следующем фрагменте программы выполняется проверка корректности завершения ввода-вывода. Для этого блокируются средства контроля компилятора за ошибками ввода-вывода, в противном случае неудачная попытка открытия файла приведет к прекращению работы программы:

```
{$I-} {Отключение контроля ошибок ввода-вывода}
Reset(F1);
If IoResult <> 0 Then Begin
        Rewrite(F1); {Создание нового файла}
        If IoResult<>0 Then Write('Ошибка при создании
        файла');
    End;
Else {Если файл существует}
While not Eof(F1) do Read(F1, X);
{Позиционирование указателя на конец файла}
{$I+} {Включение контроля ошибок ввода-вывода}
```

ПРАКТИЧЕСКИЕ РАБОТЫ

Работа № 1. Полный цикл работы с программой

Цель работы: изучить приемы работы в среде Турбо Паскаль.

***Задание*:** вычислить корни квадратного уравнения $axx + + bx + c = 0$.

Методика выполнения работы:

1. Запустите интегрированную среду Паскаль (двойной щелчок на ярлыке Турбо Паскаль).

2. Смените при необходимости рабочий каталог, установленный по умолчанию:

***<File> → <Change dir>*.**

3. Откройте новое окно для записи текста программы: ***<File>→ <New>***. Наберите в окне редактора следующую исходную программу, написанную на алгоритмическом языке Паскаль, предназначенную для вычисления корней квадратного уравнения.

```
Program kwur; (*Имя программы*)
    Uses Crt; {Использование библиотечного модуля CRT}
Var a, b, c, d, w, z, x1, x2: Real;
    {Объявление переменных}
        otvet: Char;
Begin {Начало основного блока программы}
    Clrscr; {Очистка экрана}
    WriteLn ('Вы работаете с программой вычисления корней ');
    WriteLn ('Квадратного уравнения общего вида
    axx+bx+c=0 ');
    Repeat {Начало цикла с постусловием}
    Write('Введите значение a');
    ReadLn (a); {Ввод коэффициента a}
    If a = 0 Then Writeln ('Уравнение не квадратное')
    Else
Begin
```

```
        Write ('Введите значение b=');
        ReadLn (b);
        Write ('Введите значение c=');
        ReadLn (c);
        D:= b*b — 4*a*c; {Вычисление дискриминанта}
        Z:= —b/(2*a);
        W:= sqrt(abs(d))/(2*a);
        WriteLn ('При a=',a:5:2,' b=',b:5:2,' c=',c:5:2);
        If D<0 Then
                Begin
                    WriteLn ('X1= ', Z:6:3, ' + j*',w:5:3);
                    WriteLn ('X2= ', Z:6:3, ' - j*',w:5:3);
                End
            Else
                Begin
                    x1:= z + w;
                    x2:= z — w;
                    WriteLn ('x1=', x1:5:2,' x2=', x2:5:2);
                End
    End;
    WriteLn ('Желаете продолжить работу? (y/n)');
    Readln (otvet);
    Until otvet <> 'y';{Повторять, пока  переменная otvet
        примет значение, отличное от y}
    WriteLn ('Спасибо! До свидания!');
End.
```

4. Используя команду **Save** пункта меню **File**, запишите набранную вами программу на диск под именем kwur1. Расширение указывать необязательно. Откомпилируйте исходную программу: <***Alt***> + <***F9***>**.** Исправьте допущенные ошибки. Запустите программу на выполнение: <***Ctrl***> + <***F9***>**.**

5. Запишите в тетрадь 3 уравнения для тестирования программы. Решите их на калькуляторе и сравните результаты ваших вычислений с результатами решения уравнений на ЭВМ.

6. Окончание работы — выход из среды Турбо Паскаль: <***ALT***> + <***X***>.

Работа № 2. Выполнение вычислительных операций

Цель работы: изучить приемы выполнения вычислительных операций в среде Турбо Паскаль.

Задание: вычислить значения $p = \lg(a + x^2) + \sin^2\left(\frac{z}{a}\right)$, $y =$

$= a\,\mathrm{tg}^3(a + x^2) + \sqrt{\frac{z^2}{a^2 + x^2}}$ при $a = 0{,}59$; $z = -4{,}8$; $x = 2{,}1$.

Методика выполнения работы:

```
Program pr_2;
    Uses Crt;
    Const a = 0.59; z = ?4.8; x = 2.1;
Var y, p, c, t: real;
Begin
    Clrscr;
    c:= Sin(x*x) /Cos (x*x);
    y:= a*c*Sqr(c)+Sqrt(z*z/(a*a+x*x));
    p:= (Ln(a+x*x))/Ln(10)+Sqr(Sin(z/a));
    t:= Exp(a*Ln(x));
    Writeln('При a=', a:4:2,'z=', z:4:1,' x=',
        x: 3:1);
    Writeln ('p=', p:9:4,'y=', y:9:4,' t=',
        t: 9:4);
    Readln
End.
```

Результаты вычислений:
при $a = 0.59$ $z = -4.8$; $x = 2.1$;
$p = 1.6217$; $y = 21.6350$; $t = 1.5492$.

Работа № 3. Пример использования арифметического цикла

Цель работы: изучить правила использования арифметического цикла в среде Турбо Паскаль.

Задание: вывести на экран таблицу вычислений значений y при изменении x от $a = -5$ до $b = +5$ с шагом $h = 0.5$ по соответствующим формулам:

$$y = \begin{cases} \dfrac{1}{x+2} & \text{при } x < -2 \\ 2 * x^3 & \text{при } -2 \leqslant x \leqslant +2 \\ \lg(x) + e^X & \text{при } x > 2 \end{cases}$$

Определение числа повторов: $N = (b - a)/h + 1 = (5 - (-5))/$ $/0.5 + 1 = 21$.

Методика выполнения работы:

```
Program pr_3a;
    Uses Crt; {Подключение библиотечного модуля CRT}
Var i: Integer; x,y: Real; {Описание переменных}
Begin
    Clrscr;
    Writeln (' Таблица'); {Вывод на экран шапки таблицы}
    Writeln(' x y ');
        x:= —5;
    For i: =1 To 21 Do {Арифметический цикл}
        Begin
            If x < –2 Then y:= 1/(x+2)
            Else If (x >= ?2) And (x<=2)
                Then y:= 2*Exp(3*Ln (x))
                Else y:=Ln(x)/Ln(10)+Exp(x);
            Writeln(x:7:2,y:10:4);
                x:=x+0.5;
        End;
    Readln;
End.
```

Работа № 4. Использование оператора цикла while

Цель работы: изучить правила использования оператора цикла while в среде Турбо Паскаль.

Задание: вычислить сумму членов бесконечного ряда с заданной точностью **eps = 10^{-4}** при ***x* = 5**.

$$S = 1 - \frac{\lg(x)}{2!} + \frac{\lg(2x)}{4!} - \frac{\lg(3x)}{6!} + \ldots (-1)^n \frac{\lg(nx)}{(2n)!}.$$

На экран вывести значение суммы, число членов ряда, вошедших в сумму, и последний член ряда, вошедший в сумму. Сравнить полученное на ЭВМ значение суммы членов ряда со значением, вычисленным по аналитическим формулам.

Методика выполнения работы:

```
Program pr_4;
    Uses Crt;
Var s, eps, x, Un, Uk, y: Real;
    i:Integer;
Begin
    Clrscr;
        x:= 5;
        eps:= 1e–5;
        s:= 1;
        i:= 1;
        y:= –1*2;
    Un:= (Ln(x)/Ln(10))/y;
    While Abs(Un) > eps Do
    Begin
        s:= s+Un;
        Uk:= Un;
        i:= i+1;
        y:= -y*(2*i-1)*(2*1);
        Un:=Ln(i*x)/Ln(10)/y;
    End;
    Writeln('s=', s:9:6,' n=', i,' U=', Uk: 9: 6);
    Readln;
End.
```

Результаты вычислений:
***S* = 0.724776**; ***n* = 6**; ***U* = 0.000046**.

Работа № 5. Использование итерационных операторов цикла

Цель работы: изучить правила использования итерационных операторов цикла в среде Турбо Паскаль.

Задание: составить программу для вычисления корней уравнения $x^4 - 3x^2 - 8x = 29$ на отрезке [1,9 : 2.0] с точностью eps = $= 10^{-4}$ методом простой итерации. Примем за начальное значение $x_0 = 1.92$. Вывести на экран корень уравнения до пятого знака и число итераций.

Методика выполнения работы:

заменим данное уравнение уравнением вида $x = f(x)$, равносильным данному: $x = \sqrt[4]{29 + 3x^2 + 8x}$.

Программа с использованием оператора цикла с постусловием:

```
Program pr_5a;
    Uses Crt;
Var x0, x1, eps: Real;
    n: Integer;
Begin
    Clrscr;
        Readln(x0);
        eps:= 1e-4;
        x1:= Exp(ln(29 +3*Sqr(x0) +8*x0)*(1/4));
        n:= 1;
    Repeat
        x0:= x1;
        x1:= Exp(Ln(29 +3*Sqr(x0) +8*x0)*(1/4));
        n:= n+1;
    Until Abs(x0-x1) < eps;
    Writeln(' x=', x1:7:5, ' n=', n-1);
    Readln;
End.
```

Результаты вычислений:

$x = 2.98770$; $n = 6$.

Программа с использованием оператора цикла с предусловием:

```
Program pr_5b;
    Uses Crt;
Var x0,x1,eps: Real;
n: Integer;
Begin
    Clrscr;
        eps:= 1e-4;
        X0:= 2.8;
        X1:= Exp(Ln(29 +3*Sqr(x0) +8*x0)*(1/4));
        n:= 1;
    While Abs(x0-x1)>eps do
    Begin
        x0:= x1;
        x1:= Exp(Ln(29 +3*Sqr(x0) +8*x0)*(1/4));
        n:=n+1;
    End;
    Writeln('x=',x1:10:8,' n=',n );
    Readln;
End.
```

Работа № 6. Работа с массивами

Цель работы: изучить правила работы с массивами в среде Турбо Паскаль.

***Задание**:* составить программу, сортирующую двухмерный, случайным образом сформированный массив, состоящий из заданного количества чисел в диапазоне от –10 до 10, по возрастанию столбца, номер которого задается с клавиатуры.

Методика выполнения работы:

```
Program pr_6;
 Uses Crt;
Type a= Array [1..10] Of Integer;
Var b: Array[1..10] Of a;
    sr, sb, s, i, k, et, l, y,j: Integer;
Begin
    Writeln('Введите sr и sb');
```

```
    Readln(sr,sb);
    Writeln('Введите значение номера столбца,
        по которому сортировать');
    Readln(s);
    Clrscr;
    Randomize;
        For i:= 1 To sr Do
        For j:= 1 To sb Do
            b[i,j]:=10-Random (20);
    Writeln('Исходный массив');
        For i:= 1 To sr Do
    Begin
        For j:=1 To sb Do
            Write (b[i,j]:4);
        Writeln;
    End;
        For i:= 1 To sr-1 Do
    Begin
        et:=b[i,s]; k:=i;
        For j:=i+1 To sr Do
            if b[j,s]<et Then
            Begin
                et:=b[j,s];
                k:=j;
            End;
        For l:=1 To sb Do
        Begin
            y:=b[k,l];
            b[k,l]:=b[i,l];
            b[i,l]:=y;
        End;
    End;
    Writeln('Отсортированный массив');
    For i:=1 To sr Do
    Begin
        For j:=1 To sb Do Write (b[i,j]:4);
        Writeln;
    End;
    Readln;
End.
```

Работа № 7. Использование оператора выбора case

Цель работы: изучить правила использования оператора выбора case в среде Турбо Паскаль.

Задание: составить программу, имитирующую работу микрокалькулятора. Блок-схема алгоритма приведена на рис. 4.4.

Методика выполнения работы:

```
Program pr_7;
    Var oper: Char; {Описание символа арифметического
    действия}
        x,y,z:Real; {Описание переменных, над которыми
            будут произведены действия (операндов),
            и результата вычислений}
        stop: Boolean; {Признак ошибочной операции
            и остановка}
Begin
        stop:=false;
    Repeat {Оператор цикла}
        Writeln; {Пустая строка-разделитель}
        Write('Введите x и y=');
        Readln(x,y);
        Write('операция: ');
        Readln(oper);
        Case oper Of {Выбор арифметического действия}
            '+': z:= x+y;
            '—': z:= x-y;
            '*': z:= x*y;
            '/': z:= x/y;
            Else stop:=true;
        End;
        If Not stop Then
            Writeln(' результат = ',z:6:3);
    Until stop
End.
```

Работа № 8. Использование пользовательской подпрограммы-функции

Цель работы: изучить правила работы с пользовательской подпрограммой-функцией в среде Турбо Паскаль.

Задание: составить программу для определения числа сочетаний $C_n^m = \frac{n!}{m! * (n-m)!}$, используя функцию при вычислении факториала.

Методика выполнения работы:

```
Program pr_8;
  Var n,m: Byte; {Переменная целого типа от 0 до 255
      (глобальные параметры)}
  nm: Longint;{Переменная целого типа
      от –2147483648 до +2147483647}
  {Функция вычисляет значение факториала}
Function Fact(k:Byte):Longint;
   {K — формальный параметр}
  Var p: Longint; {Локальные параметры p и i}
     i:Byte;
        Begin
           p:= 1;
           For i:= 1 To k Do p:= p*i;
           Fact:= p;
        End;
Begin
  Writeln('Введите данные для определения числа
  сочетаний');
  Readln(n,m);
  ncm:=Fact(n) Div Fact(m) Div
     Fact(n-m); {Целочисленное деление}
  Writeln('Число сочетаний = ',ncm);
  Readln;
End.
```

Работа № 9. Использование пользовательской подпрограммы-процедуры

Цель работы: изучить правила работы с пользовательской подпрограммой-процедурой в среде Турбо Паскаль.

Задание 1: составить программу для вычисления полярных координат $r = \sqrt{x^2 + y^2}$ и $f = \text{arctg}\,(y/x)$ по прямоугольным координатам x и y ($x > 0$).

Методика выполнения задания:

```
Program pr_9a;
        Uses Crt;
        Var Xi, Yi, Ri, Fi: Real;
            n, i: Integer;
Procedure Polar(x, y: Real; Var r, f: Real);
{Формальные параметры}
    Begin
        F:= Arctan(y/x);
        R:= Sqrt(x*x+y*y);
    End;
Begin
        Clrscr;
        Writeln('Введите количество точек');
        Readln(n);
            For i:=1 To n Ddo
            Begin
                Writeln('Введите значения x и y ', i,'-й точки');
                Readln(xi,yi);
                Polar(xi,yi,ri,fi); {Фактические параметры}
                Write(ri:6:3, fi:10:3);
                Writeln;
            End;
    Readln;
End.
```

Задание 2: составить программу для вычисления полярных координат $r = \sqrt{x^2 + y^2}$ и $f = \text{arctg}(y/x)$ по прямоугольным координатам x и y ($x > 0$), используя процедуру без параметров.

Методика выполнения задания:

```
Program pr_9b;
    Var X, Y, R, F: real;
        Procedure Polar;
            Begin
                F:=Arctan(y/x);
                R:=Sqrt(x*x+y*y);
            End;
    Begin
        Readln(x,y);
        Polar;
        Write(r:6:3, f:10:3);
    End.
```

Работа № 10. Операции с символами

Цель работы: изучить операции с символами в среде Турбо Паскаль.

Задание: вывести на экран все 256 символов в виде таблицы 16×16. На месте служебных символов с кодами 7 (звонок), 8 (забой), 9 (табуляция), 10 (следующая строка), 13 (начало строки), 26 (конец текста) и 27 (Esc) поставить крестики.

Методика выполнения работы:

```
Program pr_10;
    Var
        i, j: Byte;
        c: Char;
Begin
    WriteLn; {Пустая строка в начале}
    Write(' ');
    For j:= 0 to 15 Do
        Write(j:4); {Номера столбцов}
```

```
    WriteLn;
  For i:= 0 to 15 Do
  Begin
    Write(i*16:3, ' '); {Номер строки*16}
    For j:= 0 To 15 Do
    Begin
      {Символ, который нужно вывести в этом
      месте таблицы}
      c:= Chr(i*16+j);
      If ((c >=# 7) And (c <= #10)) Or
        (c =# 13) Or (c = #26) Or (c = #27) Then
          Write('XXX':4) {Нельзя показать}
        Else
          Write(c:4); {Можно показать}
    End;
    WriteLn; {Курсор — в начало следующей строки}
  End;
  Readln;
End.
```

Работа № 11. Операции со строками

Цель работы: изучить операции со строками в среде Турбо Паскаль.

Задание: дана строка символов, состоящая из слов, разделенных пробелами. Найти среднюю длину слова.

Методика выполнения работы: основная проблема в этой задаче — определить количество слов и длину каждого из них. Что считается началом слова? Ситуация, когда в паре соседних символов левый — пробел, а правый — любой другой символ. В конце слова — наоборот. Необходимо просмотреть строку в поисках таких пар и подсчитать количество слов и их суммарную длину.

```
Program pr_11;
  Var
  s: String;
```

```
    i, n, sum, last: Byte;
Begin
    Write('Введите слова, разделенные пробелами: ');
    ReadLn(s);
    {Уберем ведущие пробелы}
    While (Length(s)>0) And (s[1] = ' ') Do
        Delete(s, 1, 1);
    If Length(s) = 0 Then
        WriteLn('Ошибка! Строка не содержит слов!')
    Else begin
        n:= 0;{Количество слов}
        last:= 0;{Позиция начала слова}
        sum:= 0;{Суммарная длина слов}
        s:= s + ' ';{Дописываем пробел в конец строки,}
            {Чтобы и у последнего слова был конец}
        {Просматриваем строку с первого
        до предпоследнего символа}
        For I:= 1 To Length(s) — 1 Do
        Begin
            If (s[I] = ' ') And (s[I+1] <> ' ')
                {Начало слова}
            Then last:= I;{Запоминаем позицию}
            Else
                If (s[I+1] = ' ') And (s[I] <> ' ')
                {Конец слова}
                Then
                Begin
                    n:= n + 1;
                    sum:= sum + I — last;
                End
        End;
        WriteLn('Средняя длина слова ', sum/n:7:3);
    End;
End.
```

Работа № 12. Операции с записями

Цель работы: изучить операции с записями в среде Турбо Паскаль.

Задание: сформировать базу данных, состоящую из 10 записей по 4 поля в каждой: фамилия, имя, год рождения и телефон. Вывести на экран записи, в которых содержатся сведения об Иванове.

Методика выполнения работы:

```
Program pr_12;
    Uses Crt;
Type
    person = Record
    sname,name: String[14];
    gr: Integer;
    tele: String[8];
    End;
Var spisok: Array[1..10] Of Person;
    i:Integer;
Begin
    For i:= 1 To 10 Do
    Begin
        With spisok[i] Do
        Begin
            Writeln('Фамилия');
            Readln(name);
            Writeln('Имя');
            Readln(name);
            Writeln('Год рождения');
            Readln(gr);
            Writeln('Телефон');
            Readln(tele);
        End;
    End;
    Writeln('Список Фамилия Имя Год рождения Телефон');
    For i:= 1 To 10 Do
```

```
    Begin
        With spisok[i] Do
            If sname = 'Иванов' then
            Writeln (sname:15, name:10, gr:8, tele:10);
        Writeln;
    End;
    Readln
End.
```

Работа № 13. Множества

Цель работы: изучить правила работы с множествами в среде Турбо Паскаль.

Задание: даны два множества: **X1** и **X2**, содержащие элементы типа **Byte**. Сформировать новое множество **Y**, равное разности множеств **X1** и **X2**, и выделить из него подмножество **Y1**, содержащее элементы, делящиеся без остатка на 5 и на 3. На экран вывести множества и их мощность.

Методика выполнения работы:

```
Program pr_13;
    Uses Crt;
    Type SetByte= Set Of Byte;
    Const N= 10;
Var
        X1, X2, Y, Y1: SetByte;
        M, M1, i, A, B: Byte;
Procedure ShowSet(S: SetByte; Str: String; Ms: Byte);
    Var T: Byte;
    Begin
        Writeln('Множество' ,Str, '(мощность — ',Ms, ') :');
        For T:= 0 To 255 Do
            If T In S Then Write(T, '' );
        Writeln;
    End;
```

```
Begin
    X1:= [];
    X2:= [];
    Y1:= [];
    M:= 0;
    M1:= 0;
        {m,m1 — мощности множеств Y,Y1}
    For i:= 1 To N Do
        Begin
            ClrScr;
            Write('Введите ',i, '-й элемент множества X1 =>');
            Readln(B);
            X1:= X1+[B];
            ClrScr;
            Write('Введите ',i, '-й элемент множества X2 =>');
            Readln(B);
            X2:= X2+[B];
        End;
    Y:= X1 — X2;
    For A:= 1 To 255 Do
        If A In Y Then
        Begin
            Inc(M);
            If (A Mod 3 = 0) and (A Mod 5 = 0) Then
                Begin
                    Inc(M1);
                    Y1:= Y1+[A];
                End;
        End;
    ClrScr;
    ShowSet(X1, 'X1', N);
    ShowSet(X2, 'X2', N);
    ShowSet(Y, 'Y, разность X1 и X2', M);
    ShowSet(Y1, 'чисел, кратных 5 и 3', M1);
    Readln;
End.
```

Работа № 14. Операции с файлами

Цель работы: изучить операции с файлами в среде Турбо Паскаль.

Задание: дан текстовый файл. Создать его копию, которая не должна содержать пустых строк. Сначала с клавиатуры запрашивается путь доступа к файлу до тех пор, пока файл не удастся открыть для чтения. Затем запрашивается другой путь доступа до тех пор, пока не выяснится, что не существует файла с таким именем, после чего информация из первого файла записывается во вновь созданный второй файл.

Методика выполнения работы:

```
Program pr_14;
    {$I-} {Директива компилятора — выключение
    контроля ошибок ввода-вывода}
Var
     fromF, toF: Text;
    fromName, toName: String;
        s: String;
Begin
    Repeat
        Write('Введите имя исходного файла');
        ReadLn(fromName);
        Assign(fromF,fromName); {Пытаемся открыть
        для чтения}
        Reset(fromF); µ\13
    Until IoResult=0; {Если открыть
        не удалось — на начало цикла}
    Repeat
        Write('Введите имя файла-приемника');
        ReadLn(toName);µ\13
        Assign(toF, toName); {Проверка существования}
        Reset(toF); {Если открыть удалось —
            на начало цикла}
    Until IoResult <> 0;
    {Открываем по-настоящему — для записи}
    Rewrite(toF);
```

```
 While Not Eof(from F)
Do Begin
    {Читаем строку из первого файла}
    ReadLn(fromF,s);
    WriteLn(s); {Выводим на экран}
    If s <> '' Then
        WriteLn(toF, s); {Выводим во второй файл}
End;
Close(fromF);
Close(toF); {Закрываем оба файла}
Readln
End.
```

Тема 7.2

РАЗЛИЧНЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

7.2.1. Машинная графика

Машинная (компьютерная) графика объединяет в себе процессы создания, хранения и обработки моделей объектов и их изображений с помощью ЭВМ. Эта технология проникла в область экономического анализа и моделирования различного вида конструкций, незаменима в производстве и рекламной деятельности, делает занимательным досуг. Формируемые и обрабатываемые с помощью цифрового процессора изображения могут быть демонстрационными и анимационными. К демонстрационной группе, как правило, относят коммерческую (деловую) и иллюстративную графику, к анимационной — инженерную и научную, а также связанную с рекламой, искусством, играми, когда выводятся не только одиночные изображения, но и последовательность кадров в виде фильма (интерактивный вариант). Интерактивная машинная графика одно из наиболее прогрессивных направлений новых информационных технологий, претерпевающее бурное развитие в области появления новых графических станций и специализированных программных средств, позволяющих создавать реалистические

объемные движущиеся изображения, сравнимые по качеству с кадрами видеофильма.

Машинная графика реализуется с помощью различных прикладных программ (текстовых процессоров и редакторов), в частности Microsoft Word (см. тему 4.8) и CorelDRAW.

CorelDRAW на сегодняшний день самый популярный графический редактор, работающий с векторной графикой. В настоящее время в эксплуатации находится версия 9.0 этой программы. Для учебно-методических целей эта версия представляет особый интерес, поскольку имеет русскоязычный аналог. На примере этой программы мы рассмотрим основные приемы создания и редактирования графических образов.

Главное окно, **элементы интерфейса**. Интерфейс программы CorelDRAW выполнен в традициях, ставших стандартом для приложений в операционных системах Windows.

После того как графический редактор CorelDRAW запущен, на экране появляется рабочее окно программы (рис. 7.6). Элементы управления программы сосредоточены в строке меню, на панели инструментов *Стандартная*, панели свойств и панели инструментов *Графика*. Основную же часть окна составляет *рабочая область,* где в центре представлена *рабочая страница,* — это не более чем ориентир: создавать объекты можно как на странице, так и вне ее, но при выводе на печать будет напечатано только то, что находится внутри рабочей страницы.

Заголовок, кнопки управления, строка *меню,* расположенные в верхней части окна, а также полосы прокрутки и строка состояния аналогичны таким же элементам любого приложения Windows.

В правой части окна находится *палитра цветов*, с помощью которой вы можете задавать цвет элементам изображения. Сверху и слева в окне расположены *измерительные линейки,* предназначенные для точного позиционирования объектов и определения их размеров.

На панели инструментов *Стандартная* содержатся инструменты; часть из них: *New — Создать, Open — Открыть, Save — Сохранить* и т. д., — имеется и во многих других программах Windows, а часть присуща только приложениям CorelDRAW: *Import — Импортировать, Export — Экспортировать, Application*

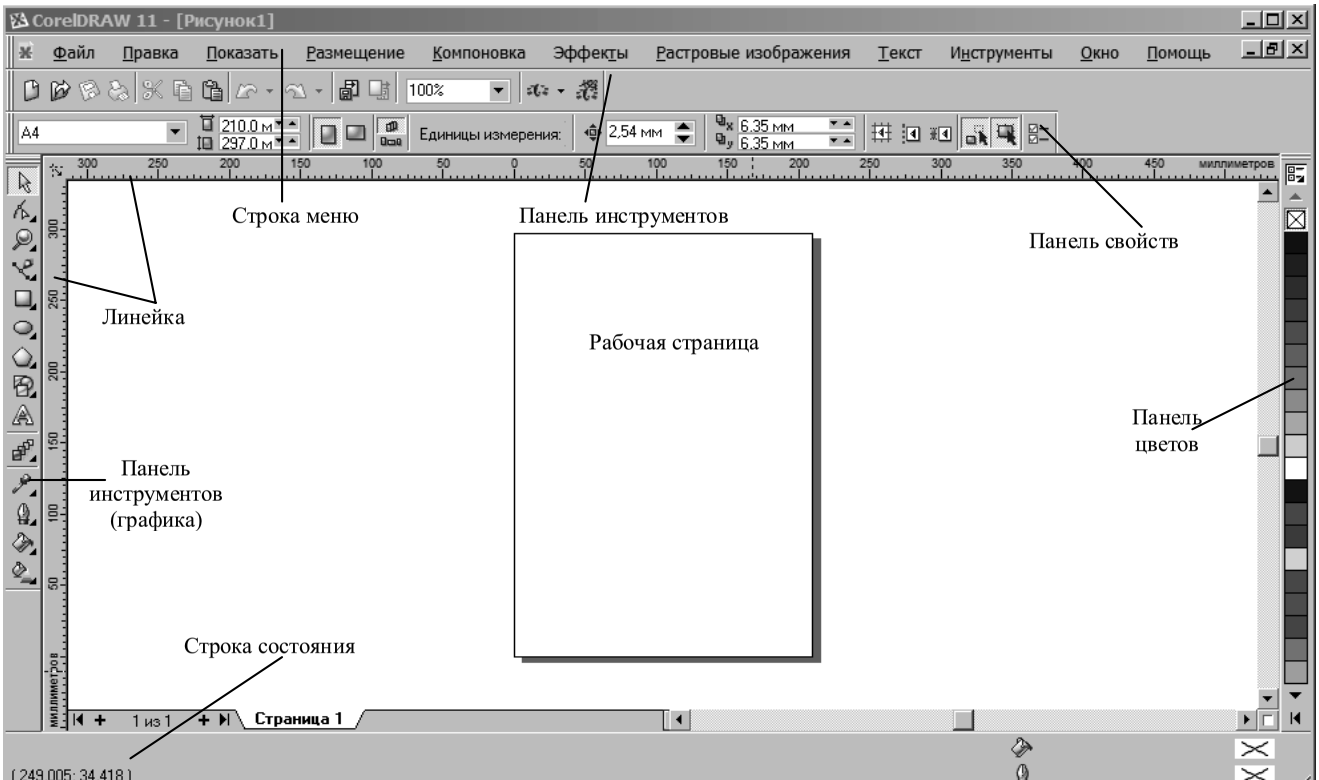


Рис. 7.6. Основное окно программы CorelDRAW

Launcher — Кнопка запуска приложений, What's this? — Справочная система.

Панель свойств, расположенная по умолчанию непосредственно под панелью инструментов *Стандартная,* обладает уникальными интерактивными свойствами. Состав ее элементов управления динамически меняется в зависимости от типа выбранного объекта. Так, при выборе текста на панели свойств появляются элементы управления свойствами текста, при выборе линии — элементы управления свойствами линии. Каждый тип объектов имеет на панели свойств присущий только ему комплект элементов управления.

Особенностью интерфейса CorelDRAW является наличие таких элементов, как стыкуемые панели, которые обладают свойствами «приклеиваться» при перетаскивании мышью к одной из сторон рабочего поля. **Стыкуемые панели** — это особого вида диалоговые окна, которые служат для настройки действия основных инструментов. При желании панели можно «открепить» и сделать «плавающими», благодаря чему удается очень быстро получить доступ к содержащимся в них элементам управления. Одним из наиболее значительных их преимуществ, по сравнению с диалоговыми окнами, является то, что можно вернуться к работе над страницей, оставив *Стыкуемую панель* открытой.

Панель инструментов *Графика* по умолчанию расположена вдоль левого края окна и содержит все основные инструменты, применяемые для создания и рисования объектов, текстов и эффектов, а также манипулирования ими.

Если у вас возникает желание произвести настройку интерфейса и параметров CorelDRAW, следует воспользоваться командой меню *Инструментальные средства → Опции.* Все настройки подразделяются на три основных раздела:

— ***рабочая область*** — содержит средства управления параметрами интерфейса;

— ***документ*** — позволяет управлять свойствами разрабатываемого документа;

— ***глобальное*** — управляет параметрами взаимодействия редактора с аппаратными устройствами и другими внешними компонентами.

Создание нового документа. Создание и сохранение документа осуществляется с помощью стандартных способов Windows.

После того как вы создали новый документ, необходимо настроить параметры страницы и рабочего окружения. По умолчанию страница нового документа имеет формат А4 (210 × 297 мм) — самый распространенный формат бумаги — и вертикальную ориентацию (рис. 7.7).

Рис. 7.7. Настройка параметров страницы

Существует два способа настройки параметров страницы:

— *способ А* — воспользуйтесь командой меню *Макет → Параметры страницы*. В диалоговом окне *Параметры страницы* можно выбрать 19 стандартных форматов; установите свой формат, укажите ориентацию: *Книжная* или *Альбомная;*

— *способ Б* — на панели свойств выбираем из списка *Тип* формат бумаги (здесь же будет показан размер бумаги), из списка *Единицы измерения* — миллиметры или другую удобную для вас единицу. Для изменения ориентации рабочей страницы щелкните на одной из кнопок: *Книжная* или *Альбомная*.

Приемы создания простейших объектов. Наше знакомство с редактором CorelDRAW начнем с создания простейших геометрических объектов: прямоугольников, эллипсов, многоугольников, спиралей, прямых и кривых линий.

Прямоугольник, Квадрат, Эллипс, Окружность

Выберите инструмент *Прямоугольник* или *Эллипс* на панели инструментов *Графика* и методом протягивания на рабочей странице создайте прямоугольник или эллипс. В процессе протягивания в строке состояния отображаются точные координаты **начальной** и **конечной** точек, а также **центра** прямоугольника или эллипса и его **размеры** по **ширине** и **высоте**. Координаты, удовлетворяющие вашим условиям, можно задать на панели свойств.

Если в ходе построения прямоугольника или эллипса удерживать клавишу <*Ctrl*>, будет нарисован квадрат или окружность. Прием рисования фигуры «от центра» выполняется при нажатой клавише <*Shift*>. При работе с прямоугольником можно изменить форму его углов, т. е. скруглить их, воспользовавшись элементами управления параметрами скругления углов, расположенными на панели свойств (рис. 7.8).

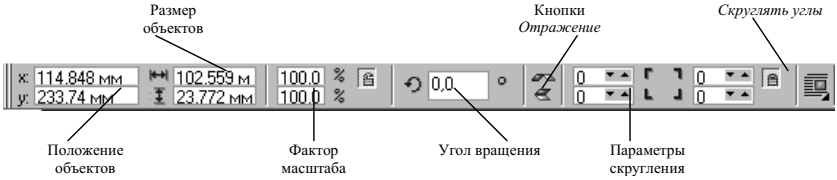


Рис. 7.8. Панель свойств для задания параметров прямоугольника

Эллипсы могут принимать вид дуг и секторов, для этого необходимо щелкнуть на соответствующей кнопке: *Сектор* или *Дуга,* которые расположены также на панели свойств. Можно задать величину углов наклона начала и конца сектора или дуги (рис. 7.9).

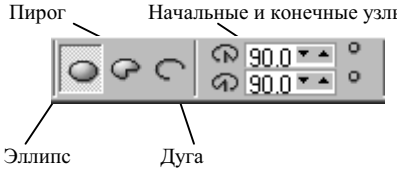


Рис. 7.9. Панель свойств для задания параметров эллипса

Многоугольник, спираль, звезда, клетки

Инструмент *Многоугольник* предназначен для рисования многоугольников с различным числом сторон. Он содержит панель инструментов *Вспомогательная*, с ее помощью создаются спирали, звезды и клетки. Если при их построении использовать клавиши <*Ctrl*> и <*Shift*>, можно получить фигуры правильной формы. Перед созданием многоугольника или после этой операции вы можете изменить внешний вид объекта с помощью управляющих элементов, расположенных на пане-

ли свойств, задав любое число вершин или сторон многоугольника (рис. 7.10).

Рис. 7.10. Панель свойств для задания параметров многоугольника

Для создания *звезды* используют кнопку-переключатель *Многоугольник/Звезда*, при щелчке на которой будет создаваться звезда; повторное нажатие вернет вас к созданию многоугольника.

С помощью инструментов *Спираль* и *Миллиметровка* получают соответствующие фигуры, т. е. спираль и клетки миллиметровки. Рисуются эти объекты точно так же, как и описанные выше. Предварительно можно настроить параметры создаваемого объекта. Для клетки задается количество ячеек по горизонтали и вертикали (рис. 7.11).

Рис. 7.11. Задание параметров клетки

Чтобы задать число витков спирали воспользуйтесь элементом управления *Обороты Спирали* (рис. 7.12).

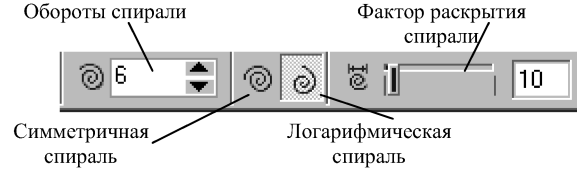


Рис. 7.12. Задание параметров спирали

Обычные спирали создаются при щелчке на кнопке *Симметричная Спираль,* а логарифмические спирали — при щелчке на кнопке *Логарифмическая.* С помощью ползунка *Фактор Раскрытия Спирали* мы можем изменить расстояние между соседними витками спирали, т. е. она будет расширяться с каждым витком.

Рисование линий. Все имеющиеся в CorelDRAW инструменты рисования линий, по сути, предназначены для создания кривых, соединяющих точки; причем эти кривые представляют собой векторные фигуры, а точки — узлы. Можно создавать замкнутые и незамкнутые кривые, соединительные линии, динамически привязанные к объектам или формам: все будет зависеть от того, какой инструмент вы выберите.

Линию можно наделить рядом свойств, таких, как цвет, толщина или узоры. Линии могут быть прямыми или кривыми и содержать один или более узлов. Если начальная и конечная точки линии совпадут, то она становится замкнутым контуром. Инструменты для рисования линий подразделяются на пять основных типов, сгруппированных на панели инструментов *Вспомогательная* (рис. 7.13):

Рис. 7.13. Вспомогательная панель для рисования

— *СвободноеРисование;*

— *Безье;*

— *Артистические Носители;*

— *Размерность;*

— *Строки Соединителя и интерактивный инструмент Соединитель.*

В зависимости от того, какой инструмент выбран, изменяется вид панели свойств.

Для создания обычных прямых или кривых линий служит инструмент *Свободное Рисование;* он действует, подобно карандашу:

— кривую линию рисуют протягиванием мыши;

— прямую — щелчками в начальной и конечной точках;

— при нажатой клавише <*Ctrl*> для прямой линии можно задать фиксированный угол наклона (с шагом 15°);

— воспользовавшись панелью свойств, соответствующими переключателями можно задать тип и толщину линии, ее форму на концевых точках;

— для незамкнутых линий можно воспользоваться кнопкой *Авто-Закрывать Кривую* и, соединив конечные точки (рис. 7.14), превратить линию в замкнутый контур.

Рис. 7.14. Панель свойств для рисования линий

Инструмент *Безье* обеспечивает максимально возможную степень контроля над **формой кривой** в процессе рисования. Приемы рисования те же: линии создаются щелчками в опорных точках. Для рисования кривых необходимо после щелчка слегка протянуть мышь от опорной точки. На экране появится пунктирная линия, длина и направление которой меняются с передвижением мыши. Эту пунктирную линию называют **манипулятором кривизны**. Внешний вид кривой *Безье* определяется длиной и наклоном манипулятора кривизны, а также координатами опорных точек.

С помощью инструмента *Артистические Носители* можно создавать **жирные кривые** линии переменной толщины. Их отличие от описанных выше заключается в том, что данный инструмент создает форму с закрытым контуром. *Артистические Носители* предоставляют доступ к пяти разным режимам рисования и другим параметрам, представленным на панели свойств (рис. 7.15):

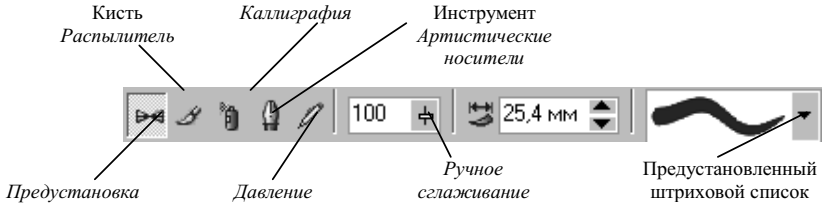


Рис. 7.15. Панель свойств для настройки параметров рисования кривых линий

— щелкнув на кнопке *Предустановка,* вы тем самым выбираете **работу с заготовками**;

— в *Предустановленном Штриховом Списке* к создаваемой линии можно применить готовую векторную фигуру; здесь содержится комплект из 23 разных стилей *Пера;*

— параметры панели свойств позволяют изменять **ширину** создаваемого объекта и устанавливать **степень сглаживания**;

— щелкнув на кнопке *Кисть,* вы можете нарисовать линию или применить разнообразные **стили мазков кистью** к уже имеющейся линии;

— при выборе режима *Распылитель* графическое изображение располагается вдоль создаваемой или готовой кривой, объекты **распыляются по всей ее длине**; их размер, расстояние между ними, частота и количество поворотов определяются с помощью панели свойств, здесь же выбираются различные стили данного режима;

— в режиме *Каллиграфия* создаются линии, **толщина** которых зависит от наклона; при этом угол наклона устанавливается на панели свойств с помощью управляющих элементов;

— с помощью кнопки *Давление, чувствительное к нажиму* можно нарисовать линию, толщина которой будет зависеть от **степени нажима**. В местах сильного нажима линия будет толще, в местах слабого нажима — тоньше. Чаще всего данный режим выбирают, если есть графический планшет и специальное *Перо*, применяемое для ввода.

Чтобы создать **размерные линии,** воспользуйтесь инструментом *Размерность* на панели инструментов *Вспомогательная;* при этом на панели свойств будет отображено шесть разных режимов проставления размеров, каждый из которых соответствует конкретным целям. До начала рисования можно указать тип размерных линий: расставляемые автоматически, вертикальные, горизонтальные, наклонные, выносные и угловые. Процедура рисования размерной линии состоит из трех этапов:

— первым щелчком кнопки мыши вы указываете точку, от которой измеряется расстояние;

— вторым — конечную точку, до которой измеряется расстояние;

— третьим — точку местоположения надписи с размером.

На панели свойств также расположены группы различных раскрывающихся меню, полей ввода текста и командных кнопок, с помощью которых вы можете управлять внешним видом и свойствами размерных линий (рис. 7.16):

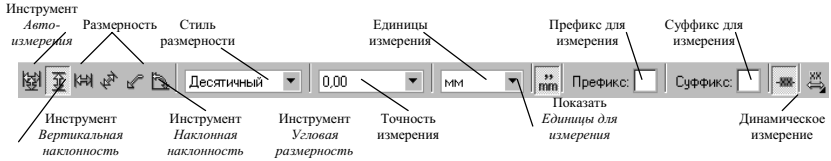


Рис. 7.16. Панель свойств для управления внешним видом и свойствами размерных линий

— раскрывающийся список *Стиль Размерности* предназначен для того, чтобы определить формат представления результатов измерения: *Десятичный, Дробный, Инженерный, Архитектурный*.

— раскрывающийся список *Точность Измерения* позволяет настраивать формат отображения чисел при заданных единицах измерения;

— единицы измерения для размерных линий выбираются в раскрывающемся списке *Единицы Измерения;* с помощью кнопки *Показать Единицы для Измерения* можно скрывать или отображать единицы измерения на размерных линиях;

— в полях *Префикс* и *Суффикс для Измерения* при необходимости проставляют текстовые или цифровые данные. В поле *Префикс* текст предшествует размерному числу, а в поле *Суффикс* — следует за ним;

— кнопка *Динамическое Измерение* включает или отключает режим динамического представления размещения данных во время выбора опорных точек;

— раскрывающееся меню *Позиция текста* содержит кнопки выбора стиля размещения надписей на размерных линиях.

В CorelDRAW существует три способа **соединения линий и объектов**:

— первый способ заключается в рисовании линий между фигурами **вручную**;

— второй способ основан на использовании инструмента *Строки Соединителя,* который находится на панели инструмен-

тов *Вспомогательная*. В данном случае соединительная линия будет связывать объекты между ближайшими точками привязки;

— третий способ — наиболее эффективный — состоит в использовании интерактивного инструмента *Соединитель,* который позволяет создавать динамические соединения между объектами с помощью комбинаций вертикальных и горизонтальных линий. Связанные таким образом объекты можно изменять и перемещать по странице, не нарушая при этом соединения.

Преобразование объектов. Различные приемы работы с контурами объектов. Все создаваемые в CorelDRAW объекты имеют одинаковый вид и толщину контура. Но иногда для создания иллюстраций требуется применить отличающиеся друг от друга контуры. Для этих целей используется специальный инструмент *Диалог Пера Контура,* расположенный на панели *Вспомогательная* инструмента *Контур* и предназначенный для установки различных атрибутов контура объекта (рис. 7.17).

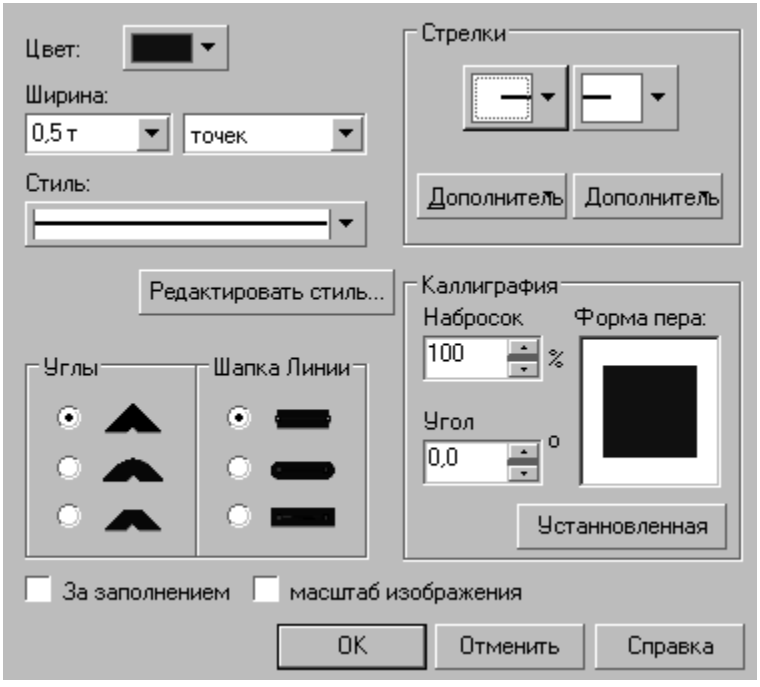


Рис. 7.17. Установка атрибутов контура объекта

С помощью элементов управления можно изменять толщину контура, а также открывать диалоговые окна *Перо Контура* и *Цвет Контура* (рис. 7.18).

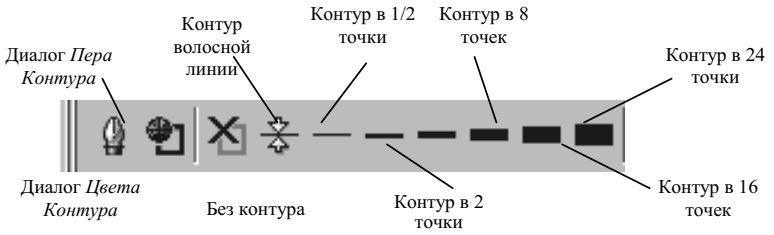


Рис. 7.18. Диалоговое окно *Перо Контура*

Но для того чтобы точно откорректировать все свойства контура объекта, нужно воспользоваться набором соответствующих параметров в диалоговом окне *Перо Контура:*

— **изменить цвет абриса** выделенного объекта можно, выбрав нужный цвет на пиктограмме цветовой палитры;

— в поле *Толщина* расположены числа, обозначающие **толщину контура**;

— рядом в раскрывающемся списке выбираются единицы измерения;

— в списке *Стиль* можно выбрать **стиль контура**: непрерывную или прерывистую линию;

— переключатели *Углы* и *Концы линий* позволяют выбрать один из **вариантов углов** и **окончания линий**;

— флажок *Позади заполнения* располагает контур под заливкой; если же он не установлен, контур будет расположен над заливкой. При расположении под заливкой половина контура не видна, что особенно удобно при работе с текстом;

— если установлен флажок *Масштабировать с изображением,* то при изменении **размеров объекта** пропорционально будет меняться и толщина контура;

— в группе полей *Стрелки* выбирают **наконечники**, которые расположатся в начале и конце линии;

— командную кнопку *Options* можно использовать как для создания **новых стилей** наконечников, так и для редактирования уже существующих;

— группа полей *Каллиграфия* определяет наклон и форму *Пера,* формирующего контур.

Простейшие операции по изменению размера и формы объектов. Перед тем как сделать что-либо в CorelDRAW, необходимо создать объект, а прежде чем сделать что-либо с этим объектом, его следует **выделить** с помощью инструмента *Указания,* выбранного на панели инструментов *Графика*. При этом вокруг объекта, независимо от его формы или размера, появится восемь **маркеров выделения**, которые позволяют не только увидеть выделенный в данный момент объект, но и изменить его размер.

Группу объектов выделяют щелчками левой клавиши мыши при нажатой клавише <*Shift*>. При этом в *Строке состояния* выводится информация о выделенных объектах. Повторное нажатие клавиши <*Shift*> на контуре одного из выделенных объектов отменяет его выделение. Об этом можно также узнать из *Строки состояния*.

Изменение размера объекта производится после его выделения путем перетаскивания соответствующих маркеров.

Чтобы **переместить объект**, его необходимо выделить с помощью инструмента *Указания* и мышью перетащить на новое место. Если при этом удерживать клавишу <*Ctrl*>, движение будет происходить только по вертикали или по горизонтали.

Чтобы **зеркально отразить объект**, надо один из маркеров выделения перетащить на другую сторону объекта, пока его зеркальное отражение не примет требуемую форму или положение. С помощью угловых маркеров выделения можно отражать объект относительно вертикальной, горизонтальной или наклонной оси, выполняя при этом масштабирование. Нажатием клавиши <*Ctrl*> можно сделать размер отражаемого изображения равным размеру оригинала, а нажатием клавиши <*Shift*> — отразить объект относительно центральной точки.

Чтобы **повернуть объект**, необходимо его выделить и затем щелкнуть на нем, что приведет к появлению маркеров поворота и скоса в виде стрелок. Стрелки по углам служат для вращения объекта, стрелки по сторонам объекта используются для его деформации. При этом все перемещения будут происходить вокруг центра вращения. Нажатие клавиши <*Ctrl*> приводит к повороту или скосу объекта на углы, кратные 15°.

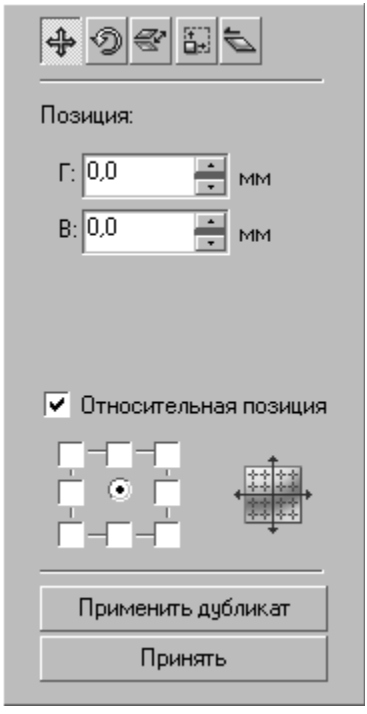


Рис. 7.19. Окно панели *Преобразования*

Для задания **точных значений** параметров преобразований лучше открыть прикрепленную панель *Преобразования,* выполнив команду <*Alt*> + <*F7*> (рис. 7.19). Эта панель предоставляет более «интуитивный» способ размещения, поворота и т. д. Все параметры преобразования, включая дополнительные параметры некоторых функций, устанавливаются в одном месте. В верхней части панели расположено пять кнопок инструментов преобразования: *Положение, Вращать, Масштабировать и Отражать, Размер* и *Наклонить.*

Существует два способа **размещения объекта** или **группы объектов** с помощью панели *Преобразования* — относительно **текущего объекта** и относительно **системы координат**, связанной с документом.

Чтобы переместить объект первым способом, следует установить флажок *Относительное положение* и ввести значение горизонтального и вертикального перемещения объекта. Для ввода расстояний можно воспользоваться кнопками приращения.

Чтобы переместить объект в определенное место на странице (второй способ), необходимо сбросить флажок *Относительное положение* и ввести горизонтальную и вертикальную координаты нового положения выбранной якорной точки относительно линеек. Эти якорные точки используются для установки положения преобразованного объекта по отношению к исходному объекту.

Поворот объекта задается только двумя параметрами: углом поворота и центром вращения. Угол вводится в поле *Угол* в диапазоне от −360° до 360° с точностью до 0,001°.

Центр вращения можно задать несколькими способами:

— установите флажок *Относительный Центр* и щелкните на центральной якорной точке. Координаты центра H и V будут установлены равными 0. Чтобы изменить положение центра вращения, введите координаты центра в поле *Центр;*

— сбросьте флажок *Относительный Центр*. Введите горизонтальную и вертикальную координаты центра поворота. Щелкните на кнопке *Применить* или *Применить к дубликату,* чтобы повернуть объект вокруг этой точки на указанный угол;

— поворот объекта можно произвести относительно одной из якорных точек — угловой или серединной, — после чего следует щелкнуть на кнопке *Применить;*

— третий инструмент панели *Преобразования — Масштабировать и Отразить* — предназначен для выполнения двух действий. При масштабировании объекта устанавливаются такие параметры, как *Масштаб,* включающий поля ввода *H* и *V*, якорная точка и флажок *Непропорциональная*. Кнопки отражения позволяют отразить объект по горизонтали, по вертикали или относительно обеих осей сразу. В поле *Масштаб* можно указать проценты масштабирования, что приведет к созданию отраженного масштабированного изображения. Якорные точки определяют, в каком месте по отношению к исходному объекту будет создан преобразованный объект.

Инструмент *Размер* панели *Преобразования* позволяет точно установить общие размеры объекта группы объектов. При выделении объекта в полях *Размеры H и V* указываются его размеры. Если установить флажок *Непропорциональная,* можно ввести независимые значения для высоты и ширины объекта. Если этот флажок сброшен, достаточно установить только ширину или только высоту — при преобразовании будут сохранены пропорции объекта. Якорные точки определяют размещение нового объекта.

Инструментами *Наклон* или *Скос* вдоль горизонтальной оси можно привязать объект с помощью якорной точки к верху, низу или центру исходного объекта. Инструментом *Наклон* вдоль вертикальной оси можно привязать объект к левому краю, правому краю или к центру исходного объекта.

Порядок объектов. Каждый объект в CorelDRAW имеет свое место среди других объектов. Это значит, что он может нахо-

диться впереди или позади остальных объектов документа. Второй создаваемый объект помещается на первый, даже если они не пересекаются по горизонтали или вертикали. Третий объект рисуется перед вторым и т. д. Независимо от того, как тщательно спланирован документ, иногда приходится изменять порядок некоторых объектов.

Для работы над упорядочением объектов существует команда меню *Упорядочить → Порядок*. Существует выбор из семи команд изменения порядка объектов: *На Перед* — помещает выделенный объект(ы) перед всеми остальными объектами; *На Зад* — помещает выделенный объект(ы) позади всех остальных объектов; *Вперед Один (Назад Один*) — перемещает объект(ы) на одну позицию вперед либо на одну позицию назад в порядке объектов; *Вперед на (Позади*) — помещает выделенный объект(ы) перед или за объектом, указываемым после выбора команды; *Обратный порядок* — обращает порядок следования выделенных объектов.

Редактирование прямых и кривых линий. Чтобы получить достаточно сложный объект, необходимо управлять формой уже нарисованных кривых. Как уже говорилось, форма кривой определяется свойствами узлов. Каждый узел содержит хотя бы один маркер, который может быть размещен в любом месте. Изменяя положение маркера, можно управлять формой кривой *Безье.* Выделив любую часть этой кривой, можно трансформировать или редактировать ее, а также изменять характеристики прямой, кривой или узла.

Редактирование кривой *Безье* производится инструментом *Форма,* позволяющим прямые линии преобразовывать в кривые, а узлы определить как узлы излома, гладкие или симметричные. Большинство изменений формы выполняется с помощью элементов управления, расположенных на панели *Свойства:* можно удалять, объединять или разъединять узлы, трансформировать прямые линии в кривые (и наоборот), переключать состояние узлов, а также изменять их порядок или трансформировать выделенные узлы.

Однако перед тем как произвести какие-либо изменения с объектом, выделите его и затем выберите инструмент *Форма*. После этого можно модифицировать узловые точки объекта (рис. 7.20). Выберите один из узлов кривой и либо переместите

Рис. 7.20. Панель свойств при работе с инструментом *Форма*

его в нужное положение, либо воздействуйте на управляющие касательные. Обратите внимание, что на панели свойств будут отображены допустимые операции для работы с прямыми линиями или узлами:

— с помощью кнопок *Добавить Узел* или *Удалить Узел* можно **добавлять новые узлы** к кривой или удалять выделенные узлы;

— если выделить два узла и щелкнуть на кнопке *Соединить Два Узла,* то узлы объединятся, в результате получится **непрерывная кривая**. Можно объединить только начальный и конечный узлы кривой, состоящей из одного сегмента. Если выделить один узел и щелкнуть на кнопке *Разбить Кривую,* получится два совмещенных узла, и кривая будет разбита на два сегмента;

— с помощью кнопок *Конвертировать Кривую в Линию* и *Конвертировать Линию в Кривую* можно преобразовывать выделенные **прямые** в **кривые** и наоборот.

Организация элементов рисунка. **Группировка объектов**. При создании сложных рисунков чаще всего приходится работать с группой объектов как с единым целым. Способ связывания объектов, позволяющий обращаться с ними как с единым целым, называется **группировкой**. К группе объектов можно применить эффект пошагового перехода в другую группу, эффект перспективы и др. Чтобы сгруппировать несколько объектов, нужно выделить их, а затем:

— либо выполнить команду меню *Упорядочить* → *Сгруппировать*;

— либо щелкнуть на кнопке *Сгруппировать* панели свойств;

— либо нажать комбинацию клавиш <*Ctrl*> + <*G*>.

В результате получится единая группа объектов с общим комплектом маркеров выделения. Этот факт будет отображен в *Строке состояния*.

Довольно часто используется прием группировки **нескольких групп в главную**. Следует отметить, что каждая группа объектов будет расположена на соответствующем слое, о чем вы найдете сообщение в *Строке состояния*.

Чтобы **разложить группу объектов на составляющие** ее элементы, нужно выделить объекты, а затем:

— либо выполнить команду меню *Упорядочить → Разгруппировать;*

— либо щелкнуть на кнопке *Разгруппировать* на панели свойств;

— либо нажать комбинацию клавиш <*Ctrl*> + <*U*>.

В данном случае происходит отмена команды *Группировать* первого уровня. Если же группа содержит вложенные группы, то они при этом будут сохранены.

Чтобы **разгруппировать все выделенные отдельные группы**, нужно еще раз выполнить команду *Упорядочить → Разгруппировать Все*. Эта команда отменяет групповую взаимосвязь всех элементов группы. независимо от того, являются они элементами вложенных групп или нет.

Выравнивание и распределение объектов. В процессе создания изображений часто приходится располагать объекты по одной линии или на равном расстоянии друг относительно друга. Решить эти задачи можно с помощью команд *Выровнять* и *распределить*. Благодаря этим командам вы одним-двумя щелчками можете выровнять объекты, а применив различные общие опорные точки, — равномерно распределить их; при этом CorelDRAW сам вычислит расстояние между объектами, учитывая их размеры. Команды *Выровнять* и *Распределить* содержатся в одном диалоговом окне. Данные команды применимы только к группе объектов.

Команда *Выровнять* позволяет быстро, причем разными способами, выровнять объекты на странице. Для этого либо щелкните на кнопке *Выровнять* на панели *Свойства,* либо выберите меню *Упорядочить → Выровнять и распределить,* после чего появится диалоговое окно *Выровнять и распределить* (рис. 7.21), открытое на вкладке *Выровнять*. В окне вы можете установить необходимые флажки выравнивания, которые действуют как переключатели. Выравнивать можно лишь в каком-то **одном направлении**:

— при выравнивании объектов **по вертикали** вы можете установить один из флажков: *Слева*, *По центру* или *Справа;*

— при выравнивании **по горизонтали** необходимо установить один из флажков: *Сверху*, *По центру* или *Снизу;*

Рис. 7.21. Диалоговое окно *Выровнять и распределить*

— если вы хотите выровнять объекты по **краям страницы**, установите флажок *Край страницы* в комбинации либо с одним из флажков выравнивания по вертикали: *Слева* или *Справа,* либо с одним из флажков выравнивания по горизонтали: *Сверху* или *Снизу;*

— если необходимо выровнять один или несколько выделенных объектов **по ближайшей к нему опорной точке** страницы, установите флажок *Выровнять по сетке* в комбинации с каким-либо флажком выравнивания по вертикали или по горизонтали.

Команда *Распределить* позволяет **автоматически разместить объекты** через определенные интервалы. На вкладке *Распределить* диалогового окна *Выровнять и распределить* можно выбрать опции для равномерного распределения объектов с учетом их ширины и высоты либо для равномерного распределения центров объектов. Распределение объектов может выполняться относительно опорных точек выделенных объектов или краев страницы документа.

7.2.2. Приложения машинной графики

Одним из последних достижений в области инструментальных средств для решения прикладных задач является **MathCad** — физико-математический пакет прикладных программ, в по-

следнюю версию которого включена система искусственного интеллекта **SmartMath** (разработка NASA), позволяющая выполнять математические вычисления не только в числовой, но и в аналитической форме.

Пакет прикладных программ MathCad предназначен для:

— проведения расчетов с действительными и комплексными числами;

— решения линейных и нелинейных уравнений и систем уравнений;

— упрощения, развертывания и группировки выражений;

— транспонирования, инвертирования (обращения) матриц и нахождения детерминанта (определителя);

— построения двухмерных и трехмерных графиков.

— оформления научно-технических текстов, содержащих сложные формулы;

— дифференцирования и интегрирования, аналитического и численного;

— проведения статистических расчетов и анализа данных.

Графическая среда MathCad позволяет записывать математические формулы в привычной виде, гибко и выразительно представлять данные графически.

Основное окно программы MathCad показано на рис. 7.22. Документ MathCad состоит из областей различного типа. **Текстовые области** создаются щелчком на кнопке с буквой **A** панели инструментов. **Математические области** возникают, если щелкнуть левой кнопкой мыши на свободном месте рабочего окна: появляется красный крестик — визир, фиксирующий место ввода формулы. Области на экране легко можно перетаскивать с помощью мыши или перемещать с помощью команд <*Cut*> и <*Insert*> в меню *Edit*.

Большинство математических формул записывается в рабочем документе MathCad так же, как на листе бумаги. Знаки арифметических операций вводятся с помощью клавиш <+>, <–>, <*>, </>.

Для ввода скобок, определяющих порядок выполнения арифметических операций, используется клавиша <*Space*> (*пробел*).

В большинстве случаев система тут же выдает ответ после нажатия на клавишу <=> с клавиатуры или с 1-й палитры операторов. В среде MathCad знак «=» означает числовой, а знак

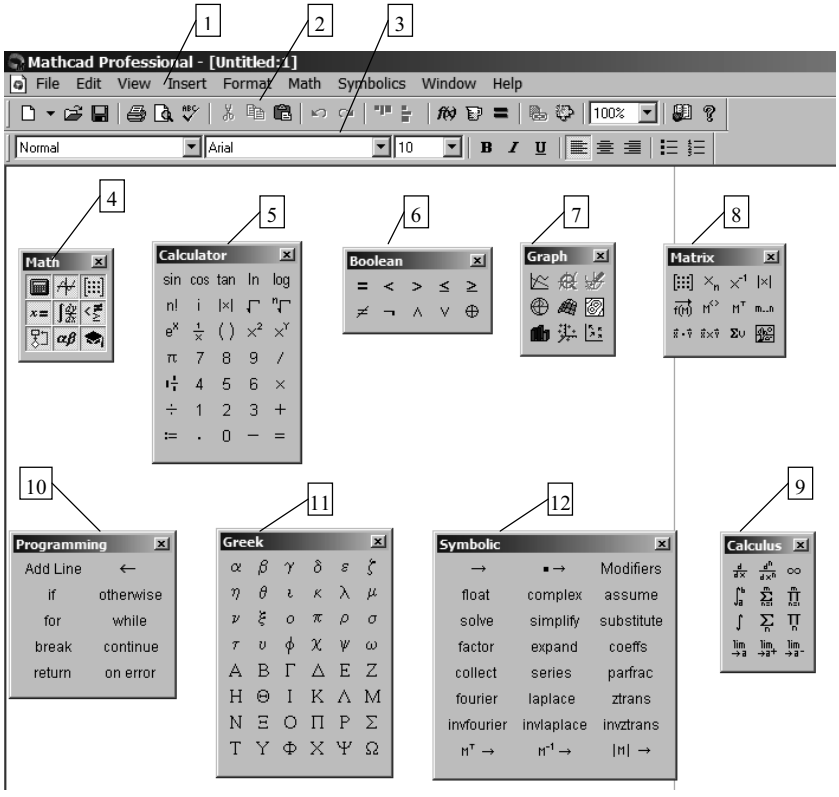


Рис 7.22. Рабочее окно программы MathCAD
1 — строка меню; *2* — панель инструментов *Стандартная*; *3* — панель инструментов *Форматирование*; *4* — панель управления *Математическая*; *5* — панель управления *Арифметическая*; *6* — панель управления *Вычисление*; *7* — панель управления *Графическая*; *8* — панель управления *Матрица*; *9* — панель управления *Исчисление*; *10* — панель управления *Программирование*; *11* — панель управления *Графический алфавит*; *12* — панель управления *Аналитические вычисления*

«→» (стрелка вправо) — символьный вывод значения переменной, функции, выражения.

Если последовательно вводить: 27/5+11=, в результате получится: 27/(5 + 11) = 1,6875. А если вводить: 27/5<пробел>+11=, в результате получится: 27/5 + 11 = 16,4.

При вводе более сложных операций используют кнопки палитр операторов MathCad, находящиеся на экране слева. Для перехода от одной палитры операторов к другой надо щелкнуть на цифре над палитрой.

Стандартные математические функции (такие, как cos, sin, arcsin, log, exp) можно вводить посимвольно или вставлять из прокручивающегося списка. Чтобы вызвать прокручивающийся список встроенных функций MathCad, нужно выбрать пункт *Insert Function* из меню *Math*.

Для редактирования выражения следует щелкнуть левой кнопкой мыши правее элемента выражения, подлежащего изменению, а затем нажать клавишу <*Backspace*> и ввести нужный элемент. Для немедленного пересчета значения выражения надо щелкнуть левой кнопкой мыши в стороне от выражения. Все вычисления могут производиться с высокой точностью — число значащих цифр задается из меню системы и практически не ограничено.

В математическом процессоре MathCad заложен последовательный принцип расчетов: значения всех переменных, которые используются в математическом выражении, должны быть определены заранее. Символ определения «:=» (его можно ввести с 1-й палитры операторов или нажав клавишу <:>) позволяет определять переменные и функции:

$$a := 5 \quad (a - 8) * (a + 3) = -24.$$

Важно следить за тем, чтобы все переменные и функции были определены левее и/или выше тех выражений, где они используются.

MathCad позволяет строить семь видов двухмерных и трехмерных графиков. На каждом из двухмерных графиков может одновременно находиться до 16 различных кривых, имеющих по шесть атрибутов. Можно создавать собственные библиотеки графических элементов, размещать в рабочем документе MathCad произвольные графические изображения.

Для построения графика надо определить с помощью кнопки *m...n* диапазон независимой переменной, а затем создать область графика с помощью кнопки внизу 1-й палитры. После этого вводятся выражения, откладываемые по осям X и Y (в средние поля ввода на соответствующих осях). Для каждой оси может быть введено несколько выражений.

Документ MathCad, в котором совмещены текст, графика и формулы, выглядит как страница научной статьи или учебника; при этом формулы являются «живыми» — стоит внести изменения в любую из них, как MathCad пересчитает результаты, перерисует графики и т. д. Можно анимировать график, записав его эволюцию при изменяющихся значениях параметров, а затем произвести мультипликацию со звуковым сопровождением.

Документы MathCad могут быть особым образом «сшиты» в электронные книги. При этом они, сохраняя все свои свойства, оказываются организованными в структуру, обладающую гипертекстовыми ссылками, навигацией, контекстным поиском, открывающимися окнами и т. д.

Доступ к таким электронным книгам может осуществляться по локальным и глобальным сетям: MathCad имеет средства для выхода в Internet и загрузки документов с помощью Internet-протокола.

В системе имеются разнообразные способы ввода числовых данных: с клавиатуры, из других приложений (например, из электронных таблиц), с использованием технологии OLE или DDE или буфера обмена, непосредственно из файлов, с применением разнообразных функций файлового доступа.

ПРАКТИЧЕСКИЕ РАБОТЫ

Работа № 1. Построение графиков функций

Цель работы: освоить построение графиков функций в декартовой системе координат.

Задание: требуется построить графики функций $f(x) = -\frac{4}{x}$ и $g(x) = \frac{1}{x}$, при изменении x от -10 до $+10$ с шагом 0,5.

Методика выполнения работы:

1. Выведите на экран панели инструментов, необходимые для работы:

— для вывода панели 1 выполните команду *F10 → View → Math Palette;*

— щелчком на третьей слева пиктограмме панели 1 выведите на экран панель 3 *Math Palette (Графика)* для работы с графиками;

— щелчком на первой пиктограмме панели 1 выведите на экран панель 4 *Arithmetic Palette (Счет),* предназначенную для набора различных математических формул (рис. 7.23).

2. Установите курсор на рабочем поле и введите с клавиатуры**:** f(x):= –4/x. Нажмите клавишу <*Enter*>. На экране появится: $f(x) = -\frac{4}{x}$.

3. Формирование вектора значений. Введите с клавиатуры: x: –10;10. Нажмите клавишу <*Enter*>. На экране появится: x:= –10..10. При такой записи шаг изменения аргумента берется по умолчанию равным единице. Если такая точность не устраивает пользователя, то указывается первое (минимальное) значение аргумента, затем, через запятую, — второе, равное первому значению плюс шаг, а далее, после нажатия клавиши <;> (*точка с запятой*), — верхний предел диапазона изменения ар-

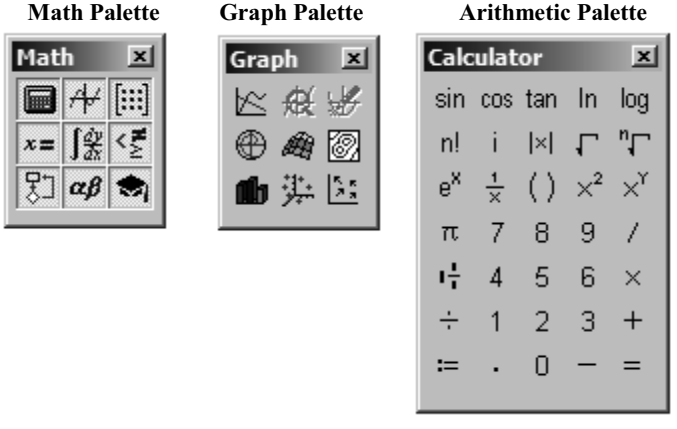


Рис 7.23. Панели инструментов, используемые при построении графиков функций

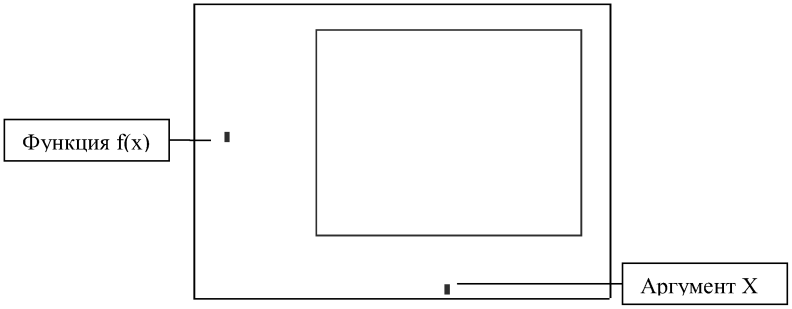


Рис 7.24. Заготовка графика функции

гумента *x*. Так, для задания шага изменения аргумента, равного 0,5, необходимо набрать: x:= –10, –9.5;10.

4. **Построение графика** начинается с вывода на экран дисплея **заготовки графика** — двух вложенных прямоугольников с **черными квадратиками** у левой и нижней сторон (рис. 7.24):

— щелкните левой кнопкой мыши в рабочей области экрана в предполагаемой точке расположения верхнего левого угла рисуемого графика;

— щелкните на кнопке *Графики* панели инструментов 3 *Graph Palette*, или выполните команду *Graph, X-Y Plot* в меню *Insert,* или нажмите комбинацию клавиш <*Shift*> + <*2*>.

5. Заполните заготовку графика **именем функции** и **именем аргумента**:

— щелчком мыши установите курсор в точку *Функция f(x)* — черный квадратик у левой стороны прямоугольника;

— наберите: f(x);

— щелчком мыши установите курсор в точку *Аргумент x* — черный квадратик у нижней стороны прямоугольника;

— наберите: x;

— нажмите клавишу <*Enter*>. График появится на экране (рис. 7.25).

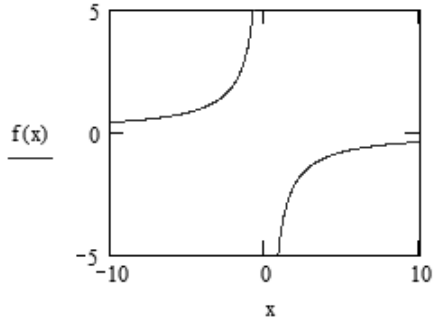


Рис. 7.25. График функции $f(x) = -\frac{4}{x}$

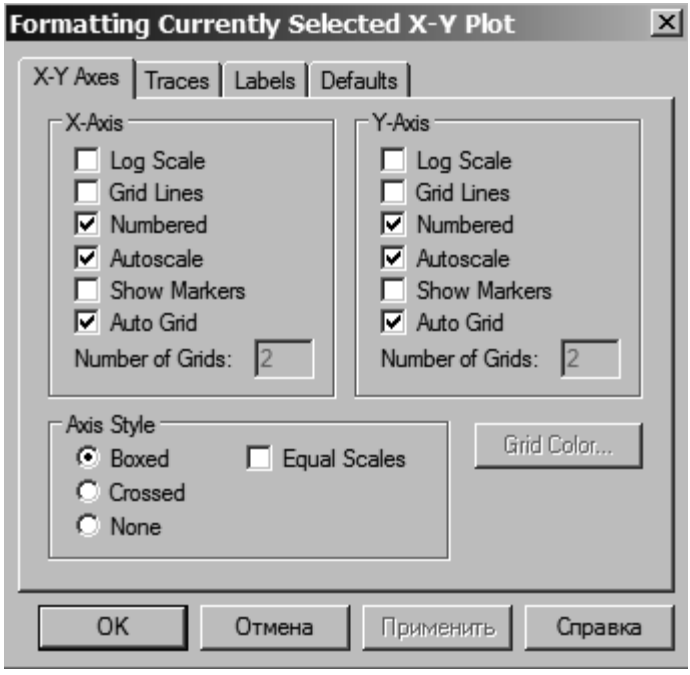


Рис 7.26. Вкладка диалогового окна *Форматирование графика — Редактирование осей*

6. Для оформления графика координатными осями выполните следующие действия:

— дважды щелкните мышью на графике — появится диалоговое окно *Formatting Currently Selected X-Y Plot;*

— выберите вкладку *X-Y Axes* (рис. 7.26);

— в поле выбора *Axes Style* щелкните на кнопке *Crossed* для представления графика с изображением осей координат;

— в поле *X-Axis* уберите флажок *Auto Grid*; в поле ввода *Number of Grids* введите число 5, что означает разметку оси *X;*

— в поле *Y-Axis* уберите флажок *Auto Grid,* в поле ввода *Number of Grids* введите число 4.

7. Измененный график появится на экране (рис. 7.27).

8. Чтобы модернизировать график**:**

— установите курсор на графике и двумя щелчками мыши вызовите диалоговое окно *Formatting Currently Selected X-Y Plot;*

— в диалоговом окне выберите вкладку *Traces* (рис. 7.28);

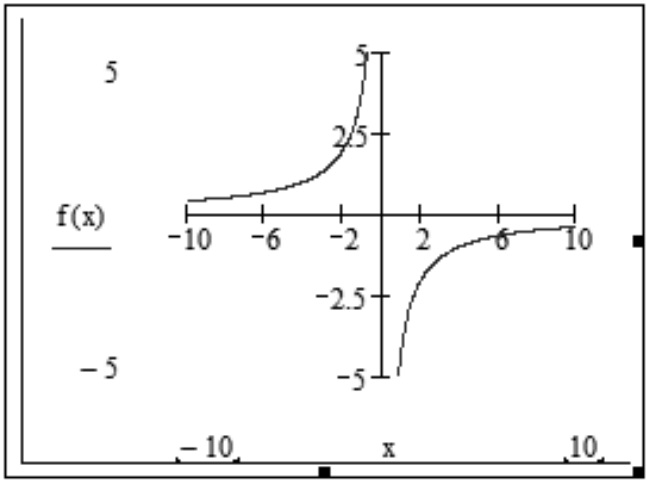


Рис. 7.27. График функции $f(x) = -\frac{4}{x}$ после редактирования осей координат

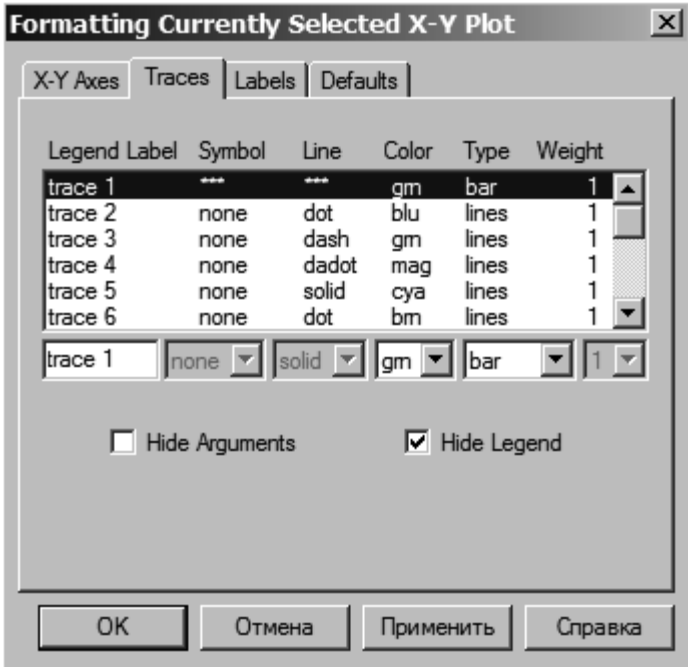


Рис 7.28. Вкладка диалогового окна *Форматирование графика — Изменение вида графика*

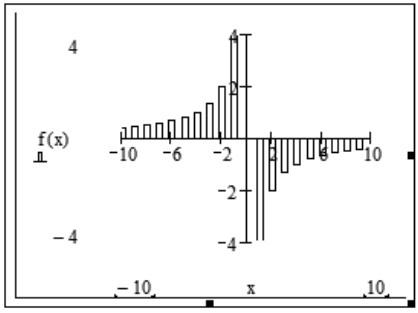


Рис. 7.29. График функции $f(x) = -\frac{4}{x}$ после редактирования вида графика

— в однострочном поле *Color* установите цвет — **grn** (зеленый);

— в однострочном поле *Type* установите: **bar** (прямоугольник);

— щелкните на кнопке *OK*. Новое представление графика изображено на рис. 7.29.

9. Чтобы добавить новый график к существующему:

— установите курсор на рабочем поле в строке справа от функции $f(x)$;

— введите с клавиатуры: g(x):(1/x);

— нажмите клавишу <*Enter*>. На экране появится: $g(x):=\frac{1}{x}$;

— выделите график;

— щелкните на графике функции. Установите курсор справа от f(x);

— введите с клавиатуры запятую. Курсор перейдет на следующую строку;

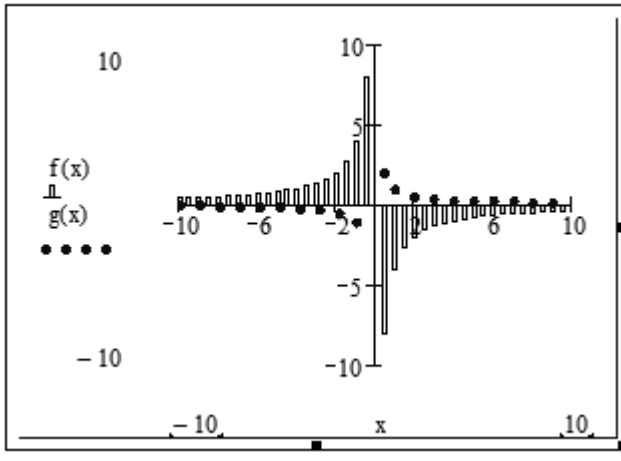


Рис. 7.30. График функций $f(x) = -\frac{4}{x}$ и $g(x) = \frac{1}{x}$

— наберите на клавиатуре: g(x). Нажмите клавишу <*Enter*>;

— измените цену деления на оси *X*, изменив строку x:= –10..10 на x:= –10, –9.5..10;

— линию графика $g(x)$ изобразите по своему усмотрению, используя диалоговое окно *Formatting Currently Selected X-Y Plot*. На рис. 7.30 выбран точечный график (*Type* — **points**, *Weight* — **3**) голубого (**blue**) цвета.

Работа № 2. Решение системы уравнений

Цель работы: изучить правила решения системы уравнений с помощью пакета MathCad.

Задание: решив систему уравнений

$$\begin{cases} 7x_1 - x_2 - 4x_3 = 2, \\ -6x_1 + 6x_2 + x_2 = 1, \\ -4x_1 + x_2 + 5x_3 = 2, \end{cases} \qquad (1)$$

найти значения x_1, x_2, x_3.

Методика выполнения работы: вначале требуется обозначить переменные. Примем следующие обозначения:

A — матрица коэффициентов системы;

B — вектор свободных членов;

X — вектор результатов решения.

1. Выведите на экран панели инструментов, необходимые для работы. Для этого:

— выполните команду *F10* → *View* → *Math Palette* для вывода панели 1 (рис. 7.31);

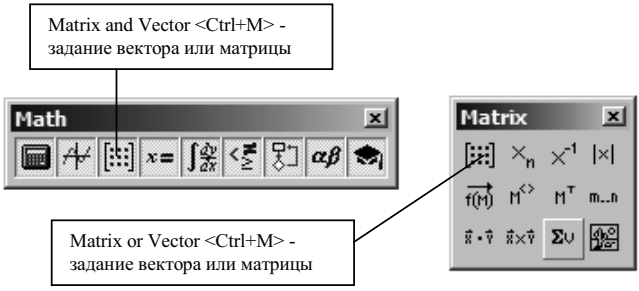


Рис 7.31. Панели инструментов пакета MathCAD, используемые при работе с матрицами

— щелкните на четвертой слева пиктограмме этой панели для вывода на экран панели 2 *Matrix or Vector.*

2. Задайте матрицу **A** коэффициентов системы:

— в левом верхнем углу рабочего поля окна документа щелкните левой кнопкой мыши;

— наберите прописными буквами: ORIGIN:=1, чтобы начать индексацию результатов решения системы с номера 1;

— щелкните в рабочей области окна в месте расположения матрицы;

— введите с клавиатуры имя матрицы: **A**;

— щелкните на пиктограмме с изображением стилизованной матрицы на панели 2;

— задайте размер матрицы **A**: 3×3;

— щелкните на кнопке *OK;*

— на экране появится заготовка для матрицы: **A:=** $\begin{vmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{vmatrix}$

— введите значения элементов матрицы: установите курсор на верхнем левом черном прямоугольнике матрицы и введите значение: 7;

— нажмите клавишу <*Tab*>. Курсор переместится на одну ячейку вправо; последовательно введите значения:

$$\begin{vmatrix} 7 & -1 & -4 \\ -6 & 6 & 1 \\ 4 & 1 & 5 \end{vmatrix}$$

Нажмите клавишу <*Enter*>.

3. Установите курсор в рабочей области окна под матрицей **A** и, последовательно выполняя указания пункта 2, введите с клавиатуры имя матрицы: **B:**;

— нажмите комбинацию клавиш <*Ctrl*> + <*M*> и задайте размер матрицы **B:** — 3×1;

— введите матрицу: **B:** $\begin{vmatrix} 2 \\ 1 \\ 2 \end{vmatrix}$;

— нажмите клавишу <*Enter*>.

4. Создайте обратную матрицу $\mathbf{A^{-1}}$:

— введите с клавиатуры: **A**. Нажмите комбинацию клавиш <*Shift*> + <*6*> и введите: **–1**. Введите с клавиатуры знак «=»;

— нажмите клавишу <*Enter*>.

5. Для **нахождения корней** системы линейных уравнений требуется вычислить определитель **det**. В изучаемом пакете вычисление определителя осуществляется записью следующего выражения: **det := | A |**. Наберите его, используя соответствующую пиктограмму панели инструментов 2. Выведите на экран полученное значение: **det=<*Enter*>**. Появится запись: **det=105**. Если значение определителя **det** не равно нулю, т. е. матрица коэффициентов **A** невырождена, задача имеет однозначное решение во всех случаях, и для любого вектора **B** найдется единственный вектор **X**, удовлетворяющий заданной системе уравнений (1).

6. Чтобы найти вектор **X**:

— введите с клавиатуры: **X:A ^ –1:**

— нажмите два раза клавишу <*Ж*> (*вправо*) и наберите: ***B:**

— нажмите клавишу <*Enter*>. На экране появится: $\mathbf{X:=A^{-1}\cdot B}$.

7. Введите с клавиатуры: **X=** и нажмите клавишу <*Enter*>. На экране появится результат решения: матрица 3 Г 1. Убедитесь, что все компоненты вектора **X** равны единице.

8. Чтобы вывести на экран значения $\mathbf{X_1, X_2, X_3}$:

— введите с клавиатуры: X[1= и нажмите клавишу <*Enter*>. На экране появится: **X1**= (значение первого корня);

— введите с клавиатуры: X[2= и нажмите клавишу <*Enter*>. На экране появится: **X2**= (значение второго корня);

— так же получите и значение третьего корня.

Работа № 3. Решение нелинейных уравнений

Цель работы: поиск корня нелинейного уравнения с помощью функции Root пакета MathCad.

Задание: определить значение корня уравнения $\boldsymbol{x + \lg(x) + \ln(x/10) = 11.1}$ с точностью 10^{-3}, если известно, что $x \in [10;11]$.

Методика выполнения работы: многие уравнения **не имеют** аналитических решений. Они могут решаться **численными ме-**

тодами с заданной погрешностью. Для простейших уравнений вида $F(x) = 0$ решение находится с помощью функции **root (Выражение, Имя_переменной)**. Функция **root** возвращает значение переменной, при котором выражение становится равным нулю, т. е. $F(x) = 0$.

Для решения уравнения надо сначала задать **начальное значение** переменной. Функция всегда имеет **несколько решений**, поэтому выбор решения определяется начальным значением переменной.

Введем условные обозначения:

***f*(*x*)** — функция, приравниваемая к 0;

TOL — точность вычисления;

x — начальное значение переменной;

***x*1** — приближенное решение функции ***f*(*x*)**.

1. Выведите на экран панели инструментов, необходимые для работы:

— для вывода панели 1 (см. рис. 7.31) выполните команду *<F10> → View → Math Palette;*

— щелчком на первой пиктограмме панели 1 выведите на экран панель 4 *Arithmetic Palette (Счет),* предназначенную для набора различных математических формул.

2. Задание вида функции и условий:

— в рабочей области экрана с клавиатуры введите функцию: **f(x):= x + lg(x) + ln(x/10) — 11.1**;

— в рабочей области экрана введите точность: **TOL:=10^{-3}** и начальное значение переменной: **x:=10**;

— функции, которые не заданы в MathCad в явном виде, необходимо выразить через другие функции, например: **lg(x)= =ln(x)/ln(10)**.

3. Решение нелинейного уравнения с помощью функции **root**:

— в рабочей области экрана введите с клавиатуры: **x1:=root(f(x),x)**;

— нажмите клавишу *<Enter>*.

4. Вывод на экран значения **x1**:

— введите с клавиатуры: **x1= <Enter>**. На экране появится приближенное значение **x1**. По умолчанию количество знаков после запятой равно трем;

— если требуемая точность превышает 10^{-2}, необходимо изменить формат вывода результата на экран, выполнив команду *<F10> → Format → Number → Displayed Precision.*

7.2.3. Элементы численных методов

Численные методы — это методы приближенного или точного решения задач математики, основанные на построении конечной последовательности действий над конечным множеством чисел. Численные методы являются предметом изучения *вычислительной математики.* Для решения и исследования задач прикладной математики в настоящее время принята и представляется наиболее эффективной следующая методология.

1. Составляется *математическая модель (ММ)* процесса. Обычно ММ формулируется в терминах интегральных и дифференциальных уравнений функций непрерывного аргумента. Это так называемая континуальная ММ — экономный способ описания конечной совокупности (ансамбля) дискретных объектов, когда количество этих объектов становится большим (таковой ММ является, например, интегро-дифференциальное уравнение Больцмана, описывающее поведение ансамбля частиц в некотором объеме).

2. Осуществляется переход от континуальной к дискретной ММ. Он заключается в замене функций непрерывного аргумента функциями дискретного аргумента, а уравнений континуальной ММ — конечно-разностными уравнениями. При этом интеграл заменяется конечной суммой, а производная — разностным отношением. В результате, как правило, приходят к системе большого количества уравнений с большим количеством неизвестных (дискретная ММ).

3. Составляется численный метод, или *вычислительный алгоритм,* для решения полученной системы уравнений с некоторой указанной точностью.

4. Производится программирование, т. е. перевод вычислительного алгоритма на язык ЭВМ.

Указанные этапы составляют «технологическую цепочку» современной вычислительной математики. Содержащиеся

в ней переходы от исходной совокупности дискретных объектов (например, ансамбль молекул в заданном объеме газа) к континуальной модели, а затем к другой системе дискретных объектов (разностная сетка) необходимы для уменьшения объема перерабатываемой информации. Так, в нашем примере ансамбль очень большого количества частиц (10^{24}) заменяется совокупностью ячеек сетки в значительно меньшем количестве (10^6).

В экономике конечная совокупность дискретных объектов, как правило, также непосредственно описывается дискретной моделью.

При переходе от континуальной к дискретной ММ производится замена континуального *оператора* соответствующим дискретным. Так, дифференциальный оператор заменяется разностным, интеграл — суммой и т. д. Такая замена приводит к появлению погрешности аппроксимации. В практических вычислениях следует учитывать также *погрешность округления,* возникающую в ЭВМ при операциях над машинными числами, имеющими ограниченное количество *значащих цифр.* Учитывая это, получают реальный вычислительный алгоритм, в отличие от теоретического вычислительного алгоритма. Это привело к необходимости проводить анализ погрешности округления и гарантированных оценок точности реальных вычислений.

Существуют разнообразные численные методы для решения многих важных классов задач.

Основой численного метода решения задач математической физики является дискретизация задачи с последующим сведением полученных, вообще говоря, нелинейных уравнений к системе линейных алгебраических уравнений. В связи с этим численные методы по способу дискретизации можно подразделить на проекционные и конечно-разностные, а по способу решения линейной системы — на прямые и итерационные.

В последнее время большое значение приобретают нерегулярные системы, к которым приводят задачи о потоках в различного рода сетях (тепловых и энергетических, в трубопроводах). Здесь теория разностных схем сочетается с *теорией графов.*

7.2.4. Создание диалоговых программ

Проблема эффективного взаимодействия человека и ЭВМ возникла с появлением первых вычислительных машин. На начальном этапе внедрения ЭВМ непосредственно с ними работали только программисты и операторы, общение же специалистов в сфере управленческой деятельности происходило опосредованно: путем выдачи программистам заданий на разработку программ решения задач и последующего получения распечаток готовых результатов вычислений на ЭВМ.

Расширение сфер использования ЭВМ, совершенствование их аппаратных и программных средств, а главное — появление и широкое внедрение ПК привели к тому, что в процесс непосредственного взаимодействия с ЭВМ был вовлечен массовый пользователь.

Специфика работы пользователей ПК требует создания таких средств и методов общения с вычислительной системой, благодаря которым они (пользователи), не зная архитектуры и принципов функционирования ЭВМ, не владея профессиональными приемами программирования, могли бы удовлетворять свои информационные потребности в ходе взаимодействия с машиной.

Человек и компьютер в процессе решения различных задач, связанных с обработкой информации, могут взаимодействовать в различных режимах. Различают **пакетный** и **интерактивный** (запросный, диалоговый) режимы взаимодействия пользователя с ЭВМ. Сами же ЭВМ также могут работать в различных режимах: одно- и многопрограммном, разделения времени, реального времени, телеобработки и т. д. При этом разработчики имеют целью удовлетворение потребности пользователей в максимально возможной автоматизации решения разнообразных задач.

Пакетный режим был наиболее распространен в практике централизованного решения задач, когда большой удельный вес занимали задачи отчетности о производственно-хозяйственной деятельности экономических объектов разного уровня управления.

Организация вычислительного процесса при пакетном режиме строилась без доступа пользователя к ЭВМ. Его функции

ограничивались подготовкой исходных данных по комплексу организационно взаимосвязанных задач и передачей их в центр обработки, где формировался пакет, включающий задание ЭВМ на обработку программы, исходные, нормативно-расценочные и справочные данные. Пакет вводился в ЭВМ и реализовывался в автоматическом режиме без участия пользователя и оператора, что позволяло минимизировать время выполнения заданного набора задач. При этом работа ЭВМ могла проходить в однопрограммном или многопрограммном — что предпочтительнее — режиме, так как обеспечивалась параллельная работа основных устройств машины. В настоящее время пакетный режим реализуется применительно к электронной печати.

Интерактивный режим предусматривает непосредственное взаимодействие пользователя с информационно-вычислительной системой и может носить характер запроса (как правило, регламентированного) или диалога с ЭВМ.

Запросный режим необходим пользователям для взаимодействия с системой через значительное число абонентских терминальных устройств (в том числе удаленных на довольно большое расстояние от центра обработки) с целью решения оперативных задач справочно-информационного характера (например, резервирование билетов на транспорте, номеров в гостиничных комплексах и т. п.). ЭВМ в подобных случаях реализует систему массового обслуживания, работает в режиме разделения времени, при котором несколько независимых абонентов (пользователей) с помощью устройств ввода-вывода имеют в процессе решения своих задач непосредственный и практически одновременный доступ к ЭВМ. Запросный режим позволяет дифференцированно, в строго установленном порядке предоставлять каждому пользователю время для общения с ЭВМ, а после окончания сеанса отключать его.

Диалоговый режим открывает пользователю возможность непосредственно взаимодействовать с вычислительной системой в допустимом для него темпе работы, реализуя повторяющийся цикл выдачи задания, получения и анализа ответа. При этом ЭВМ сама может инициировать диалог, сообщая пользователю последовательность шагов (представление меню) для получения искомого результата.

Обе разновидности интерактивного режима (запросный, диалоговый) основываются на работе ЭВМ в режимах реального времени и телеобработки, которые разработаны в развитие режима разделения времени. Поэтому обязательными условиями функционирования системы в этих режимах являются:

— постоянное хранение в запоминающих устройствах ЭВМ необходимой информации и программ и поступление лишь в минимальном объеме исходной информации от абонентов;

— наличие у абонентов соответствующих средств связи с ЭВМ для обращения к ней в любой момент времени.

Диалоговый режим взаимодействия пользователя и ЭВМ возник как альтернатива пакетному, когда потребовалось обеспечить возможность оперативного вмешательства пользователя в процесс электронной обработки информации. На практике же весьма часто можно наблюдать совместное использование этих режимов, помогающее, за счет их частных преимуществ организовать более эффективную технологию решения задач на ЭВМ.

Диалог, или режим диалога, — это активный обмен информационными сообщениями между участниками процесса, когда прием, обработка и выдача сообщений производятся в реальном масштабе времени. В литературе диалоговый режим, нередко реализуемый в системах коллективного пользования на основе разделения времени, называют **интерактивным режимом**.

Диалог может быть парным, когда число его участников равно двум, и множественным — при большем числе участников. При парном диалоге его субъектами могут выступать как люди, так и технические средства в виде ЭВМ. Причем возможны такие сочетания, как «человек — человек», «человек — ЭВМ» и «ЭВМ — ЭВМ». Однако общие принципы реализации диалога при различных комбинациях остаются неизменными.

Типы диалога и формы его реализации на ЭВМ. Анализ различных диалоговых систем, реализующих вычислительные, информационные, справочные и обучающие функции, а также комплексные задачи, показал, что наиболее распространенными типами организации диалога являются меню, шаблоны, команды, естественный язык.

Диалог типа «меню» благодаря наглядности имеет широкое распространение при решении самых различных задач на раз-

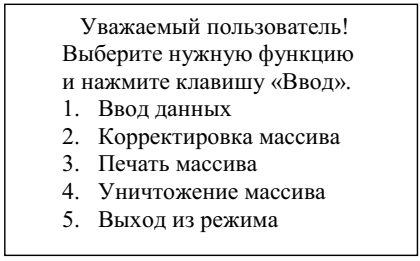


Рис. 7.32. Содержание кадра

ных классах и моделях ЭВМ. Данный тип диалога наиболее удобен для конечного пользователя, так как средством отображения сообщений между человеком и ЭВМ является экран видеотерминала. Шаг диалога начинается, как правило, с выдачи системой некоторого входного сообщения, альтернативный ответ на которое позволяет продолжить или завершить диалог (рис. 7.32). Завершается шаг диалога после ввода пользователем ответного сообщения и его обработки системой. Выход из диалоговой системы возможен в случае прохождения всей цепочки шагов, обеспечивающих полное решение задач, путем прерывания диалога по инициативе пользователя и выхода из системы.

Машинная реализация диалога типа «меню» осуществляется несколькими способами, однако при всех вариантах в качестве входного сообщения на экран видеотерминала выводится некоторое подмножество функций системы, реализация которых возможна в текущем состоянии диалога.

Наиболее простым частным случаем диалога типа «меню» является режим ответа ДА/НЕТ, когда пользователю в качестве альтернативы предлагается всего два варианта ответа: ДА или НЕТ.

Машинная реализация этого режима может быть осуществлена одним из следующих способов, когда:

— ответ набирается полностью: ДА или НЕТ;

— ответ задается с помощью первых букв: Д или Н (Y или N);

— одна из альтернатив (ДА или НЕТ) предполагается по умолчанию и реализуется клавишей ввода, а другая — клавишей отмены.

Выбрать требуемую функцию пользователь может различными способами (в зависимости от реализации операции в программной среде диалоговой системы):

— набором на клавиатуре требуемой директивы или ее сокращенного обозначения;

— набором на клавиатуре номера необходимой функции;

— подведением курсора в строку экрана с нужной пользователю функцией;

— нажатием функциональных клавиш, запрограммированных на реализацию данной функции.

Диалог типа «шаблон» — это инициируемый системой режим взаимодействия конечного пользователя и ЭВМ, на каждом шаге которого система воспринимает только синтаксически ограниченное входное сообщение пользователя в соответствии с заданным форматом. Как и при диалоге типа «меню», инициирующая роль в начале диалога принадлежит ЭВМ. Возможные варианты ответа пользователя ограничиваются форматами, предъявляемыми ему на экране видеотерминала. Поэтому гибкость пользователя такой системой относительно невысока, хотя и выше, чем при описанных выше схемах организации диалога; резко снижается и операционная сложность реализации диалога данного типа на машине.

Диалог типа «шаблон» может быть реализован несколькими способами, самыми распространенными из которых являются:

— указание системой на экране дисплея формата вводимого пользователем сообщения;

— резервирование места для сообщения пользователя в тексте сообщения системы на экране терминала.

Диалог типа «шаблон» обычно применяется для ввода данных, значения которых или понятны (например, поле для записи даты, фамилии, названия предприятия и т. д.), или являются профессиональными терминами, известными пользователю по его предметной области.

Следует различать *жесткий шаблон,* когда количество вводимых пользователем символов обязательно должно соответствовать числу разрядов, выделенных тем или иным способом на экране дисплея, и *свободный шаблон,* когда задается предельно допустимое поле, в которое вносится конкретное значение (например, фамилия работающего при формировании справочника).

Частным случаем диалога данного типа является режим, называемый **простым запросом.** Наибольшее распространение он получил в автоматизированных системах сбора и формирования массивов данных. При этом пользователю предоставляется возможность вводить массив, состоящий из более чем одного

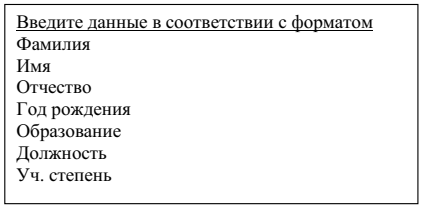


Рис. 7.33. Информационный кадр прямого вопроса

сообщения, по формату, заданному системой. Диалог в этом случае сводится лишь к одному шагу, а в качестве сообщений на экране компьютера могут быть выведены анкетные данные работающих либо номенклатура материальных ценностей и т. п. Примером может служить приведенный на рис. 7.33 информационный кадр для создания машинной личной карточки работника предприятия.

Диалог типа «команда» инициируется пользователем, при этом выполняется одна из его директив (команд), допустимых на данном шаге диалога. Перечень допустимых команд, как правило, отсутствует на экране дисплея, однако его можно вызвать на экран с помощью специальной директивы или функциональной клавиши на ПЭВМ (обычно клавиши <*F1*>). Если пользователь ввел ошибочную команду, т. е. отсутствующую в списке, или команду с нарушениями формата или синтаксиса, выдается предупредительное сообщение, и система остается в начале текущего шага диалога, ожидая его продолжения.

Взаимодействием на естественном языке называется диалог такого типа, когда и инициирование, и ответ со стороны пользователя ведутся на языке, близком к естественному (как в плане используемого словаря, так и в плане построения сложных языковых конструкций). Пользователю предоставляется возможность свободно формулировать нужную ему задачу, однако с употреблением обусловленных программной средой слов и фраз, а также с заранее установленным синтаксисом языка. В связи с неоднозначностью естественных языков система должна иметь возможность задавать вопросы, уточняющие формулировку пользователя и предметную область рассматриваемой проблемы. При диалоге подобного типа пользователь должен быть готов к тому, что диалоговая система может не распознать его запрос с первого раза, а потому потребуется его повторение или уточнение. Одной из разновидностей такого диалога является *речевое общение с системой*.

На практике в системе, как правило, используется сочетание нескольких типов диалога, а не только один тип в чистом виде. Это повышает гибкость системы, упрощает ее, делает диалог более разнообразным и интенсивным.

В дружелюбной диалоговой системе, ориентированной на непрофессионального пользователя, человек, работая на ЭВМ, должен воспринимать не весь диалог, а лишь визуальное (иногда и звуковое) его отображение. Большая часть программных средств управления диалогом должна быть скрыта от пользователя, а ему предоставляется минимум для поддержания внешнего диалога.

Аппаратные средства современных ЭВМ позволяют отображать на экране дисплея графические и цветные изображения движущихся объектов различной природы, сложные структуры данных и лишь простым нажатием клавиш управлять событиями, возникающими на экране. Кроме того, во многие профессионально-ориентированные ППП по рекомендации психологов введены игровые моменты с целью предоставления пользователю кратковременного отдыха, например в случае переутомления и возникающих из-за этого ошибок.

Важным для всех категорий пользователей программных средств, работающих в режиме диалога, является включаемая в них в обязательном порядке система помощи и средств обучения (HELP), существенно ускоряющая как процесс освоения диалоговой системы, так и процесс работы. Другое требование к системе — простота работы с ППП. Освоение основных функций любого пакета диалогового типа не должно требовать специальных знаний в области языков программирования, архитектуры ЭВМ и пр.

Поскольку пользователю приходится работать с различными диалоговыми программными системами, целесообразно закладывать в них некоторое однообразие и определенные сложившиеся традиции (например, использование функциональных клавиш *<F1>* и *<F10>* соответственно для вызова помощи и выхода из системы, применение управляющих клавиш или их комбинаций для управления состоянием процесса вычислений и т. д.).

Кроме того, диалоговые системы должны использовать достижения эргономики и современного дизайна. Привлекатель-

ное внешнее оформление диалога (по гамме цветов, графическому оформлению, расположению различных объектов, многооконность и т. п.) не только настраивает пользователя на работу с системой, но и делает ее приятной и менее утомительной.

В настоящее время все более широкое распространение в качестве формы диалогового взаимодействия пользователя с ЭВМ приобретает его работа в мультимедийной среде, где под термином «мультимедиа» понимается объединение взаимодействия различных каналов передачи информации (в первую очередь, изобразительной и звуковой) от машины к человеку и обратная связь: действия человека должны существенно влиять на ход событий, воспроизводимых компьютерной системой.

Технология создания диалоговых систем может быть представлена в виде определенной последовательности шагов:

— на ***первом шаге*** анализируется подлежащая решению задача, выявляются ее характерные признаки, формулируются методы и средства ее решения, уточняются источники информации, объемы информационных массивов, вычислительные ресурсы и т. д.;

— ***второй шаг*** — отображение этих представлений в виде интерактивного взаимодействия, синтезированного из простейших компонентов процедур, т. е. составление укрепленной логической схемы;

— ***третий шаг*** — формирование полного подробного сценария диалогового решения задачи на основе построенной структуры человеко-машинного диалога и его отображение в виде информационной базы, информационных кадров, обеспечивающее информационно-вычислительный процесс по заданному сценарию;

— ***четвертый шаг*** — обеспечение автоматизированного ведения гибкого диалога путем написания программы на алгоритмическом языке или на базе логико-лингвистического процессора.

Важной проблемой при создании диалоговых систем является обеспечение *реактивности,* т. е. достаточно быстрой циркуляции сообщений как между функциональными задачами (программами), так и между задачами и пользователем. Обмен сообщениями должен осуществляться в режиме реального вре-

мени, без зависаний системы, которые психологически угнетающе действуют на партнеров по диалогу, а в системе «человек — ЭВМ» — на пользователя.

Для конечных пользователей — специалистов управления диалоговая система должна быть достаточно прозрачной и требовать от них лишь выполнения обычных действий в соответствии с их служебными обязанностями. При этом пользователь должен получать от системы необходимые разъяснения (подсказки) по содержанию требуемых от него действий.

Возможность взаимодействия пользователя с системой с помощью языка диалога может быть обеспечена двумя способами:

— ***первый способ*** предполагает, что с системой общается пользователь-неспециалист. Поэтому ему нет необходимости знать весь объем команд (директив, сообщений) системы и правил их исполнения. Он должен знать свой пароль, свое меню, т. е. обладать лишь достаточным объемом знаний, система же сама должна подсказать пользователю, что он может отвечать в той или иной ситуации;

— при ***втором способе*** общение с диалоговой системой осуществляет пользователь-специалист (например, администратор системы). В этом случае диапазон выполняемых им функций более широк и может включать такие операции, как:

— изменение состава технологических цепочек, значений параметров, правил запуска и т. д.;

— вывод на экран текстов справочных сообщений, форм таких сообщений и внесение в них новой информации;

— преобразование данных из одной формы представления в другую;

— собственно операции над данными.

В число функций при втором способе общения с диалоговой системой входят:

— контроль состояния всей диалоговой системы и правильности взаимодействия процессов;

— контроль состояния процессов;

— обеспечение надежности функционирования системы в целом и отдельных процессов;

— адаптация и развитие системы.

При разработке диалога управления процессами применяются файловые системы организации обмена данными и их

преобразования в памяти ЭВМ, утилиты для работы с программными модулями, специальные языковые средства.

При переходе к массовому применению ПЭВМ, работающих в режиме диалога, появляется возможность отказа от использования традиционных — бумажных — носителей информации, работа с которыми трудоемка, требует специальной подготовки операторов и нарушает индивидуальность автоматизированной обработки информации; применение таких ПЭВМ в местах возникновения информации (на складах, в цехах, в функциональных управленческих отделах и т. д.) позволяет автоматизировать процесс изготовления и заполнения первичной документации.

Диалоговая технология способствует приближению средств вычислительной техники к пользователю, однако она требует применения повышенных мер защиты информации от несанкционированного доступа. Доступ к машине имеют пользователи, выполняющие различные функции, связанные с решением задач, за которые они несут ответственность как юридические лица. Чтобы исключить порчу данных и программ, располагающихся на жестких «винчестерских» дисках ПЭВМ, в структуру диалога включаются дополнительные модули, обеспечивающие защиту данных от несанкционированного доступа.

Другой аспект диалоговой технологии на ПЭВМ — юридический. Автоматизация составления первичной документации приводит к ужесточению контроля за работой пользователей, заполняющих первичный документ (в данном случае термин «документ» толкуется несколько шире и может подразумевать файл или фрагменты базы данных). Поскольку в его заполнении могут принимать участие несколько лиц, каждый работник аппарата управления, имеющий доступ к машине, несет юридическую ответственность за корректность вносимой в документ (файл) информации, что заверяется подписями в нем.

В условиях сетевого использования ЭВМ автоматизируется доставка исходной информации к месту ее основной обработки, а результатной — к месту ее использования. Применение каналов связи обеспечивает ввод данных со скоростью, в несколько раз более высокой, чем при вводе перфоносителей или при простой передаче магнитных носителей в центр обработки.

Одновременно за счет создания централизованных и распределенных баз данных достигается сокращение объема вводимой информации и общего времени на ее обработку.

В качестве примера приведем фрагмент общей схемы, характеризующей диалоговый режим решения экономических и других управленческих задач на ПЭВМ (рис. 7.34). Диалог пользователя с системой начинается с проверки его полномочий. Для этого в систему заранее должны быть введены коды или пароли пользователей, которые могут периодически меняться для обеспечения более надежной защиты данных. Включив терминал или ПЭВМ, пользователь или получает от системы запрос на ввод своего кода, или по своей инициативе сразу вводит свой пароль. Автоматическая сверка введенного пароля с перечнем паролей, хранящихся в памяти ЭВМ, позволяет идентифицировать пользователя, определить его полномочия и приоритеты и либо подключить его к системе, либо отказать ему в работе с ней.

Поскольку данные могут вводиться с использованием первичных документов, составленных вне эксплуатации ПЭВМ, в схему включены блоки *11—14*, предусматривающие и эту операцию. Выбор типа работы, ее конкретного вида и операции по ней предусмотрены блоками *8—10*. Операции по вводу исходных данных, их записи на магнитный носитель и контролю представлены на схеме блоками *15—20*. Если исходные данные введены верно, то они записываются в рабочие массивы (блоки *21—22*), в противном случае производится корректировка информации (блоки *23—24*).

Занесенная в рабочие массивы информация хранится до момента наступления решения необходимых задач. При этом пользователь, выбрав требуемую задачу на основании имеющихся и отобранных программных средств и информационных массивов, переходит к этапу собственно ее решения (блоки *25—28*). Последующие технологические операции, не представленные на приведенном фрагменте схемы, сводятся к отображению результатной информации на экране или в виде печатной копии, контролю полученной информации, ее оформлению и использованию. Завершение одной задачи позволяет перейти к блоку *25*, выбрать новую задачу и обеспечить ее решение.

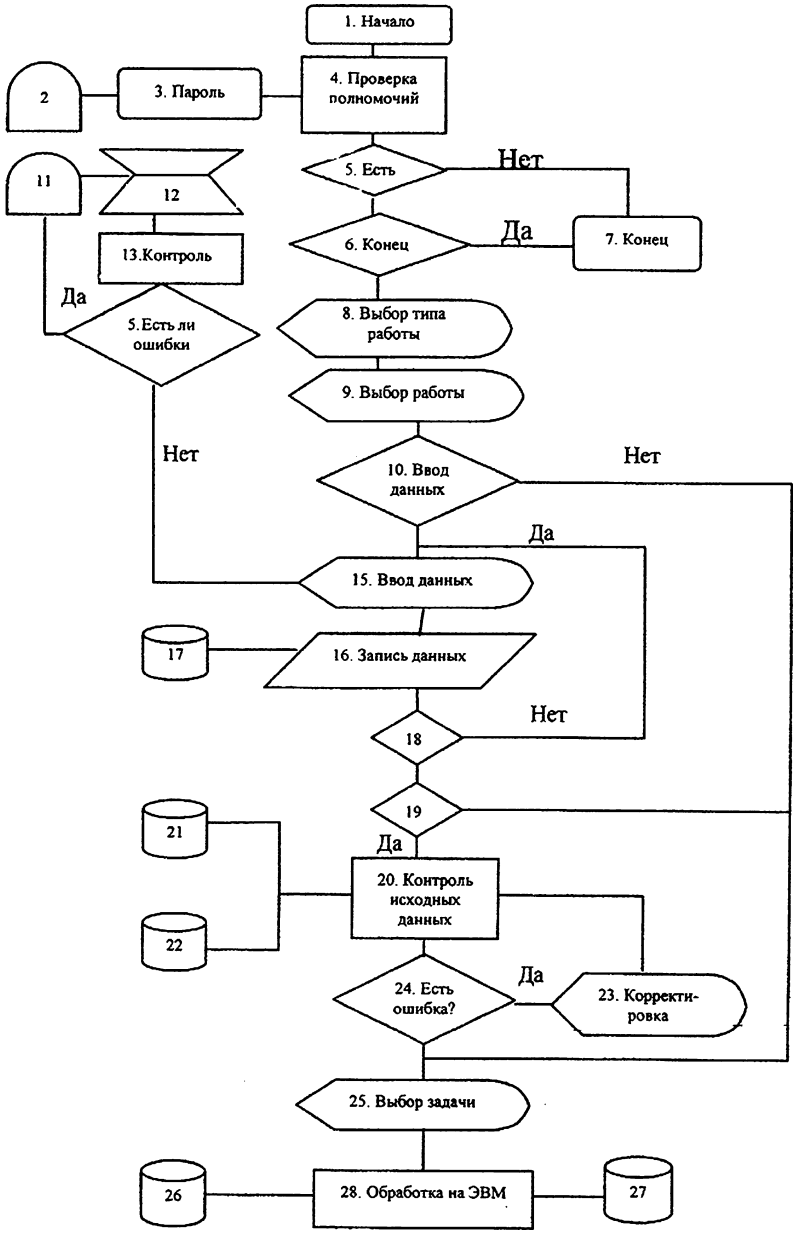


Рис. 7.34. Схема технологии реализации диалога при решении экономических задач

7.2.5. Жизненный цикл программного обеспечения

Жизненный цикл программного обеспечения включает в себя шесть этапов:

— анализ требований;
— определение спецификаций;
— проектирование;
— кодирование;
— тестирование;
— сопровождение.

Рассмотрим эти этапы.

Анализ требований. При разработке программного обеспечения такой анализ исключительно важен. Ошибки, допущенные на этом этапе, даже при условии безупречного выполнения последующих этапов могут привести к тому, что разработанный программный продукт не будет соответствовать требованиям практики, сферы его применения. Для создания конкурентоспособного продукта в процессе анализа требований необходимо получить четкие ответы на следующие вопросы: что должна делать программа? В чем состоят реальные проблемы, разрешению которых она должна способствовать? Что представляют собой входные данные? Какими должны быть выходные данные? Какими ресурсами располагает проектировщик?

Определение спецификаций. Этот этап можно рассматривать как формулировку выводов, следующих из результатов предыдущего этапа. Требования к программе должны быть представлены в виде ряда спецификаций, явно определяющих рабочие характеристики будущей программы. В их число могут входить скорость выполнения, объем потребляемой памяти, гибкость применения и др.

Проектирование. На данном этапе создается общая структура программы, которая должна удовлетворять спецификациям; определяются общие принципы управления и взаимодействия между различными компонентами программы.

Кодирование. Задача этого этапа — перевод на язык программирования конструкций, записанных на языке проектирования.

Тестирование. На этапе тестирования производится всесторонняя проверка программ. Суть тестирования более подробно будет рассмотрена ниже.

Сопровождение. Это этап эксплуатации системы. Каким бы изощренным ни было тестирование программ, к сожалению, в больших программных комплексах чрезвычайно тяжело устранить абсолютно все ошибки. Устранение обнаруженных при эксплуатации ошибок — первейшая задача этапа сопровождения. Однако это далеко не все. Проводимый в ходе сопровождения анализ опыта эксплуатации программы позволяет обнаруживать узкие места или неудачные проектные решения в тех или иных частях программного комплекса. В результате может быть принято решение о проведении работ по совершенствованию разработанной системы. Сопровождение может также включать в себя проведение консультаций, обучение пользователей системы, оперативное снабжение их информацией о новых версиях системы и т. п. Качественное проведение этапа сопровождения в большой степени определяет коммерческий успех программного продукта.

Рассмотрим этап *тестирования* программы более подробно. Существуют три аспекта проверки программы:

— направильность;

— навычислительную сложность;

— наэффективность реализации.

Проверка правильности удостоверяет, что программа делает в точности то, для чего она была предназначена. Математическая безупречность алгоритма не гарантирует правильности его перевода в программу. Аналогично ни отсутствие диагностических сообщений компилятора, ни разумный вид получаемых результатов не дают достаточной гарантии правильности программы. Обычно проверка на правильность заключается в разработке и проведении набора тестов. Кроме того, для расчета программ иногда можно сверить получаемые решения с уже известными. В общем случае нельзя дать общего решения для проведения проверки программы на правильность.

Проверка вычислительной сложности заключается в экспериментальном анализе сложности алгоритма или экспериментальном сравнении двух и более алгоритмов, решающих одну и ту же задачу.

Проверка эффективности реализации имеет целью отыскание способа «заставить» правильную программу работать быстрее или расходовать меньше памяти. Чтобы улучшить программу,

пересматриваются результаты реализации в процессе построения алгоритма. Не рассматривая всех возможных вариантов и направлений оптимизации программ, приведем здесь некоторые полезные способы увеличения скорости выполнения программ.

П е р в ы й способ основан на следующем правиле: сложение и вычитание выполняются быстрее, чем умножение и деление; целочисленная арифметика быстрее арифметики вещественных чисел. Таким образом, $X + X$ лучше, чем $2X$, а $i + 0{,}5$ хуже, чем $(2i + j)0{,}5$ или $(i + i + j)0{,}5$. При выполнении операций над целыми числами следует помнить, что, благодаря применению двоичной системы счисления, умножение на числа, кратные двум, можно заменить соответствующим количеством сдвигов влево. Поэтому 10A выполняется дольше, чем Ashl 3 + Ashll.

В т о р о й способ заключается в удалении избыточных вычислений.

Пример.

```
Root 1: = (–b + sqrt(sqr(6) – 4ac))/(2.0a);
Root 2: = (–b – sqrt(sqr(6) – 4ac))/(2.0a).
```

Лучшим решением является следующее:

```
DenomA: = a + a;
DenomC: = c + c;
Discrim: = sqrt(bb – DenomA * DenomC);
Rootl: = (–b + Discrim)/DenomA;
Root 1: = (–b – Discrim)/DenomA.
```

Легко увидеть, что в первом случае потребовалось выполнить четыре операции сложения/вычитания, шесть операций умножения, две операции деления, два вызова функции *sqr* и два вызова функции *sqrt;* во втором — пять операций сложения/вычитания, две умножения, две деления и одно обращение к функции *sqrt.*

Т р е т и й способ проверки эффективности реализации основан на способности некоторых компиляторов строить коды для вычисления логических выражений так, что вычисления прекращаются, если результат становится очевидным. Например, в выражении A or B or C, если A имеет значение «истина», то переменные B и C уже не проверяются. Таким образом, можно сэкономить время, разместив переменные A, B, C так, чтобы

первой стояла переменная, которая вероятнее всего будет истинной, а последней — та, которая реже всего принимает истинное значение.

Однако следует быть осторожными и в следующем примере: Root (A) or B or C. Root (A), может, и чаще принимает значение «истина», но представляет собой вызов функции, возможно, выполняющей сложные и длительные вычисления. Тогда может оказаться, что запись B or C or Rool {A) является более эффективной.

Ч е т в е р т ы й способ — исключение циклов.

Пример.

Фрагмент:

```
for i: = 1 to 1000 do
A[i]: =0;
for i:= 1 to 1000 do
for i: = 1 to 10 do
A[i]: =A[i] + C[i,j]
```

можно переписать так:

```
for i: = 1 to 1000 do begin
B: = C[i,1 ];
for j = 2 to 10 do
A[i]: = B B:= B + C[i,j];
End.
```

В данном примере выигрыш достигается, во-первых, за счет уменьшения количества циклов (два, а не три), а во-вторых — за счет того, что с введением временной переменной B уменьшено количество операций вычисления адресов элементов массива.

П я т ы й способ — развертывание циклов.

Пример.

Запись:

```
for i:= 1 to 1000 do
for j: = 1 to 3 do
A[i]:=A[i] + C[i,j]
```

можно переписать так:

```
for i: = 1 to 1000 do
A[i]: = A[i] + C[i,1] + C[i,2] + C[i,3].
```

Выигрыш в скорости вычислений налицо.

Это далеко не полный перечень способов оптимизации. Здесь приведены лишь самые очевидные из них. Следует, кроме того, заметить, что не всегда стоит увлекаться погоней за быстродействием, так как при этом чаще всего ухудшается удобочитаемость программ. В том случае, когда выигрыш получается мизерный, вряд ли стоит предпочесть его в ущерб ясности и читабельности программы.

Тема 7.3

ОБЗОР И КРАТКАЯ ХАРАКТЕРИСТИКА СОВРЕМЕННЫХ ЯЗЫКОВ И СРЕДСТВ ПРОГРАММИРОВАНИЯ

Напомним: первые поколения ЭВМ строились на классических принципах, сформулированных американским математиком Джоном фон Нейманом в 1946 г., когда начались разработки цифровых ЭВМ с программным управлением. Одним из основных принципов фон Неймана является принцип хранимой программы. Под программой вычислительной машины понимается описание алгоритма решения задачи, заданное на языке вычислительной машины. Таким образом, языки программирования — это формальные языки общения человека с ЭВМ, предназначенные для описания совокупности инструкций, выполнение которых обеспечивает правильное решение требуемой задачи, т. е. для описания подлежащих обработке данных (информации) и алгоритмов (программ). Основная роль языков программирования заключается в планировании действий по обработке информации. Любой язык программирования основан на системе понятий, на основе которой человек может выражать свои соображения.

Теоретическую основу языков программирования составляют алгоритмические языки. В настоящее время для ЭВМ разработано значительное количество языков программирования. Они отличаются друг от друга различными свойствами, а значит, и областью применения. Существуют разные подходы к классификации языков программирования. На рис. 7.35 представлена схема, в которой языки программирования систематизированы по возможностям и ориентации на конкретную сферу применения.

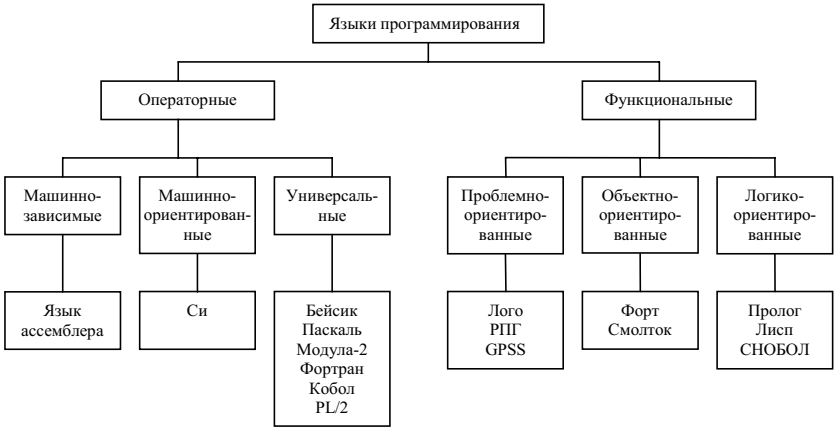


Рис 7.35. Классификация языков программирования

Первыми используемыми языками программирования были машинные языки, задаваемые системами команд ЭВМ. Класс машинно-зависимых языков представлен **Ассемблером**. Язык Ассемблера делает доступными все программно-управляемые компоненты ЭВМ, поэтому он применяется для написания программ, явно использующих специфику конкретной аппаратуры. Ассемблер наиболее трудоемкий язык программирования, из-за его низкого уровня не удается построить средства отладки, которые существенно снизили бы эту трудоемкость. Программирование на языке Ассемблера оперирует терминами команд аппаратной части машины и поэтому требует от программиста детального знания технических компонентов компьютера.

Каждый компьютер имеет такую систему программирования. Как правило, она разрабатывается и поставляется фирмой — изготовителем ЭВМ. Применение машинных языков требует излишней детализации пути решения задачи, поскольку даже наиболее современные ЭВМ способны понимать лишь относительно простые инструкции, тогда как человек способен концентрировать детали в выражениях более общего характера.

Использование языка Ассемблера ограничивается областью системного программирования: он применяется для программирования микропроцессоров, разработки операционных систем или их отдельных компонентов, программ обмена меж-

ду системным блоком и периферийным устройством (драйвером) и т. д. Поэтому с развитием программирования появились языки, ориентированные на определенные проблемные области, соответствующие более высокому уровню абстракции. Эти языки получили название **алгоритмических языков высокого уровня (АЯВУ)**. Их отличительной чертой является способность выразить задачу лаконично и сжато, обеспечивая программиста возможностью более четко представлять свои задачи и разрабатывать эффективные методы их решения. Преобразование описания задачи на АЯВУ в описании на машинном языке осуществляется специальной программой, называемой **транслятором**. Уже имеется целый ряд языков программирования, ориентированных на те или иные области применения. Наиболее близкими к машинным языкам являются машинно-ориентированные языки, позволяющие использовать особенности ЭВМ для повышения эффективности программ.

К классу машинно-ориентированных языков можно отнести язык **Си**. Он является результатом попытки объединить достоинства низкоуровневых возможностей АЯВУ. Язык Си часто называют языком Ассемблера со встроенными структурами данных. Использование структур данных позволяет более систематично подходить к реализации задачи на языке Си и сокращает объем текстов разрабатываемых программ. Особенность данного языка заключается в максимальном использовании возможностей конкретной вычислительной архитектуры на основе битовых операций, функций и назначений. Благодаря этому программы на языке Си компактны и работают очень быстро. Однако синтаксис этого языка достаточно сложен, поэтому чтение текстов программ на нем требует определенного навыка. Язык Си первоначально был ориентирован прежде всего на разработку системных программ. Он, в частности, послужил главным инструментом для создания операционных систем MS DOS и UNIX. В настоящее время язык Си применяется главным образом для создания системных и прикладных программ, в которых скорость работы и объем памяти служат основными параметрами.

Большинство языков программирования являются машинно-независимыми, что позволяет использовать однажды записанную на таких языках программу на различных ЭВМ. В зави-

симости от подхода к описанию задачи языки программирования подразделяют на процедурно-ориентированные и проблемно-ориентированные. Процедурно-ориентированные языки предоставляют программисту средства задания и детального определения последовательности действий, которые ЭВМ должна выполнить для решения задачи, т. е. средства, определяющие, как должна действовать машина. Проблемно-ориентированные (или функциональные) языки определяют задачу в терминах функции, которую надо выполнить машине, т. е. определяют, что должно быть сделано.

Процедурно-ориентированные языки программирования обычно имеют операторную структуру и содержат средства выражения характерных алгоритмических действий, таких, как вычисление выражений, проверка условий, циклические вычисления, включения программ и др. Проблемно-ориентированные языки предоставляют пользователям средства определения более или менее широкого набора функций, подлежащих исполнению. Следует заметить, что четкой границы между этими категориями языков нет, и в пределах одного языка могут сочетаться средства языков обеих категорий.

Большинство современных языков программирования ориентировано на тот или иной круг задач. Так, наиболее широко используемыми в мировой практике вычислений являются языки:

— Кобол и PL/1 — для обработки экономической информации;

— Фортран (исторически первый язык высокого уровня) — для решения инженерных и научных задач;

— Бейсик, Паскаль, Лого — для обучения программированию;

— Пролог, Лисп — для решения задач искусственного интеллекта;

— Симула-1, Смолток — для описания задач моделирования дискретных событий;

— Модула-2, Ада — для управления реальными объектами;

— Снобол, Комит и др. — для манипуляции с текстами.

Наиболее широко представлен класс универсальных языков программирования. Среди них можно выделить такие популярные языки высокого уровня, как Бейсик, Паскаль, Фортран, Кобол, Модула-2, PL/1 и др.

Исторически одним из самых распространенных языков стал **Бейсик**. Это объясняется прежде всего тем, что Бейсик прост в освоении и использовании. Написать на нем небольшую программу в 20—30 строк и тут же получить результат ее работы можно буквально за несколько минут. В язык Бейсик, как правило, встраиваются удобные функции для работы с экраном дисплея, клавиатурой, магнитными накопителями, принтером, коммуникационными каналами, что позволяет относиться к нему как к продолжению аппаратуры ПЭВМ. Чтобы освоить какую-нибудь особенность или режим работы аппаратных средств, проще всего написать и выполнить соответствующую программу на этом языке.

Для различных типов ПЭВМ, которые существенно отличаются друг от друга, были разработаны соответствующие версии языка Бейсик. Для ПЭВМ типа IBM PC и совместимых с ними ПЭВМ наиболее удачной считается версия фирмы Microsoft. Она обеспечивает использование языка Бейсик для решения задач обработки больших массивов данных (работа с файлами), инженерно-технических и научных расчетов (с помощью большого набора математических функций), обработки текстов (благодаря эффективной работе со знаковыми последовательностями), а также для решения комплексных задач (за счет создания оверлейных программных структур). Появление мощных компиляторов, таких, как, например, Quik Basic и Visual Basic фирмы Microsoft, поставило Бейсик в ряд с другими языками высокого уровня и придало ему дополнительную популярность.

Язык **Паскаль** можно считать одним из самых распространенных, хотя он и создавался как учебный. Со временем Паскаль зарекомендовал себя в качестве отличного инструмента для решения серьезных задач, так как его разработчик специально конструировал язык, позволяющий создавать хорошо структурированные программы. Широкое применение языка Паскаль пользователями IBM PC и совместимых с ними ПЭВМ обусловлено появлением его оригинальной версии — **Турбо Паскаль** фирмы Borland International. Турбо Паскаль характеризуется такими важными особенностями, как полноэкранное редактирование и убавление, графика, звуковое сопровождение и связи с дисковой ОС. Система программирования

на языке Турбо Паскаль сама является резидентной программой. Она позволяет пользователю вводить его программы и выполнять их немедленно, не тратя времени на компилирование. Вместе с тем Турбо Паскаль создан как средство быстрой разработки не очень больших программ (с числом строк до 500). Более длинные программы приходится сегментировать и использовать оверлейные структуры.

Стремление к созданию подлинно универсального и эффективного инструмента программирования привело к разработке нового языка — **Модула-2**. Он предложен автором языка Паскаль Н. Виртом. Основная цель, поставленная при его создании, — обеспечить высокоуровневыми языковыми средствами коллективную разработку надежных, эффективных инструментов программирования и использовать возможности аппаратуры. Таким образом, язык Модула-2 призван заполнить нишу между языками Паскаль и Си. В него целиком вошли все удачные средства и конструкции языка Паскаль, высокоуровневое представление низкоуровневых возможностей (например, оставаясь на уровне АЯВУ, можно оперировать машинно-независимыми регистрами и отдельными командами).

Фортран — первый язык программирования высокого уровня — активно используется и на современных ПК. Близость его конструкции к традиционной архитектуре ЭВМ (имеется в виду традиционная фоннеймановская архитектура) сделала Фортран очень популярным. Применяется этот язык главным образом при разработке прикладных систем, ориентированных на научные исследования, инженерные задачи, автоматизацию проектирования и другие области, где накоплены обширные библиотеки стандартных программ.

Язык **Кобол** был разработан специально для решения экономических задач. В отличие от языка Фортран, Кобол дает возможность составлять более удобочитаемые программы, которые могут быть понятны и непрограммисту. В программах на этом языке особенно проявляется самодокументируемость, что облегчает их исправление и усовершенствование, а при обработке данных сложной структуры он бывает эффективнее языка Паскаль. Кобол широко распространен на больших и средних машинах, а на ПЭВМ используется мало, хотя фирмой

Microsoft разработано несколько его версий для операционных систем MS DOS и Windows. Наиболее удачной версией языка Кобол на сегодняшний день является Кобол/U, в который встроены средства генерации отчетов с использованием языка РПГ.

Фирмой IBM в развитие идей языков Фортран, Алгол и Кобол был предложен язык **PL/1**, получивший особенно широкое применение на больших машинах. PL/1 разрабатывался как универсальный язык программирования, поэтому располагает богатым набором средств обработки цифровой и текстовой информации. Однако эти достоинства делают его весьма сложным для изучения и использования.

Класс проблемно-ориентированных языков программирования представлен гораздо скромнее, чем класс универсальных языков, к которым можно отнести языки Лого, РПГ и систему программирования OPSS.

Язык **Лого** создан с целью обучения школьников основам алгоритмического мышления и программирования. Это диалоговый процедурный язык, реализованный на основе интерпретатора с возможностью работы со списками и на их основе с текстами и оснащенный развитыми графическими средствами, доступными для детского восприятия. Лого реализован для большинства ПЭВМ, применяемых в школах. Хотя он был разработан задолго до того, как началось массовое использование ПК, именно с их появлением Лого приобрел популярность.

Язык **РПГ**, или генератор отчетов, включает многие понятия и выражения, связанные с машинными методами составления отчетов и проектирования форм выходных документов. Он имеет ограниченную область применения и используется главным образом для печати отчетов, записанных в одном или нескольких файлах базы данных. Многие системы, реализованные на ПЭВМ типа IBM PC и располагающие языком РПГ, имеют также и другой язык высокого уровня (прежде всего Кобол), применимый для таких вычислительных процедур, для которых РПГ не подходит.

Интересные возможности предоставляет система программирования **GPSS** фирмы Westi, ориентированная на моделирование систем с помощью событий. В терминах этого языка

легко описываются и исследуются класс моделей массового обслуживания и другие системы, работающие в реальном масштабе времени.

В последние годы внимание разработчиков программного обеспечения все более привлекает объектно-ориентированный подход к программированию, идеология которого наиболее полно реализована в языках Форт и Смолток.

Форт своеобразный язык: он сочетает в себе свойства операционной системы, интерпретатора и компилятора одновременно. Основной его чертой является открытость, которая позволяет строить новые определения функций на базе ранее определенных. Программист может легко добавить новые операции, типы данных или определения основного языка. Форт позволяет поддерживать многозадачный режим работы, широко используя принцип реентерабельности (одновременного доступа) программ. Структура этого языка позволяет создавать очень компактные трансляторы. Программирование на языке Форт требует специальных навыков, поэтому он находит применение при решении сложных задач имитационного моделирования, в графических системах, в системах искусственного интеллекта как средство построения баз знаний, в промышленных разработках.

Объектно-ориентированный язык **Смолток** предназначен для решения нечисловых задач и находит широкое применение при проектировании систем искусственного интеллекта. Программа на этом языке состоит из множества функциональных автономных объектов и средств взаимодействия между ними. Каждый объект содержит некоторую структуру данных и процедуры манипулирования ими. Внешние свойства объекта проявляются в функциях, которые объект выполняет для других объектов. В современных реализациях языка Смолток широко используется многооконный режим.

К функциональным языкам программирования можно отнести языки Лисп, Пролог и Снобол.

Язык **Лисп** — прекрасное инструментальное средство для построения программ с использованием методов искусственного интеллекта. Имеется несколько реализаций Лисп-трансляторов для ПЭВМ различных классов. Особенность этого языка заключается в удобстве динамического создания новых

объектов. В качестве порождаемых программой объектов могут фигурировать и сами программы, которые внешне ничем не отличаются от данных. Это обусловливает возможность построения адаптирующихся и самоизменяющихся программ. Память в языке Лисп используется динамически: когда создается новый объект, то для него из незанятой памяти берется столько ячеек, сколько нужно для хранения всех элементов. При этом не требуется никакого заблаговременного резервирования памяти, как в других языках (например, в языке Паскаль). При уничтожении объекта занятая им память автоматически освобождается.

Язык **Пролог** получил широкую известность в связи с японским проектом создания вычислительных систем пятого поколения. Одной из первых реализаций был интерпретатор микро-Пролог фирмы Programming Logic Associates для машин фирмы IBM. Затем появилась мощная система программирования Турбо Пролог, разработанная фирмой Borland International для этих же машин и предназначенная для создания широкого класса систем искусственного интеллекта, в том числе персональных экспертных систем. От стандартного языка Пролог версия Турбо Пролог отличается прежде всего наличием встроенных средств типизации данных и большей структурированностью исходных текстов программ.

Язык **Снобол** располагает мощными средствами манипулирования строками и сравнения с образцом. В основном он используется для обработки текстов на естественном языке и применяется в экспертных системах. Известны некоторые версии языка Снобол, реализованные для ПК, но его применение ограничено сферой искусственного интеллекта.

Нетрудно заметить, что такого алгоритмического языка высокого уровня, который был бы идеальным для всех случаев, не существует. Наиболее важная задача программного обеспечения заключается в том, чтобы определить, какой язык является наилучшим в каждой конкретной ситуации. Нередко такой выбор диктуется очень простыми причинами — доступностью того или иного транслятора и умением составлять программы на данном языке. Однако если в распоряжении пользователя имеется достаточно большой выбор языков программирования, то необходимо учитывать следующие обстоятельства:

— назначение разрабатываемой программы (будет ли она использоваться временно или постоянно, будет ли модернизироваться и развиваться);

— время выполнения программы (имеется в виду соотношение вычислительных процедур и процедур ввода-вывода);

— ожидаемый размер программы (хватит ли памяти для реализации всей программы целиком или ее следует разделить на отдельные взаимодействующие модули);

— необходимость сопряжения разрабатываемой программы с другими пакетами или программами, в том числе составленными на других языках программирования;

— предусматривается ли возможность переноса программы на другие типы ЭВМ;

— основные типы данных, с которыми будет работать программа (целые и вещественные числа, строки, списки и другие типы структур);

— характер и уровень использования аппаратных средств (дисплея, клавиатуры, НМД и др.);

— возможность и целесообразность использования имеющихся стандартных библиотек программ, процедур, функций.

С учетом этих критериев возможности языков программирования могут весьма сильно различаться. Поэтому правильный выбор АЯВУ является непростой задачей. Для программиста или даже коллектива программистов характерно начинать использование ПЭВМ с языков Бейсик или Паскаль. На языке Бейсик (и его разновидностях) разработано довольно много хороших прикладных систем. Наличие компиляторов для языка Бейсик сделало его еще более привлекательным, так как позволяет быстро переходить от экспериментальной, интерпретируемой версии программы к окончательной. В последнее время появились разработки программ на этом языке для систем искусственного интеллекта и экспертных систем.

Создание информационных и вычислительных сетей и распределенных баз данных вызвало необходимость в специализированных языках программирования, обслуживающих эти области использования ЭВМ. Потребовались специальные средства организации человеко-машинного взаимодействия, диалога человека с ЭВМ (QBE, SQL).

* * *

Завершает изучение курса практическая работа по поиску информации в сети Internet.

ПРАКТИЧЕСКАЯ РАБОТА «ПОИСК ИНФОРМАЦИИ В СЕТИ INTERNET»

Цель работы: научиться осуществлять поиск информации в сети Internet с помощью поисковых систем.

Задание 1: поиск информации с использованием поисковой системы Yandex.

Методика выполнения задания:

1. Запустите Internet Explorer.

2. В адресную строку (рис. 7.36) введите адрес поисковой системы: www.yandex.ru

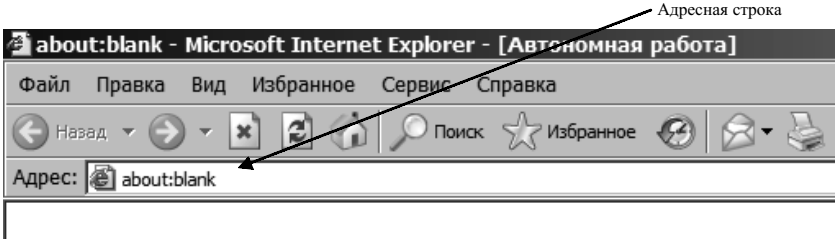


Рис. 7.36. Поиск информации в Internet

3. Если такой строки у вас нет, зайдите в меню *Вид* → *Панели инструментов* и поставьте галочку напротив слов *Адресная строка*.

4. После того как загрузится стартовая страница, в строку поиска введите с клавиатуры: «система счисления». Щелкните на кнопке *Найти*.

5. Под строкой поиска вы увидите, сколько всего страниц найдено. Запишите их количество в тетрадь. В одном окне поисковая система отображает список из 10 ссылок.

6. На рис. 7.37 показано, как выглядит список найденных страниц. В списке указаны заголовок найденной страницы, ад-

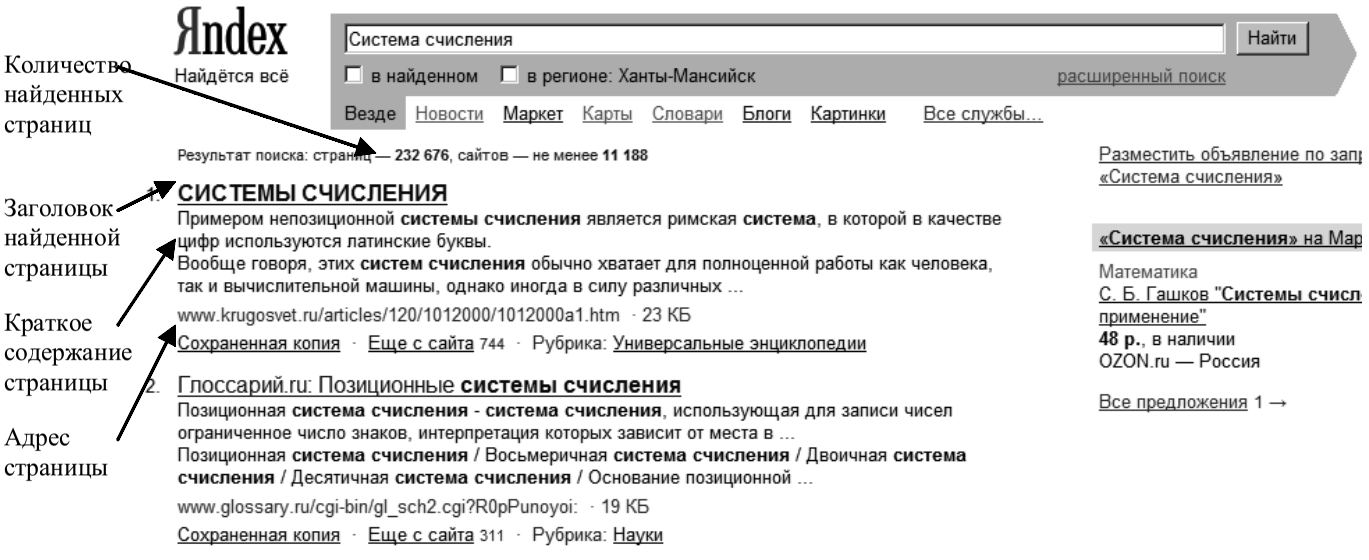


Рис. 7.37. Окно поисковой системы Yandex

рес и стро́ки из этой страницы, где встречаются слова, которые вы ввели в строку поиска.

7. Чтобы открыть нужную вам страницу и прочитать ее содержимое подробнее, щелкните на ссылке, и эта страница откроется в отдельном окне Internet Explorer.

8. Проведите расширенный поиск. Найдите примеры перевода из одной системы счисления в другую. Для этого задайте новый поиск среди найденных страниц (поставьте галочку в поле *В найденном,* в строку поиска введите слова: «примеры перевода», щелкните на кнопке *Найти*). Обратите внимание: количество найденных страниц изменилось.

9. Для перехода к списку из следующих 10 страниц щелкните на нужной кнопке: 2, или 3, или 4 и т. д. либо на слове *Следующая*.

10. Выберите подходящую страницу, скопируйте ее содержание в буфер обмена и вставьте в документ Microsoft Word.

Задание 2: поиск информации с использованием поисковой системы Google.

Методика выполнения задания:

1. Запустите Internet Explorer.

2. В адресную строку введите адрес поисковой системы: www.google.ru

3. В поле для ввода ключевых слов введите с клавиатуры: «система счисления». Щелкните на кнопке *Поиск в Google*.

4. Запишите в тетрадь количество найденных страниц.

5. Проведите расширенный поиск. Найдите примеры перевода из одной системы счисления в другую. Для этого задайте новый поиск среди найденных страниц (перейдите по ссылке *Поиск в найденном,* которая находится в конце страницы; в строку поиска введите с клавиатуры: «примеры перевода»; щелкните на кнопке *Поиск в найденном*).

6. Выберите подходящую страницу, скопируйте ее содержание в буфер обмена и вставьте в документ Microsoft Word.

Задание 3: сравнение результаты работы поисковых систем.

Сравните результаты работы двух поисковых систем по количеству найденных страниц и их соответствию запросу.

УПРАЖНЕНИЯ

Разработать алгоритм и составить программу на алгоритмическом языке Паскаль.

1. В интервале от *a* до *b* найти все парные простые числа. Парными простыми числами называют два простых числа, разность между которыми равна двум, например: 3 и 5, 11 и 13, 17 и 19.

2. Найдите все трехзначные числа, сумма цифр которых равна заданному числу *n*.

3. Число Армстронга — это такое число из *k* цифр, для которого сумма *k*-х степеней его цифр равна самому числу, например: $153 = 1^3 + 5^3 + 3^3$. Найти все числа Армстронга из двух, трех и четырех цифр.

4. Палиндром — это такое сочетание цифр, которые читаются одинаково слева направо и справа налево, например: 121, 55, 4884.

5. Найти все палиндромы, для которых их квадраты также палиндромы (в заданном интервале от *a* до *b*).

6. Счастливым будем считать такое число из шести цифр, в котором сумма левых трех цифр равна сумме правых трех цифр, например: 457961; $4 + 5 + 7 = 9 + 6 + 1 = 16$.

Найти все счастливые билеты и подсчитать их количество (номера билетов — от 0 до 999 999). Если в числе меньше шести цифр, то недостающие начальные цифры считаются нулями.

7. Дано натуральное число *n*. Среди чисел 1, ..., *n* найти все такие, запись которых совпадает с последними цифрами записи их квадрата, например: $6^2 = 36, 25^2 = 625$.

8. Найти наименьшее общее кратное двух заданных чисел.

9. Дано натуральное число *n*. Получить все пифагоровы тройки натуральных чисел, каждое из которых не превосходит *n*, т. е. все такие тройки натуральных чисел *a, b, c,* что $a^2 + b^2 + c^2$ ($a \leqslant b \leqslant c \leqslant n$).

10. Найти все совершенные числа в интервале от *a* до *b*. Совершенным называется такое натуральное число, которое равно сумме всех своих делителей, за исключением самого числа, например: $28 = 1 + 2 + 4 + 7 + 14$.

КОНТРОЛЬНЫЕ ВОПРОСЫ К РАЗДЕЛУ 7

1. **Что называется алгоритмическим языком или языком программирования?**
2. **Раскройте понятие «машинный язык».**
3. **Дайте определения понятий «алфавит языка», «лексика», «синтаксис языка».**
4. **Какие лексические структуры языка Паскаль вы знаете?**
5. **Что такое «типы данных»?**
6. **Перечислите операторы языка Паскаль.**
7. **Сформулируйте понятия «процедуры», «функции» и «рекурсии» языка Паскаль.**
8. **Для чего в языке Паскаль используются массивы, запись и файлы?**
9. **Как вы понимаете термин «машинная графика»?**
10. **С помощью каких прикладных программ реализуется машинная графика?**
11. **Назовите простейшие объекты векторной графики. Какими свойствами они обладают?**
12. **В результате каких операций можно получить сложный объект?**
13. **Какие операции можно провести над группой объектов?**
14. **С помощью каких операций можно модифицировать (изменить) форму простейших объектов?**
15. **В чем принципиальное отличие простого и художественного текстов? Кратко охарактеризуйте типы текста.**
16. **В чем заключается смысл динамической связи объектов текста?**
17. **Какие типы CorelDRAW вы знаете? Назовите параметры и эффекты, создаваемые с их помощью.**
18. **Как можно осуществить просмотр и выбор цвета в CorelDRAW?**
19. **Назовите основные средства преобразования и управления позиционированием объектов.**
20. **С помощью каких средств можно определить положение и изменить размер объектов на странице?**
21. **Для каких целей предназначен пакет прикладных программ MathCad?**
22. **Что такое численные методы?**
23. **В чем состоят особенности пакетного режима обработки данных?**
24. **Расскажите о принципах организации диалогового общения человека и ЭВМ.**
25. **Каким требованиям должны отвечать диалоговые системы обработки данных?**
26. **Назовите этапы создания диалоговых систем.**
27. **Проведите анализ различных типов машинного диалога.**
28. **Какие типы машинной реализации диалога вы знаете?**

29. **В чем состоят особенности диалоговой технологии решения задач на ПЭВМ?**
30. **Какими способами и методами обеспечивается в диалоге сохранность информации?**
31. **Приведите пример диалогового решения задачи на ПЭВМ.**
32. **Каковы критерии оценки диалоговых систем?**
33. **Из каких этапов состоит жизненный цикл программного обеспечения?**
34. **Что такое языки Ассемблера?**
35. **Каковы особенности алгоритмических языков высокого уровня?**
36. **Перечислите известные вам процедурно-ориентированные и проблемно-ориентированные языки программирования.**
37. **Как вы понимаете термин «функциональные языки программирования»?**
38. **Расскажите о роли информатики в информатизации общества.**

Заключение

Современное общество характеризуется небывалым ростом объема информационных потоков и в сфере экономики, и в социальной сфере. Особенно это касается отраслей промышленности, торговли, финансово-кредитной деятельности. В промышленности такой рост обусловлен увеличением объема производства, усложнением выпускаемой продукции, используемых материалов, технологического оборудования, расширением производства, внешних и внутренних связей экономических объектов. Рыночные отношения предъявляют повышенные требования к своевременности, достоверности и полноте информации, без чего немыслима эффективная маркетинговая, финансово-кредитная, инвестиционная деятельность.

К известным видам ресурсов (материальным, трудовым, энергетическим, финансовым) прибавился новый — информационный. Только на основе своевременного пополнения, накопления, переработки информационного ресурса, т. е. владения достоверной информацией, возможны рациональное управление любой сферой человеческой деятельности, принятие правильных решений. Особенно актуально это для сферы экономики. Применение современных ЭВМ дает возможность переложить трудоемкие операции на автоматические или автоматизированные устройства, которые могут работать со скоростью, превышающей скорость обработки информации человеком в миллионы раз.

Использование ЭВМ приводит к коренной перестройке технологии производства практически во всех отраслях промышленности, коммерческой и финансово-кредитной деятельности и, как следствие, — к повышению производительности и улучшению условий труда людей. Именно поэтому современный специалист должен владеть теоретическими знаниями в области информатики и практическими навыками использования вычислительной техники, техники связи и других средств управления.

В условиях перехода от системы жесткого командного распределения ресурсов к рыночным отношениям интенсивно развиваются новые формы организации труда, производственных и межличностных отношений; растет потребность в разнообразной информации, и в частности в оперативных сведениях коммерческого и правового характера. Все эти изменения требуют, чтобы будущие специалисты-профессионалы, являясь пользователями компьютерных информационных систем, были готовы к работе в новых условиях, владели основами информационной технологии, умели оценивать действия информационных систем, качество обработки, точность и полноту информации, закладываемой в основу принимаемых управленческих решений.

Одно из важных направлений переработки информации — ее вычислительная обработка — реализуется современными ЭВМ. Нынешнему состоянию вычислительной техники предшествовал полувековой период, который принято делить на этапы — поколения ЭВМ. Главным показателем, определяющим отнесение конкретной ЭВМ к тому или иному поколению, считается тип ее элементной базы. Смене поколений сопутствовало изменение основных технико-эксплуатационных и экономических характеристик ЭВМ, и в первую очередь таких, как производительность, емкость памяти, надежность, габаритные размеры и стоимость. Важным фактором развития ЭВМ было и остается стремление разработчиков уменьшить трудоемкость подготовки задач для решения их на машинах, облегчить связь человека с ЭВМ и повысить эффективность их использования.

В середине 1980-х гг. началась разработка ЭВМ пятого поколения на базе сверхбольших интегральных схем (СБИС). Модели машин пятого поколения ориентированы на потоковую архитектуру, реализацию интеллектуального человеко-машинного интерфейса, обеспечивающего не только системное решение задач, но и способность машины к логическому мышлению, самообучению, ассоциативной обработке информации и получению логических выводов. Предполагается, что общение человека с ЭВМ будет осуществляться на естественном языке, в том числе и в речевой форме.

Развитие микропроцессорной вычислительной техники, интегральных сетей связи, новых информационных технологий

привело к бурному подъему индустрии переработки информации, появлению новой науки — информатики. Информатика (наука о совокупности процессов получения, передачи, обработки, хранения, представления и распространения информации во всех сферах человеческой деятельности) охватывает как теоретический аспект — методологию информационной деятельности в условиях массовой компьютеризации, так и практический — информационную технологию эффективного применения комплекса технических средств для конкретных приложений.

В будущем возникнут сложнейшие проблемы взаимодействия искусственного интеллекта в виде сверхсложных вычислительных сетей пятого поколения с такими элементами, как суперЭВМ, актуализируемые базы знаний, персональные компьютеры, новейшие локальные и глобальные телекоммуникационные системы, а также проблемы взаимодействия сложившегося естественного социального интеллекта, промышленных роботов массового применения и естественной рабочей силы. Для этого потребуются сверхсложные социальные технологии.

Таким образом, объект информатики, охватывающий все элементы ИТ: технические средства, математическое, алгоритмическое, программное, лингвистическое обеспечение, средства связи во взаимодействии с людьми, — резко усложняется. Особенно сложные инженерные проблемы информатики возникают применительно к телематическим системам в виде безлюдных производств и технологий, в которые интегрированы организационные человеко-компьютерные системы, робототехнические комплексы и новейшие средства связи. Это уже информо-динамические объекты — новые исторические феномены с заложенными в них творческими системами (*Creative Systems*); в процессе их создания и эксплуатации мы попадаем в исключительную зависимость от теории — нового комплекса знаний и нового мышления.

Один и тот же объект может изучаться различными науками. *Объект познания* — это некий фрагмент реального мира, а *предмет познания* — это выбранная для исследования методами данной науки сторона, грань, аспект объекта.

Информатизация общества в части материально-технической базы, математического и программного обеспечения ИТ

изучается различными науками: кибернетикой, системотехникой, теорией информации, а в части формирования функциональных подсистем — различными общественными науками: экономикой, правоведением, психологией. В формировании ИТ участвуют и науки, относящиеся к той или иной автоматизируемой области: медицина (внедрение ЭВМ в систему здравоохранения), педагогика (компьютеризация учебного процесса), военные науки (использование ЭВМ в военном деле), экономика и т. д. Каждая из этих наук рассматривает компьютеризацию со своей стороны, прилагает к ней свои законы и принципы.

А какую же сторону рассматриваемого объекта выбирает информатика, делая ее своим предметом? Она выбирает содержательную, смысловую сторону создания и функционирования информационных систем и технологий, связанную с их сущностью, социальной отдачей, полезностью, местом в общественных системах, историческим значением (т. е. как с фактором радикального прогресса и выхода общества на качественно новые исторические рубежи).

Очень часто в литературе по информатике отмечается только наше отставание от Запада и США, совершенно замалчивается опыт СССР, советских ученых и специалистов в области информатизации. Этот опыт должен быть осмыслен и проанализирован, из него необходимо сделать правильные выводы.

В конце 1960-х гг. (подчеркнем: в те же годы, что и на Западе) в СССР под руководством В.М. Глушкова (Институт кибернетики АН УССР) был разработан проект создания общегосударственной автоматизированной системы (ОГАС). Согласно концепции ОГАС, все ее территориальные и отраслевые звенья должны были создаваться на единой научно-методической основе, суть которой заключается в системном (комплексном) подходе, обеспечивающем целевую, проблемную, информационную и программно-техническую совместимость всех звеньев управления народным хозяйством, т. е. построение системы управления как единого целого путем определения главной деятельности системы и последующего развертывания ее в иерархическую структуру целей методом декомпозиции.

Осуществить такой подход не удалось из-за огромной масштабности и трудоемкости проекта; отсутствия в 1960-х —

начале 1970-х гг. математического аппарата системотехники, технических и программных средств; сложившейся в те годы организации управления.

Одним словом, автоматизировать сразу все на голом месте, игнорируя естественную эволюцию любой системы — ее постепенное наращивание и усложнение, оказалось невозможным. В результате многолетних работ на разных территориях страны появилось множество информационно разобщенных ведомственных систем, функционирующих независимо друг от друга. Хотя, казалось бы, в силу привязки всех отраслевых звеньев ОГАС к той или иной территории при формировании своих информационных баз они должны были использовать всестороннюю информацию о «своей» территории. Однако преобладание отраслевых (ведомственных) интересов над территориальными и ограниченность полномочий местных органов власти в вопросах управления территориями исключали появление таких стимулов.

В концепции информатизации нашей страны в конце 1980-х гг. указанные недостатки были в большей части устранены, и во главу информатизации был поставлен региональный принцип: создание развитой информационной среды будет происходить регионально по мере готовности регионов к решению проблем индустриализации получения и обработки информации (т. е. создания и развития крупного машинного производства в информационной сфере), а также психологических, правовых, экономических и социальных проблем.

Эти проблемы порождены противоречиями между необходимостью использования во всех сферах человеческой деятельности больших объемов информации и невозможностью оперативно формировать такие объемы с помощью традиционных информационных средств, технологий и систем связи.

Страны Запада и Япония уже два десятилетия разрабатывают проблемы информатизации общества. В последние годы там развернуты соответствующие общенациональные программы, поддерживаемые огромными государственными субсидиями. Информатизация считается прорывом в будущее. На это пошли все развитые страны, придав информатизации высшие приоритеты, подчинив этой цели основные ресурсы и усилия. Из объекта теоретического анализа информатизация преврати-

лась в критерий оценки могущества и фактор выживания той или иной страны в борьбе за экономическое и политическое превосходство, стала важнейшим ориентиром для выработки внутренней и внешней стратегии государства.

Стержень всей информатизации — создание новых информационных структур, основой которых являются технические структуры. Развитие технической базы информатизации — приоритетная задача, без решения которой нельзя двигаться вперед. Вычислительные системы пятого поколения будут иметь такие функциональные блоки, как база знаний, машины решения задач и логических выводов, интеллектуальный интерфейс. Особенно заметна разница между ЭВМ пятого поколения и машинами предыдущих поколений на уровне интерфейса — способов взаимодействия с человеком. Они будут приспособлены к человеку, в отличие от машинного интерфейса современных ЭВМ, заставляющего человека приспосабливаться к машине и говорить не на естественном, а на искусственном, машинном языке.

С точки зрения пользователей, системы пятого поколения ЭВМ должны соответствовать таким принципам, как:

— простота применения, когда от пользователя не требуется профессиональных знаний;

— моделирование человеческих функциональных возможностей, таких, как построение доказательств и принятие решений;

— автоматический выбор релевантных запросов данных из машинных хранилищ большого объема;

— гибкость конфигурации, обеспечивающая приспособленность ЭВМ к выполнению широкого диапазона работ;

— автоматизация программирования;

— высокая надежность.

Важнейшим аспектом информатизации является формирование и развитие индустрии информатики — весьма специфической сферы экономики.

Важно понимать крайности в трактовке понятия «информатизация». С одной стороны, нельзя отождествлять ее с компьютеризацией, так как в этом случае все дело сведется к созданию технической базы, насаждению ЭВМ в не подготовленную для них социальную среду, нагромождению бесполезных с хо-

зяйственной точки зрения пирамид из электроники. С другой стороны, нельзя к информатизации сводить все социально-экономические преобразования: слишком широкое ее понимание нивелирует значение других шагов и направлений социальной перестройки.

Информатизацию общества следует понимать как создание и развитие *информационной среды:* комплекса факторов, обеспечивающих наилучшие условия функционирования информационного ресурса с учетом автоматизированных способов его переработки и использования в целях социального прогресса. Можно сказать и иначе: информатизация сводится к формированию ИТ и созданию условий для эффективного их использования в различных общественных системах.

Таким образом, информатизация предполагает более широкий подход к компьютеризации, включающий преобразование всего комплекса средств и условий развертывания информационных процессов: создание соответствующей технической базы, модернизацию организационно-экономических, юридических, человеческого факторов. Иными словами, это целостный процесс формирования новой автоматизированной сферы как необходимого условия эффективного задействования ЭВМ, их сетей, интегрированных АСУ, робототехнических комплексов, банков данных и т. д. Речь идет о программируемом изменении информационной основы функционирования различных общественных систем и подсистем, замене в допустимых пределах бумажной информации человеко-машинными диалоговыми системами, создании новых, несравненно более эффективных моделей деятельности людей.

Развитие прикладной информатики — технологии использования компьютерной техники для реализации конкретных приложений — прошло три этапа: первый — решение задач прямого счета, второй — создание информационной поддержки принятия решений (он предусматривает использование традиционных экономико-математических методов и моделей для решения экономических и других видов задач) и современный, третий этап — поиск методов решения интеллектуальных задач с применением новых информационных и интеллектуальных технологий, созданием экспертных систем, использованием эвристических методов моделирования исследуемых ситуаций,

баз данных и знаний, машинного вывода результатов исследования конкретных ситуаций.

Современная материально-техническая база информатики позволяет широко использовать автоматизированные рабочие места специалистов, занятых во всех сферах и на различных уровнях управленческой деятельности; создавать вычислительные системы, которые в пределах специализированной предметной области способны принимать решения на уровне эксперта-профессионала (экспертные системы), и информационно-коммуникационные сети, формируемые на базе ЭВМ и систем передачи данных.

Экспертные информационные системы, банки данных, базы знаний являются мощным средством накопления интеллекта в конкретных сферах человеческой деятельности, способствующим принятию профессионалом управленческих решений.

С появлением микроЭВМ и ПК возникли локальные вычислительные сети. Они позволили поднять на качественно новую ступень управление объектами, повысить эффективность применения ЭВМ и качество обрабатываемой информации, реализовать безбумажную технологию, создать новые технологии. Благодаря объединению ЛВС и глобальных сетей пользователи получили доступ к мировым информационным ресурсам. Новейшие современные ИТ, такие, как WWW-технология, Internet, интрасети, технологии информационных хранилищ и распределенных объектов и др., содействуют повышению интеллектуальных способностей человека, быстрому росту его творческих возможностей.

Литература

1. *Алексеева И.В., Васяшин А.В., Острейковский В.А.* Информатика / Под ред. В.А. Острейковского. Обнинск: Обнинский институт атомной энергетики, 1996.

2. *Вигдорчик Г.В., Воробьев А.Ю., Праченко В.Д.* Основы программирования на Ассемблере для СМ ЭВМ. М.: Финансы и статистика, 1987.

3. *Вирт Н.* Алгоритмы и структура данных. М.: Мир, 1989.

4. *Власов В.К., Королев Л.Н., Сотников А.Н.* Элементы информатики / Под ред. Л.Н. Королева. М.: Наука, 1988.

5. Вопросы прикладной информатики / Под ред. Р.М. Юсупова. СПб.: СПИИРАН, 1993.

6. *Гудман С., Хидетниеми С.* Введение в разработку и анализ алгоритмов. М.: Мир, 1981.

7. *Древс Ю.Г.* Информационные системы и процессы. М.: МИФИ, 2003.

8. *Зуев Е.А.* Прогнозирование на языке Турбо Паскаль 6.0, 7.0. М.: Веста: Радио и связь, 1993.

9. Информатика: Энциклопедический словарь для начинающих. М.: Педагогика-Пресс, 1994.

10. *Косцов А., Косцов В.* Толковый англо-русский и русско-английский словарь компьютерных терминов. Более 5000 слов и словосочетаний. М.: Мартин, 2006.

11. *Кричевский Р.Е.* Сжатие и поиск информации. М.: Радио и связь, 1989.

12. Лабораторный практикум по информатике / Под ред. проф. В.А. Острейковского. М.: Высшая школа, 2001.

13. *Левин В.И.* Все об информации. М.: Росмэн-Пресс, 2003.

14. *Лихачева Г.Н.* Информационные технологии в экономике. М.: МЭСИ, 1998.

15. *Острейковский В.А.* Автоматизированные информационные технологии в экономике. Сургут: Сургутский гос. ун-т, 2000.

16. *Острейковский В.А.* Информатика. М.: Высшая школа, 2005.

17. *Першиков В.И., Савинков В.М.* Толковый словарь по информатике. М.: Финансы и статистика, 1991.

18. Словарь по кибернетике / Под ред. В.С. Михалевича. 2-е изд. Киев.: Гл. ред. УСЭ им. Бажана, 1989.

19. *Фридманд А.Я.* и др. Информатика и компьютерные технологии. Основные термины: Толковый словарь. М.: Астрель, 2003.

20. Экономическая информатика и вычислительная техника / Под ред. В.П. Косырева, А.Ю. Королева. М.: Финансы и статистика, 1996.

21. Электронные вычислительные машины: В 8 кн. Кн. 2. Основы информатики / Под ред. А.Я. Савельева. М.: Высшая школа, 1991.

Учебное издание

Острейковский Владислав Алексеевич,
Полякова Ирина Викторовна

ИНФОРМАТИКА

ТЕОРИЯ И ПРАКТИКА

Учебное пособие

Заведующая редакцией *Л.В. Дудник*
Редактор *Т.И. Балашова*
Дизайн обложки *А.Л. Чириков*
Технический редактор *Л.А. Данкова*
Корректор *Е.В. Туманова*
Компьютерная верстка *ООО «Бета-Фрейм»*

Подписано в печать 29.08.2008. Формат 60x90 $^{1}/_{16}$
Гарнитура «Таймс». Печать офсетная.
Усл. печ. л. 38,0. Тираж 5000 экз. Заказ №

Общероссийский классификатор продукции
ОК-005-93, том 2; 953 005 — учебная литература

ООО «Издательство Оникс»
105082, Москва, ул. Б. Почтовая, д. 7, стр. 1
Отдел реализации: тел. (499) 619-02-20, 619-31-88
Интернет-магазин: www.onyx.ru