Министерство науки и высшего образования Российской Федерации

Томский государственный университет
систем управления и радиоэлектроники

**А.О. Семкин, А.С. Перин**

# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. ОБЩИЕ ВОПРОСЫ ИНФОРМАТИКИ, АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Учебное пособие для студентов
технических направлений подготовки и специальностей

Томск
Издательство ТУСУРа
2020

УДК 004(075.8)
ББК 32.973я73
С307

**Рецензенты:**

**Касымов Д.П.**, канд. физ.-мат наук, зав. лаб. физ. и вычисл. механики ММФ Нац. исслед. Томск. гос. ун-та

**Плёнкин А.П.**, канд. техн. наук, доц. каф. информ. безопасности телекоммуникационных систем Ин-та компьютерных технологий и информ. безопасности Южного федерального ун-та

Одобрено на заседании кафедры сверхвысокочастотной и квантовой радиотехники, протокол № 2 от 01.10.2020

**Семкин, Артём Олегович**

С307 Информационные технологии. Общие вопросы информатики, алгоритмизации и программирования : учеб. пособие для студентов техн. направлений подготовки и специальностей / А.О. Семкин, А.С. Перин. – Томск : Изд-во Томск. гос. ун-та систем упр. и радиоэлектроники, 2020. – 163 с.

ISBN 978-5-86889-898-3

Приведены основные теоретические материалы по общим вопросам информатики. Рассмотрена аппаратная конфигурация персонального компьютера. Представлены материалы по основам алгоритмизации и программирования. Приведены модели решений функциональных и вычислительных задач. Дано введение в архитектуру вычислительных и операционных систем.

Для студентов всех форм обучения технических направлений подготовки и специальностей.

УДК 004(075.8)
ББК 32.973я73

ISBN 978-5-86889-898-3

# Оглавление

## Введение

Данное учебное пособие является частью учебно-методического комплекса и предназначено для подготовки студентов к лекционным и практическим занятиям по дисциплине «Информационные технологии».

Цель преподавания дисциплины — обеспечить базовую подготовку студентов в области использования средств вычислительной техники, а также развить навыки работы на персональных компьютерах для решения инженерных задач, сбора, передачи, обработки и хранения информации. Лекционный курс по «Информационным технологиям» знакомит студентов с назначением и принципом действия современных персональных компьютеров, основами алгоритмизации и технологиями программирования научно-технических задач, языками программирования высокого уровня, технологиями обработки и отладки программ, современным прикладным программным обеспечением, методами решения типовых инженерных задач и их программной реализацией.

Процесс изучения дисциплины направлен на формирование следующих компетенций:

– способности понимать сущность и значение информации в развитии современного информационного общества, сознавать опасности и угрозы, возникающие в этом процессе, соблюдать основные требования информационной безопасности;

– способности владеть основными методами, способами и средствами получения, хранения, переработки информации;

– способности сформировать навыки самостоятельной работы на компьютере и в компьютерных сетях, осуществлять компьютерное моделирование устройств, систем и процессов с использованием универсальных пакетов прикладных компьютерных программ.

Учебное пособие состоит из пяти разделов. Первый раздел посвящен описанию основных понятий информатики. Во втором разделе рассматриваются вопросы аппаратной конфигурация персонального компьютера. В третьем разделе приведены основы алгоритмизации и программирования. Четвертый раздел посвящен рассмотрению моделей решений функциональных и вычислительных

задач. В пятом разделе приведены общие сведения об операционных системах, базах данных и локальных сетях.

Список литературы включает источники, рекомендуемые для самостоятельного и более углубленного изучения вопросов, выносимых на практические занятия и лабораторные работы.

# 1 ОБЩИЕ ВОПРОСЫ ИНФОРМАТИКИ

## 1.1 Предмет и основные понятия информатики

Информатика — комплексная техническая наука, систематизирующая методы и способы создания, сохранения, воспроизведения, обработки и передачи данных средствами вычислительной техники, а также принципы функционирования этих средств и методы управления ими. Термин «информатика» образован из двух слов: «информация» и «автоматика» и происходит от французского *Informatique.* Этот термин введен во Франции 60-е годы XX века с началом широкого использования вычислительной техники.

Тогда же вошел в употребление термин «*Computer Science*», обозначающий науки о преобразовании информации с помощью вычислительной техники.

*Предмет* информатики как науки:

– аппаратное обеспечение средств вычислительной техники;

– программное обеспечение средств вычислительной техники;

– средства и способы взаимодействия аппаратного и программного обеспечения;

– средства и способы взаимодействия человека с аппаратными и программными средствами.

*Основная задача* информатики — систематизация приемов и методов работы с аппаратными и программными средствами вычислительной техники. Цель систематизации — выделять, внедрять и развивать передовые, более эффективные технологии автоматизации работы с данными, а также методически обеспечивать новые технологические исследования.

Информатика — практическая наука. Ее достижения должны проходить проверку на практике и приниматься в тех случаях, если они отвечают критерию повышения эффективности. В составе основной задачи сегодня можно выделить такие *основные направления* информатики для практического применения:

– архитектура вычислительных систем (приемы и методы построения систем, предназначенных для автоматической обработки данных);

– интерфейсы вычислительных систем (приемы и методы управления аппаратным и программным обеспечением);

– программирование (приемы, методы и средства разработки комплексных задач);

– преобразование данных (приемы и методы преобразования структур данных);

– защита информации (обобщение приемов, разработка методов и средств защиты данных);

– автоматизация (функционирование программно-аппаратных средств без участия человека);

– стандартизация (обеспечение совместимости между аппаратными и программными средствами, между форматами представления данных, относящихся к разным типам вычислительных систем.

## 1.2 Сигналы, данные и методы

*Данные* несут информацию о событиях материального мира и являются регистрацией сигналов, возникших в результате этих событий.

Важно понимать, что данные не тождественны информации. Из наблюдений за окружающим миром исследователь получает поток данных. Однако эти данные станут информацией только при некоторых обстоятельствах.

Объекты материального мира, включая человека, находятся в постоянном взаимодействии друг с другом и с окружающим миром. Такое взаимодействие сопровождается появлением *сигналов*, имеющих материальную природу (электромагнитную, акустическую, механическую и т. д.). Взаимодействие сигналов и физических тел порождает изменение свойств этих тел. Эти изменения можно наблюдать, измерять, фиксировать, контролировать, то есть регистрировать сигналы.

*Данные* — это зарегистрированные сигналы.

Регистрация сигналов подразумевает их отражение на каком-либо материальном носителе. Бумага является самым распространенным в мире носителем зарегистрированных сигналов — данных. Данные несут информацию о событиях, но не тождественны самой информации.

## 1.3 Понятие об информации

В литературе часто встречается определение *информации*, основанное на «знаниях» либо на объективности фактов и свидетельств. Однако средства вычислительной техники обрабатывают информацию в автоматическом режиме, без участия человека, и «знать» ничего не могут. Кроме этого, средства вычислительной техники также не требуют от информации объективного отражения в природе и обществе, т. е. могут работать с искусственной, абстрактной и даже ложной информацией. Таким образом, целесообразно ввести определение *информации*, основанное на факте взаимодействия данных и адекватных им *методов* обработки.

Из введенного определения следует, что *данные* только тогда станут *информативными*, когда будут обработаны *адекватными им методами*.

На практике это означает, что процесс получения информации начинается с регистрации сигнала, например записи слов или рисования схем на доске в аудитории — в этом случае путем локального изменения коэффициента отражения материала доски слова или схемы выделяются на фоне и таким образом формируются данные. Информацией они станут только в том случае, если аудитория будет способна *обработать* эти данные, т. е. как минимум прочитать их и интерпретировать. При этом аудитория должна не только владеть языком, на котором записаны слова, и уметь читать схемы, но и установить между ними логические связи, т. к. одного знания записи слов недостаточно, чтобы уловить смысл записанных комбинаций слов. Следовательно, *методы* должны быть адекватными *данным*.

## 1.4 Свойства информации

**Объективность и субъективность информации.** Понятие объективности информации является относительным. Это понятно, если учесть, что методы являются субъективными. Таким образом, информация может быть *более* или *менее объективной*. Более объективной принято считать информацию, менее подверженную изменениям, вносимым методами обработки данных. Так, например, фото

природного объекта можно считать источником более объективной информации в отличие от рисунка того же объекта, выполненного человеком. В ходе информационного процесса степень объективности информации всегда понижается — чем больше методов используется для обработки данных, тем менее объективна информация.

**Полнота информации.** Полнота информации определяется достаточностью данных для принятия решений или для создания новых данных на основе имеющихся. Это свойство определяет *качество информации*. Чем полнее данные, тем шире диапазон методов, которые можно использовать, тем проще подобрать метод, вносящий минимум погрешностей в ход информационного процесса.

**Достоверность информации.** В процессе регистрации сигналов, только некоторые из них являются необходимыми для формирования данных. Посторонние сигналы, шум могут быть как природного (естественного), так и искусственного происхождения, включая случаи преднамеренного искажения сигналов для нарушения информационного процесса или вмешательства в него. Если полезный сигнал зарегистрирован более четко, чем посторонние сигналы, достоверность информации может быть более высокой. При увеличении уровня шумов достоверность информации снижается. В этом случае для передачи того же количества информации требуется использовать либо больше данных, либо более сложные методы.

**Адекватность информации** — это степень ее соответствия реальной объективной ситуации. Неадекватная информация может образовываться при создании новой информации на основе неполных или недостоверных данных (примеры такой информации можно встретить в средствах массовой информации, которые не всегда проверяют достоверность или полноту данных из своих источников). Однако и полные, и достоверные данные могут приводить к созданию неадекватной информации в случае применения к ним неадекватных методов (пример неадекватной информации такого рода можно найти в истории — древние считали, что Солнце вращается вокруг Земли, ввиду использования неадекватных методов обработки и интерпретации данных, полученных при наблюдениях).

**Доступность информации** — мера возможности получить информацию. На степень доступности информации влияют одновре-

менно как доступность данных, так и доступность адекватных методов для их интерпретации. Отсутствие доступа к данным или отсутствие адекватных методов обработки данных приводят к одинаковому результату: информация оказывается недоступной. Отсутствие адекватных методов для работы с данными во многих случаях приводит к применению неадекватных методов, в результате чего образуется неполная, неадекватная или недостоверная информация (см. выше пример о вращении Солнца и Земли).

**Актуальность информации** — это степень соответствия информации текущему моменту времени. Нередко с актуальностью, как и с полнотой, связывают коммерческую ценность информации. Поскольку информационные процессы растянуты во времени, то достоверная и адекватная, но устаревшая информация может являться источником ошибочных решений. Необходимость поиска (или разработки) адекватного метода для работы с данными может приводить к такой задержке в получении информации, что она становится неактуальной и ненужной.

## 1.5 Носители данных

Самым распространенным носителем данных является бумага. На бумаге данные регистрируются путем изменения оптических характеристик ее поверхности. Изменение оптических свойств (изменение коэффициента отражения поверхности в определенном диапазоне длин волн) используется также в устройствах, осуществляющих запись лазерным лучом на пластмассовых носителях с отражающим покрытием (*CD, DVD, BluRay* и др.).

В современном мире повсеместно внедряются устройства с программируемым изменением оптических свойств — дисплеи. В ближайшее десятилетие ожидается массовое использование носителей и систем отображения данных, изменяющих или дополняющих оптические свойства окружающей человека реальности. Информация при этом будет размещаться либо на окулярах носимых очков, либо прямо на сетчатке глаза.

Носители данных в общем смысле интересны как объекты, определяющие свойства информации. Любой носитель можно

характеризовать параметром *разрешающей способности* (количеством данных, записанных в принятой для носителя единице измерения — как плотно можно разместить данные на данном носителе) и *динамическим диапазоном* (отношением амплитуд максимального и минимального регистрируемых сигналов). От этих свойств носителя нередко зависят такие свойства информации, как полнота, доступность и достоверность.

Задача преобразования данных с целью смены носителя относится к одной из важнейших задач информатики. В структуре стоимости вычислительных систем устройства для ввода и вывода данных, работающие с носителями информации, составляют до половины стоимости аппаратных средств.

## 1.6 Операции с данными

В ходе информационного процесса данные преобразуются из одного вида в другой с помощью методов. Обработка данных включает в себя множество различных операций. Выделим некоторые основные:

– сбор данных — накопление для обеспечения достаточной полноты;

– формализация (подготовка) данных — приведение данных к установленной форме, чтобы сделать их сопоставимыми между собой, то есть повысить их уровень доступности;

– фильтрация данных — исключение из набора данных, в которых нет необходимости, с целью увеличения достоверности и адекватности информации;

– сортировка данных — упорядочение данных по заданному признаку с целью повышения доступности информации (в настоящее время получили развитие системы «больших данных» (*BigData*), работающие с неформализованными и неупорядоченными данными);

– архивация данных — организация хранения данных для повышения общей надежности информационного процесса в целом;

– защита данных — комплекс мер, направленных на предотвращение утраты, воспроизведения и модификации данных;

– транспортировка данных — прием и передача данных между удаленными участниками информационного процесса;

– преобразование данных — перевод данных из одной формы в другую или из одной структуры в другую, что часто связано с изменением типа носителя.

## 1.7 Кодирование данных двоичным кодом

При работе с данными различных типов важную роль играет автоматизация их обработки. Для этого очень важно унифицировать форму их представления. Одним из методов такой унификации является система кодирования, то есть представление данных одного типа в виде данных другого типа (рисунок 1.1).


Рисунок 1.1 — Примеры различных систем кодирования

Самой распространенной системой кодирования в вычислительной технике является двоичная. Данная система основана на представлении данных последовательностью всего двух знаков: 0 и 1.

Эти знаки называются двоичными цифрами, по английский — *binary digit* или, сокращенно, *bit* (бит).

Один бит содержит информацию только об одном событии и выражает только одно из двух понятий: 0 или 1 (*да* или *нет*, *черное*

или *белое*, *истина* или *ложь* и т. п.). Два бита выражают одно из четырех понятий:

00 01 10 11

Три бита кодируют восемь различных понятий:

000 001 010 011 100 101 110 111

Увеличение количества разрядов на единицу в системе двоичного кодирования приводит к двукратному увеличению количества значений, которое может быть выражено. Таким образом, зависимость количества значений, которые могут быть выражены двоичным *словом*, от количества разрядов двоичного *слова* имеет вид:

$$N = 2^m, \qquad (1.1)$$

где $N$ — количество независимых кодируемых значений; $m$ — разрядность двоичного кодирования, принятая в данной системе.

Целые числа кодируются двоичным кодом путем деления его пополам до тех пор, пока в остатке не образуется ноль или единица. Совокупность остатков от каждого деления, записанная справа налево вместе с последним остатком, и образует двоичный аналог десятичного числа.

Для кодирования целых чисел от 0 до 255 достаточно иметь 8 разрядов двоичного кода (8 бит). Шестнадцать бит позволяют закодировать целые числа от 0 до 65535, а 24 бита — уже более 16,5 миллионов разных значений.

Кодирование дробных чисел представляет собой несколько более сложную операцию, но концептуально подход сохраняется. Сначала дробное число записывается в виде десятичной дроби. Затем десятичная дробь переписывается в *экспоненциальном* виде:

*m*,xxxxxx*Eyyy*,

где *m* — целая часть числа, обычно содержит один или два десятичных разряда; xxxxxx — дробная часть числа, может содержать до 20 десятичных разрядов и больше; *E* — символ, означающий умножение дробного числа на 10 в степени *yyy*. Такая запись позволяет применить правила кодирования целых чисел к элементам дробного, отдельно кодируя и сохраняя целую, дробную части и степень числа 10.

При этом как малые дробные, так и большие целые числа могут быть записаны и сохранены в таком виде. Например, числа

0,0000003 и 3000000 фактически отличаются в $10^{12}$ раз. Однако при экспоненциальной записи они будут отличаться только степенью числа 10: 3,0E–6 и 3,0E6. Таким образом оба числа могут быть представлены в виде двоичного кода с одинаковым числом разрядов.

Такой подход к записи чисел нередко называют «переносом запятой», а тип данных, который соответствует получающимся числам — *числами с «плавающей» запятой*.

## 1.8 Кодирование текстовых данных

Кодирование текстовой информации обычно выполняется посимвольно. Существует ряд систем кодирования, в которых каждому символу алфавита (букве, знаку препинания и т.п.) ставится в соответствие целое число. Далее это число кодируется описанным выше способом. При сохранении таких данных отмечается, что обрабатывать их следует именно как текст.

Как было показано выше, восемь двоичных разрядов (8 бит) позволяют закодировать 256 различных значений. Этого количества достаточно для представления всех символов английского и русского языка (прописных и строчных), а также всех знаков препинания, символов основных арифметических действий и некоторых общепринятых специальных символов.

Наиболее часто применяемой в вычислительной технике в силу исторически сложившейся традиции является система кодирования *ASCII* (*American Standard Code for Information Interchange* — стандартный код информационного обмена США), введенная Институтом стандартизации США (*ANSI — American National Standard Institute*). *ASCII* содержит две таблицы кодирования — базовую и расширенную. Базовая — закрепляет значения кодов от 0 до 127, а расширенная — относится к символам с номерами от 128 до 255.

Первые 32 кода базовой таблицы (0—31) используются производителями аппаратных средств. Это так называемые управляющие коды, они управляют обработкой текстовых и других данных, им не соответствуют никакие символы языков, они не выводятся ни на экран, ни на устройства печати.

Коды 32–127 представляют собой числа, которые кодируют символы английского алфавита, знаки препинания, цифры,

арифметические действия и некоторые вспомогательные символы. Базовая таблица кодировки *ASCII* приведена на рисунке 1.2.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | пробел | 48 | 0 | 64 | @ | 80 | P | 96 | ` | 112 | p |
| 33 | ! | 49 | 1 | 65 | A | 81 | Q | 97 | a | 113 | q |
| 34 | " | 50 | 2 | 66 | B | 82 | R | 98 | b | 114 | r |
| 35 | # | 51 | 3 | 67 | C | 83 | S | 99 | c | 115 | s |
| 36 | $ | 52 | 4 | 68 | D | 84 | T | 100 | d | 116 | t |
| 37 | % | 53 | 5 | 69 | E | 85 | U | 101 | e | 117 | u |
| 38 | & | 54 | 6 | 70 | F | 86 | V | 102 | f | 118 | v |
| 39 | ' | 55 | 7 | 71 | G | 87 | W | 103 | g | 119 | w |
| 40 | ( | 56 | 8 | 72 | H | 88 | X | 104 | h | 120 | x |
| 41 | ) | 57 | 9 | 73 | I | 89 | Y | 105 | i | 121 | y |
| 42 | * | 58 | : | 74 | J | 90 | Z | 106 | j | 121 | z |
| 43 | + | 59 | ; | 75 | K | 91 | [ | 107 | k | 123 | { |
| 44 | , | 60 | < | 76 | L | 92 | \ | 108 | l | 124 | \| |
| 45 | - | 61 | = | 77 | M | 93 | ] | 109 | m | 125 | } |
| 46 | . | 62 | > | 78 | N | 94 | ^ | 110 | n | 126 | ~ |
| 47 | / | 63 | ? | 79 | O | 95 | _ | 111 | o | 127 | |

Рисунок 1.2 — Базовая таблица кодировки *ASCII*

Другая распространенная кодировка носит название КОИ-8 (код обмена информацией, восьмизначный) (рисунок 1.3). Она имеет широкое распространение в компьютерных сетях на территории России и в российском секторе Интернета. Ее происхождение относится ко временам действия Совета Экономической Взаимопомощи государств Восточной Европы.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 128 | | 144 | ▒ | 160 | ▬ | 176 | ╠ | 192 | ю | 208 | п | 224 | Ю | 240 | П |
| 129 | \| | 145 | ▒ | 161 | Ё | 177 | ╣ | 193 | а | 209 | я | 225 | А | 241 | Я |
| 130 | ┌ | 146 | ▓ | 162 | ╒ | 178 | ╡ | 194 | б | 210 | р | 226 | Б | 242 | Р |
| 131 | ┐ | 147 | ⌠ | 163 | ё | 179 | Ё | 195 | ц | 211 | с | 227 | Ц | 243 | С |
| 132 | └ | 148 | ■ | 164 | ╓ | 180 | ╢ | 196 | д | 212 | т | 228 | Д | 244 | Т |
| 133 | ┘ | 149 | • | 165 | ╔ | 181 | ╖ | 197 | е | 213 | у | 229 | Е | 245 | У |
| 134 | ├ | 150 | √ | 166 | ╕ | 182 | ╤ | 198 | ф | 214 | ж | 230 | Ф | 246 | Ж |
| 135 | ┤ | 151 | ≈ | 167 | ╗ | 183 | ╥ | 199 | г | 215 | в | 231 | Г | 247 | В |
| 136 | ┬ | 152 | ≤ | 168 | ╘ | 184 | ╦ | 200 | х | 216 | ь | 232 | Х | 248 | Ь |
| 137 | ┴ | 153 | ≥ | 169 | ╙ | 185 | ╧ | 201 | и | 217 | ы | 233 | И | 249 | Ы |
| 138 | ┼ | 154 | | 170 | ╚ | 186 | ╨ | 202 | й | 218 | з | 234 | Й | 250 | З |
| 139 | ▀ | 155 | ⌡ | 171 | ╛ | 187 | ╩ | 203 | к | 219 | ш | 235 | К | 251 | Ш |
| 140 | ▄ | 156 | ° | 172 | ┘ | 188 | ╪ | 204 | л | 220 | э | 236 | Л | 252 | Э |
| 141 | ▌ | 157 | ² | 173 | ┌ | 189 | ╫ | 205 | м | 221 | щ | 237 | М | 253 | Щ |
| 142 | ▐ | 158 | · | 174 | ╜ | 190 | ╬ | 206 | н | 222 | ч | 238 | Н | 254 | Ч |
| 143 | ▐ | 159 | ÷ | 175 | ╞ | 191 | ё | 207 | о | 223 | ъ | 239 | О | 255 | Ъ |

Рисунок 1.3 — Кодировка КОИ-8

В настоящее время существует и используется большое количество различных систем кодирования текстовой информации. В связи с этим аппаратные и программные средства не всегда могут поддерживать ту или иную систему кодирования. Это выражается в некорректном отображении (замене на нечитаемые символы) текстовой информации в файлах и передаваемых пакетах данных. Задача межсистемного преобразования данных — это одна из распространенных задач информатики.

Описанную проблему призвана решить система, основанная на 16-разрядном кодировании символов. Она называется универсальной — *UNICODE*. Шестнадцать разрядов позволяют обеспечить уникальные коды для 65536 различных символов — этого поля достаточно для размещения в одной таблице символов большинства языков планеты.

## 1.9 Кодирование графических данных

Для кодирования цветных графических изображений применяется принцип декомпозиции произвольного цвета на основные составляющие. Наиболее распространенными являются две системы кодирования, основанные на декомпозиции: *RGB* для источников излучения (дисплеи, мониторы, интерактивные табло и т. д.) и *CMYK* для печатных носителей информации (газеты, буклеты и т. д.). В системе *RGB* используются три основных цвета: красный (*Red*, *R*), зеленый (*Green, G*) и синий (*Blue, B*). В системе *CMYK* используются также три основных цвета: голубой (*Cyan*, *C*), пурпурный (*Magenta, M*) и желтый (Yellow, *Y*), и один дополнительный цвет — черный (*black, K*). На практике считается (хотя теоретически это не совсем так), что любой цвет, видимый человеческим глазом, можно получить путем механического смешения перечисленных цветов для соответствующих носителей. Принцип образования цвета заключается в задании интенсивности каждого из основных цветов (их еще называют «каналами») для единицы изображения. Таким образом, каждая единица изображения на «излучающем» носителе (пиксель на экране дисплея) кодируется в виде координаты, а также комбинации значений интенсивности каждого канала *RGB*. Для печатных изданий

принцип аналогичен, только интенсивность канала задается размером печатной точки каждого из цветов системы *CMYK*.

Если для кодирования интенсивности (яркости) каждого из каналов *RGB* использовать по 256 значений (восемь двоичных разрядов, восемь бит), как это принято для полутоновых черно-белых изображений, то на кодирование цвета одной точки надо затратить 24 разряда. При этом система кодирования обеспечивает однозначное определение 16,5 млн различных цветов, что на самом деле близко к чувствительности человеческого глаза. Режим представления цветной графики с использованием 24 двоичных разрядов называется полноцветным (*True Color*).

## 1.10 Кодирование звуковой информации

Для кодирования звуковой информации разработано множество отдельных корпоративных стандартов, но если говорить обобщенно, то можно выделить два основных направления.

Метод *FM* (*Frequency Modulation*) основан на теории функциональных преобразований и разложений сложных функций в ряды. Согласно одной из теорий — преобразования Фурье — любое сложное колебание (в частности, звук) можно представить в виде суммы простейших гармонических сигналов (гармоник), каждый из которых представляет собой правильную синусоиду. Каждая синусоида характеризуется как минимум двумя числами — частотой и амплитудой и, следовательно, может быть представлена в виде как минимум двух чисел, т. е. кодов. В природе звуковые сигналы имеют непрерывный характер, то есть являются аналоговыми. Их разложение в гармонические ряды и представление в виде дискретных цифровых сигналов, т. е. дискретизацию, выполняют специальные устройства — аналогово-цифровые преобразователи (АЦП). Обратное преобразование для воспроизведения звука, закодированного числовым кодом, выполняют цифро-аналоговые преобразователи (ЦАП). При таких преобразованиях неизбежны потери информации, связанные с методом кодирования, поэтому качество звукозаписи сильно зависит от применяемого метода и параметров процесса дискретизации. Однако данный метод кодирования обеспечивает весьма компактный

код, и потому он нашел широкое применение. Основной метод борьбы с низким качеством цифровых записей звука — увеличение частоты дискретизации, но увеличение частоты дискретизации неизбежно ведет к увеличению объема сохраняемых данных. Поэтому высококачественные цифровые записи требуют носителей большого объема.

Второе направление развития систем кодирования звуковой информации — метод таблично-волнового (*Wave-Table*) синтеза. Согласно принципам работы данных систем звукозапись представляется не в виде элементарных колебаний (синусоид), а в виде совокупности заранее сохраненных образцов звучания (сэмплов, от англ. *sample* — образец) различных инструментов и других источников. Числовые коды выражают тип инструмента, номер его модели, высоту тона, продолжительность и интенсивность звука, динамику его изменения, некоторые параметры среды, в которой происходит звучание, а также прочие параметры, характеризующие особенности звука. Поскольку в качестве образцов используются «реальные» звуки, то качество звука, полученного в результате синтеза, получается очень высоким и приближается к качеству звучания реальных музыкальных инструментов. Обратной стороной такого подхода является большой объем хранимых данных и сложность программной обработки такого кода.

## 1.11 Кодирование видеоинформации

Несложно догадаться, что кодирование видеоинформации представляет собой процесс последовательного сохранения графической информации и синхронизированных с ней звуковых данных.

Для сокращения объема обрабатываемых данных существуют отдельные технологии представления видеоинформации, среди которых основной является сохранение только изменяющейся части кадра. Например, при съемке движения поезда перрон остается неподвижным, поэтому целесообразно полностью хранить только первый кадр, а в последующих кадрах перезаписывать только те части, которые изменяются при перемещении поезда.

Процессы кодирования-декодирования выполняются специальными средствами — кодеками (сокращенно от «кодер-декодер»). Кодеки могут быть как аппаратными, так и программными.

## 1.12 Основные структуры данных

Работа с большими наборами данных автоматизируется проще, когда данные упорядочены, то есть образуют заданную структуру. Существуют три основных типа структур данных: линейная, иерархическая и табличная.

### Линейные структуры (списки данных, векторы данных)

Наиболее простым примером линейной структуры данных является список. Характерной особенностью линейной структуры данных является однозначное определение элемента набора данных своим уникальным номером в массиве. Нумеруя отдельные страницы книги, мы создаем структуру списка. Студенческий журнал посещаемости — это тоже список, в котором студенты зарегистрированы под своими уникальными номерами (зачетка, студенческий билет — все это имеет свой уникальный номер).

Итак, линейные структуры данных (списки) — это упорядоченные структуры, в которых адрес элемента однозначно определяется его номером.

### Табличные структуры (таблицы данных, матрицы данных)

Таблицы данных тоже не требуют подробного описания. Достаточно вспомнить всем известную таблицу умножения. В табличных структурах элемент характеризуется адресом, состоящим из нескольких параметров. Например, для двумерных таблиц адрес состоит из номера строки и номера столбца. Нужная ячейка находится на их пересечении, а элемент выбирается из ячейки.

При хранении табличных данных количество разделителей должно быть больше, чем для данных, имеющих структуру списка. Например, когда таблицы печатают в книгах, строки и столбцы раз-

деляют графическими элементами — линиями вертикальной и горизонтальной разметки (рисунок 1.4).

Равная длина элементов таблицы упрощает ее представление. Такие таблицы называют матрицами. В случае матриц, поскольку все элементы имеют равную длину и количество их известно, необходимости в разделителях нет. В матрице, имеющей $M$ строк и $N$ столбцов, поиск элемента с адресом ($m$, $n$) сводится сначала к ее просмотру и отсчету $a\left[N\left(m-1\right)+\left(n+1\right)\right]$ символов, где $a$ — длина одного элемента. Со следующего символа начнется нужный элемент. Его длина тоже равна $a$, поэтому его конец определить нетрудно.

| Планета | Расстояние до Солнца, а.е. | Относительная масса | Количество спутников |
|---|---|---|---|
| Меркурий | 0,39 | 0,056 | 0 |
| Венера | 0.67 | 0,88 | 0 |
| Земля | 1,0 | 1,0 | 1 |
| Марс | 1,51 | 0,1 | 2 |
| Юпитер | 5,2 | 318 | 16 |

Рисунок 1.4 — Пример двумерной таблицы

Таким образом, табличные структуры данных (матрицы) — это упорядоченные структуры, в которых адрес элемента определяется номером строки и номером столбца, на пересечении которых находится ячейка, содержащая искомый элемент.

### Иерархические структуры данных

Данные, которые невозможно представить в виде списка или таблицы, часто выстраивают в иерархические структуры. Для данных структур характерно последовательное уточнение признаков элементов при движении сверху вниз. Подобные структуры широко применяют в научных систематизациях и всевозможных классификациях. Достаточно вспомнить классификацию животных и растений.

### Дихотомия данных

Основным недостатком иерархических структур данных является большой размер пути доступа. Зачастую длина маршрута превышает длину самих данных. В связи с этим разработаны методы упрощения иерархических структур с целью сокращения пути доступа. Один из методов получил название дихотомии. Его суть понятна из примера, представленного на рисунка 1.5.

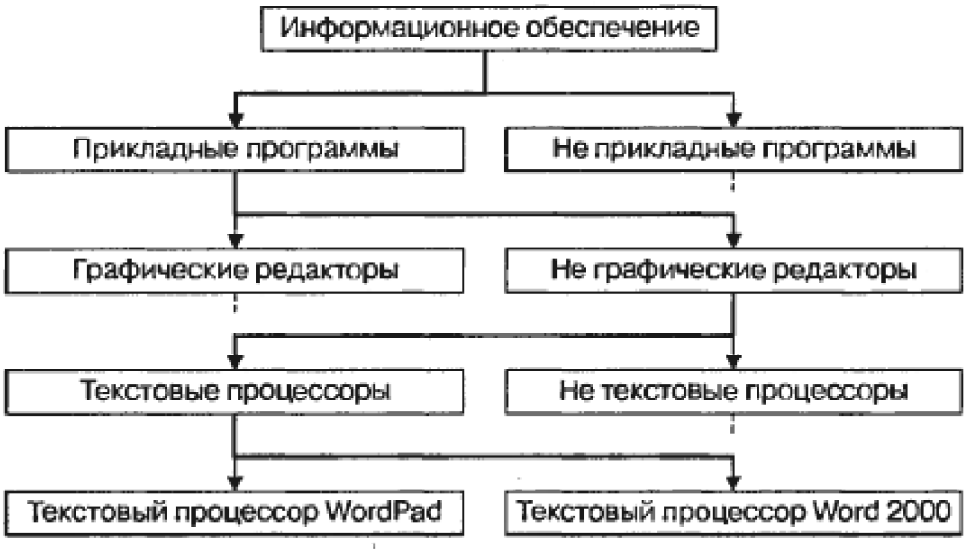



Рисунок 1.5 — Принцип действия метода дихотомии

### Неструктурированные и слабо структурированные данные

В современном мире глубина внедрения цифровых информационных систем обусловливает огромный объем обрабатываемых данных. С одной стороны, это приводит к необходимости значительных вложений в строительство программно-аппаратной инфраструктуры (центры обработки данных, хранилища и т. п.). С другой стороны, повсеместная цифровизация открывает дополнительные возможности обработки больших объемов данных с целью выявления глобальных закономерностей и их глубокого анализа.

При этом современные системы статистической обработки и глубокого анализа больших объемов данных (их называют «большими данными», или «BigData») не требуют структурирования информации. За счет алгоритмов корреляционной обработки (и других)

они (системы) могут работать и с несвязанными на первый взгляд данными. Вкупе с алгоритмами машинного обучения такие системы формируют современный тренд развития информационных систем.

## 1.13 Единицы измерения данных

На сегодняшний день известно множество различных систем и единиц измерения данных. В информатике для измерения данных используют тот факт, что разные типы данных имеют универсальное двоичное представление и потому вводят свои единицы данных, основанные на нем.

Наименьшей единицей измерения является байт. Поскольку одним байтом, как правило, кодируется один символ текстовой информации, то для текстовых документов (без учета алгоритмов сжатия) размер в байтах соответствует лексическому объему в символах (пока исключение представляет рассмотренная выше универсальная кодировка *UNICODE*).

Более крупная единица измерения — килобайт (Кбайт). 1 Кбайт равен $2^{10}$ байт (1024 байт).

Более крупные единицы измерения данных образуются добавлением префиксов мега, гига, тера, в более крупных единицах пока нет практической надобности

1 Мбайт = 1024 Кбайт = $2^{20}$ байт

1 Гбайт = 1024 Мбайт = $2^{30}$ байт

1 Тбайт = 1024 Гбайт = $2^{40}$ байт.

## 1.14 Единицы хранения данных

Задача хранения данных включает в себя решение двух проблем: обеспечения компактности при хранении и обеспечения простого и быстрого доступа к данным.

Адрес хранения данных тоже является данными, а значит, также имеет размер и также подлежит хранению. Отсюда следует, что хранение данных в виде мелких единиц, таких как байты, неудобно. Их неудобно хранить и в более крупных единицах (килобайтах, мега-

байтах и т. п.), поскольку неполное заполнение одной единицы хранения приводит к неэффективности хранения.

Поэтому в качестве единицы хранения данных принят объект переменной длины, называемый файлом. Файл — это последовательность произвольного числа байтов, обладающая уникальным собственным именем. Обычно в отдельном файле хранят данные, относящиеся к одному типу. В этом случае тип данных определяет тип файла.

Особое внимание уделяется имени файла. Фактически, имя файла — это его адрес. Без имени данные, хранящиеся в файле, не станут информацией из-за отсутствия метода доступа к ним. Кроме этого, имя файла обычно содержит сведения о типе данных, заключенных в нем. Для автоматических средств работы с данными это важно, поскольку по имени файла они могут автоматически определить адекватный метод извлечения информации из файла.

## 1.15 Понятие о файловой структуре

Уникальность имени файла гарантирует однозначность доступа к данным. В вычислительной технике уникальность имени обеспечивается автоматически — запрещено создание файла с именем, которое совпадает с уже имеющимся файлом.

Чаще всего хранение файлов организуется иерархически. Соответствующая структура называется файловой. На вершине структуры размещается имя носителя. Далее файлы объединяются в директории (папки) с поддержкой вложенных директорий-каталогов (папок). Адрес (имя) файла включает последовательное указание имени устройства и имен всех директорий. В качестве разделителя обычно используется символ «\» (обратная косая черта).

Уникальность имени файла обеспечивается тем, что полным именем файла считается собственное имя файла вместе с путем доступа к нему. Понятно, что в этом случае на одном носителе не может быть двух файлов с тождественными полными именами.

# 2 АППАРАТНАЯ КОНФИГУРАЦИЯ ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Современный персональный компьютер (ПК) — это универсальная техническая система, построенная таким образом, что его конфигурацию (состав) можно гибко изменять по мере необходимости. Тем не менее существует базовая конфигурация, считающаяся типовой и минимально необходимой (рисунок 2.1). В таком комплекте компьютер обычно поставляется. В настоящее время в базовой конфигурации рассматривают четыре устройства:

– системный блок;
– монитор;
– клавиатура;
– мышь.


Рисунок 2.1 — Базовая конфигурация персонального компьютера

## 2.1 Системный блок

Системный блок ПК — это ключевой его элемент, содержащий наиболее важные функциональные элементы компьютерной системы. Все устройства и компоненты системы, размещенные внутри системного блока, называют внутренними, а подключаемые к нему снаружи — соответственно, внешними или периферийными.

В отрыве от наполняющих устройств существенными параметрами системного блока являются параметры корпуса и электропитания. Геометрические характеристики корпуса и возможности

размещения в нем дополнительных устройств определяются параметром, который называется *форм-фактором*.

Форм-фактор корпуса согласуется с форм-фактором материнской платы компьютера, характеристика которой будет дана ниже.

Корпус ПК обычно поставляется в комплекте с блоком питания и набором кабельных изделий (шин и кабельных сборок) для питания внтренних устройств системного блока. Таким образом, мощность блока питания также является одним из параметров системного блока. В современные ПК рекомендуется устанавливать блоки питания мощностью не менее 500 Вт. Игровые и профессиональные компьютерные системы могут требовать и кратно большей мощности — 1,5 кВт и выше.

## 2.2 Монитор

Монитор — устройство визуального представления данных. Это не единственно возможное, но главное устройство вывода. Его основными потребительскими параметрами являются: тип, размер и шаг маски экрана, максимальная частота регенерации изображения, класс защиты.

Сейчас наиболее распространены мониторы двух основных типов: на основе электронно-лучевой трубки (ЭЛТ) и плоские жидкокристаллические (ЖК). ЭЛТ-мониторы обеспечивают лучшее качество изображения, но в пользу жидкокристаллических мониторов говорит их компактность, небольшой вес, идеально плоская поверхность экрана.

Размер монитора измеряется между противоположными углами видимой части экрана по диагонали. Единица измерения — дюймы. Стандартные размеры: 14"; 15"; 17"; 19"; 20"; 21". В настоящее время наиболее универсальными являются мониторы размером 15 (ЖК) и 17 дюймов (ЭЛТ), а для операций с графикой желательны мониторы размером 19–21 дюйм (ЭЛТ). Изображение на экране ЭЛТ-монитора получается в результате облучения люминофорного покрытия остронаправленным пучком электронов, разогнанных в вакуумной колбе.

Частота регенерации (обновления) изображения показывает, сколько раз в течение секунды монитор может полностью сменить изображение (поэтому ее также называют частотой кадров). Этот параметр зависит не только от монитора, но и от свойств и настроек видеоадаптера, хотя предельные возможности определяет все-таки монитор.

Частоту регенерации изображения измеряют в герцах (Гц). Чем частота выше, тем четче и устойчивее изображение, тем меньше утомление глаз, тем больше времени можно работать с компьютером непрерывно. При частоте регенерации порядка 60 Гц мелкое мерцание изображения может быть заметно невооруженным глазом. Сегодня такое значение считается недопустимым.

## 2.3 Клавиатура

Клавиатура — клавишное устройство управления персональным компьютером. Служит для ввода алфавитно-цифровых (знаковых) данных, а также команд управления. Комбинация монитора и клавиатуры обеспечивает простейший интерфейс пользователя. С помощью клавиатуры управляют компьютерной системой, а с помощью монитора получают от нее отклик.

Клавиатура относится к стандартным средствам персонального компьютера. Ее основные функции не нуждаются в поддержке специальными системными программами (драйверами). Необходимое программное обеспечение для начала работы с компьютером уже имеется в микросхеме постоянного запоминающего устройства в составе базовой системы ввода-вывода *(BIOS),* и потому компьютер реагирует на нажатия клавиш сразу после включения.

## 2.4 Мышь

Мышь — устройство управления манипуляторного типа. Представляет собой плоскую коробочку с двумя-тремя кнопками. Перемещение мыши по плоской поверхности синхронизировано с перемещением графического объекта (указателя мыши) на экране монитора.

## 2.5 Внутренние устройства системного блока

### Материнская плата

Материнская плата — основная плата персонального компьютера. На ней размещаются:

– процессор — основная микросхема, выполняющая большинство математических и логических операций;

– микропроцессорный комплект (чипсет) — набор микросхем, управляющих работой внутренних устройств компьютера и определяющих основные функциональные возможности материнской платы;

– шины — наборы проводников, по которым происходит обмен сигналами между внутренними устройствами компьютера;

– оперативная память (оперативное запоминающее устройство (ОЗУ)) — набор микросхем, предназначенных для временного хранения данных, когда компьютер включен;

– постоянное запоминающее устройство (ПЗУ) — микросхема, предназначенная для длительного хранения данных, в том числе и когда компьютер выключен;

– разъемы для подключения дополнительных устройств (слоты).

### Жесткий диск

Жесткий диск — основное устройство для долговременного хранения больших объемов данных и программ (рисунок 2.2). На самом деле это не один диск, а группа соосных дисков, имеющих магнитное покрытие и вращающихся с высокой скоростью. Таким образом, этот «диск» имеет не две поверхности, как должно быть у обычного плоского диска, а $2n$ поверхностей, где $n$ — число отдельных дисков в группе.

Рисунок 2.2 — Жесткий диск

## Видеокарта (видеоадаптер)

Совместно с монитором видеокарта образует видеоподсистему персонального компьютера. Видеокарта не всегда была компонентом ПК. На заре развития персональной вычислительной техники в общей области оперативной памяти существовала небольшая выделенная экранная область памяти, в которую процессор заносил данные об изображении. Специальный контроллер экрана считывал данные о яркости отдельных точек экрана из ячеек памяти этой области и в соответствии с ними управлял разверткой горизонтального луча электронной пушки монитора.

С переходом от черно-белых мониторов к цветным и с увеличением разрешения экрана (количества точек по вертикали и горизонтали) области видеопамяти стало недостаточно для хранения графических данных, а процессор перестал справляться с построением и обновлением изображения. Тогда и произошло выделение всех операций, связанных с управлением экраном, в отдельный блок, получивший название видеоадаптер. Физически видеоадаптер выполнен в виде отдельной дочерней платы, которая вставляется в один из слотов материнской платы и называется видеокартой. Видеоадаптер взял на себя функции видеоконтроллера, видеопроцессора и видеопамяти.

За время существования персональных компьютеров сменилось несколько стандартов видеоадаптеров: *MDA* (монохромный); *CGA* (4 цвета); *EGA* (16 цветов); *VGA* (256 цветов). В настоящее время применяются видеоадаптеры *SVGA*, обеспечивающие по выбору воспроизведение до 16,7 миллионов цветов с возможностью произвольного выбора разрешения экрана из стандартного ряда значений (640x480, 800x600, 1024x768, 1152x864; 1280x1024 точек и далее).

Разрешение экрана является одним из важнейших параметров видеоподсистемы. Чем оно выше, тем больше информации можно отобразить на экране, но тем меньше размер каждой отдельной точки и, соответственно, тем меньше видимый размер элементов изображения.

Видеоускорение — одно из свойств видеоадаптера, которое заключается в том, что часть операций по построению изображений может происходить без выполнения математических вычислений

в основном процессоре компьютера, а чисто аппаратным путем, преобразованием данных в микросхемах видеоускорителя.

### Звуковая карта

Звуковая карта явилась одним из наиболее поздних усовершенствований персонального компьютера. Она устанавливается в один из разъемов материнской платы в виде дочерней карты и выполняет вычислительные операции, связанные с обработкой звука, речи, музыки. Звук воспроизводится через внешние звуковые колонки, подключаемые к выходу звуковой карты. Специальный разъем позволяет отправить звуковой сигнал на внешний усилитель.

Основным параметром звуковой карты является разрядность, определяющая количество битов, используемых при преобразовании сигналов из аналоговой в цифровую форму и наоборот. Чем выше разрядность, тем меньше погрешность, связанная с оцифровкой, тем выше качество звучания. Минимальным требованием сегодняшнего дня являются 16 разрядов, а наибольшее распространение имеют 32-разрядные и 64-разрядные устройства.

## 2.6 Системы, расположенные на материнской плате

### Оперативная память

Оперативная память *(RAM — Random Access Memory)* — это массив кристаллических ячеек, способных хранить данные. Существует много различных типов оперативной памяти, но с точки зрения физического принципа действия различают динамическую память *(DRAM*) и статическую память *(SRAM)*.

Ячейки динамической памяти *(DRAM)* можно представить в виде микроконденсаторов, способных накапливать заряд на своих обкладках. Это наиболее распространенный и экономически доступный тип памяти.

Ячейки статической памяти *(SRAM)* можно представить как электронные микроэлементы — триггеры, состоящие из нескольких транзисторов. В триггере хранится не заряд, а состояние (включен/выключен), поэтому этот тип памяти обеспечивает более высо-

кое быстродействие, хотя технологически он сложнее и, соответственно, дороже.

Микросхемы динамической памяти используют в качестве основной оперативной памяти компьютера, а микросхемы статической памяти — в качестве вспомогательной памяти (так называемой кэш-памяти), предназначенной для оптимизации работы процессора.

Каждая ячейка памяти имеет свой адрес, который выражается числом. В большинстве современных процессоров предельный размер адреса обычно составляет 32 разряда, а это означает, что всего независимых адресов может быть $2^{32}$. Одна адресуемая ячейка содержит восемь двоичных ячеек, в которых можно сохранить 8 бит, то есть один байт данных.

Таким образом, в современных компьютерах возможна непосредственная адресация к полю памяти размером $2^{32}$ байт = 4 Гбайт. Минимальный объем памяти определяется требованиями операционной системы и для современных компьютеров составляет 128 Мбайт.

## Процессор

Процессор — основная микросхема компьютера, в которой и производятся все вычисления. Конструктивно процессор состоит из ячеек, похожих на ячейки оперативной памяти, но в этих ячейках данные могут не только храниться, но и изменяться. Внутренние ячейки процессора называют регистрами. Важно также отметить, что данные, попавшие в некоторые регистры, рассматриваются не как данные, а как команды, управляющие обработкой данных в других регистрах.

С остальными устройствами компьютера и в первую очередь с оперативной памятью процессор связан несколькими группами проводников, называемых шинами. Основных шин три: шина данных, адресная шина и командная шина.

**Адресная шина**. У процессоров семейства *Pentium* (а именно они наиболее распространены в персональных компьютерах) адресная шина 32-разрядная, то есть состоит из 32 параллельных проводников. В зависимости от того, есть напряжение на какой-то из линий или нет, говорят, что на этой линии выставлена единица или ноль.

Комбинация из 32 нулей и единиц образует 32-разрядный адрес, указывающий на одну из ячеек оперативной памяти. К ней и подключается процессор для копирования данных из ячейки в один из своих регистров.

**Шина данных**. По этой шине происходит копирование данных из оперативной памяти в регистры процессора и обратно. В современных персональных компьютерах шина данных, как правило, 64-разрядная, то есть состоит из 64 линий, по которым за один раз на обработку поступают сразу 8 байтов.

**Шина команд**. Для того чтобы процессор мог обрабатывать данные, ему нужны команды. Он должен знать, что следует сделать с теми байтами, которые хранятся в его регистрах. Эти команды поступают в процессор тоже из оперативной памяти, но не из тех областей, где хранятся массивы данных, а оттуда, где хранятся программы. Команды тоже представлены в виде байтов. Самые простые команды укладываются в один байт, однако есть и такие, для которых нужно два, три и более байтов.

В большинстве современных процессоров шина команд 32-разрядная, хотя существуют 64-разрядные процессоры и даже 128-разрядные.

Основными параметрами процессоров являются: рабочее напряжение, разрядность, рабочая тактовая частота, коэффициент внутреннего умножения тактовой частоты и размер кэш-памяти.

Рабочее напряжение процессора обеспечивает материнская плата, поэтому разным маркам процессоров соответствуют разные материнские платы (их надо выбирать совместно). По мере развития процессорной техники происходит постепенное понижение рабочего напряжения. Ранние модели процессоров *x*86 имели рабочее напряжение 5 В. С переходом к процессорам *Intel Pentium* оно было понижено до 3,3 В, а в настоящее время оно составляет менее 2 В.

Разрядность процессора показывает, сколько бит данных он может принять и обработать в своих регистрах за один раз (за один такт).

### Микросхема ПЗУ и система *BIOS*

В момент включения компьютера в его оперативной памяти нет ничего, ни данных, ни программ, поскольку оперативная память не может ничего хранить без подзарядки ячеек более сотых долей секунды, но процессору нужны команды, в том числе и в первый момент после включения. Поэтому сразу после включения на адресной шине процессора выставляется стартовый адрес. Это происходит аппаратно, без участия программ (всегда одинаково). Процессор обращается по выставленному адресу за своей первой командой и далее начинает работать по программам.

Этот исходный адрес не может указывать на оперативную память, в которой пока ничего нет. Он указывает на другой тип памяти — постоянное запоминающее устройство. Микросхема ПЗУ способна длительное время хранить информацию, даже когда компьютер выключен. Программы, находящиеся в ПЗУ, называют «зашитыми», их записывают туда на этапе изготовления микросхемы.

Комплект программ, находящихся в ПЗУ, образует базовую систему ввода-вывода *(BIOS — Basic Input Output System)*. Основное назначение программ этого пакета состоит в том, чтобы проверить состав и работоспособность компьютерной системы и обеспечить взаимодействие с клавиатурой, монитором, жестким диском и дисководом гибких дисков. Программы, входящие в *BIOS*, позволяют нам наблюдать на экране диагностические сообщения, сопровождающие запуск компьютера, а также вмешиваться в ход запуска с помощью клавиатуры.

### Энергонезависимая память *CMOS*

Для того чтобы начать работу с другим оборудованием, программы, входящие в состав *BIOS*, должны знать, где можно найти нужные параметры. По очевидным причинам их нельзя хранить ни в оперативной памяти, ни в постоянном запоминающем устройстве.

Специально для этого на материнской плате есть микросхема «энергонезависимой памяти», по технологии изготовления называемая *CMOS*. От оперативной памяти она отличается тем, что ее содержимое не стирается во время выключения компьютера, а от ПЗУ она отличается тем, что данные в нее можно заносить и изменять

самостоятельно, в соответствии с тем, какое оборудование входит в состав системы.

В микросхеме *CMOS* хранятся данные о гибких и жестких дисках, о процессоре, о некоторых других устройствах материнской платы. Тот факт, что компьютер четко отслеживает время и календарь (даже и в выключенном состоянии), тоже связан с тем, что показания системных часов постоянно хранятся (и изменяются) в *CMOS*. Таким образом, программы, записанные в *BIOS*, считывают данные о составе оборудования компьютера из микросхемы *CMOS*, после чего они могут выполнить обращение к жесткому диску, а в случае необходимости и к гибкому, и передать управление тем программам, которые там записаны.

### Шинные интерфейсы материнской платы

Связь между всеми собственными и подключаемыми устройствами материнской платы выполняют ее шины и логические устройства, размещенные в микросхемах микропроцессорного комплекта (чипсета). От архитектуры этих элементов во многом зависит производительность компьютера.

*ISA*. Историческим достижением компьютеров платформы *IBM PC* стало внедрение почти двадцать лет назад архитектуры, получившей статус промышленного стандарта *ISA (Industry Standard Architecture)*. Она не только позволила связать все устройства системного блока между собой, но и обеспечила простое подключение новых устройств через стандартные разъемы (слоты). Пропускная способность шины, выполненной по такой архитектуре, составляет до 5,5 Мбайт/с, но, несмотря на низкую пропускную способность, эта шина еще может использоваться в некоторых компьютерах для подключения сравнительно «медленных» внешних устройств, например звуковых карт и модемов.

*EISA*. Расширением стандарта *ISA* стал стандарт *EISA* (*Extended ISA*), отличающийся увеличенным разъемом и увеличенной производительностью (до 32 Мбайт/с). Как и *ISA* в настоящее время данный стандарт считается устаревшим. После 2000 года выпуск материнских плат с разъемами *ISA/EISA* и устройств, подключаемых к ним, практически прекращен.

*VLB*. Название интерфейса переводится как локальная шина стандарта *VESA* (*VESA Local Bus)*. Понятие «локальной шины» впервые появилось в конце 80-х годов XX века. Оно связано тем, что при внедрении процессоров третьего и четвертого поколений (*Intel* 80386 и *Intel* 80486*)* частоты основной шины (в качестве основной использовалась шина *ISA/EISA*) стало недостаточно для обмена между процессором и оперативной памятью. Локальная шина, имеющая повышенную частоту, связала между собой процессор и память в обход основной шины. Впоследствии в эту шину «врезали» интерфейс для подключения видеоадаптера, который тоже требует повышенной пропускной способности, так появился стандарт *VLB*, который позволил поднять тактовую частоту локальной шины до 50 МГц и обеспечил пиковую пропускную способность до 130 Мбайт/с.

*PCI*. Интерфейс *PCI* (*Peripheral Component Interconnect* — стандарт подключения внешних компонентов) был введен в персональных компьютерах во времена процессора 80486 и первых версий *Pentium*. По своей сути это тоже интерфейс локальной шины, которая связывает процессор с оперативной памятью, в которую врезаны разъемы для подключения внешних устройств. Для связи с основной шиной компьютера (*ISA/EISA*) используются специальные интерфейсные преобразователи — мосты *PCI (PCI Bridge)*. В современных компьютерах функции моста *PCI* выполняют микросхемы микропроцессорного комплекта (чипсета).

*FSB*. Шина *PCI*, появившаяся в компьютерах на базе процессоров *Intel Pentium* как локальная шина, предназначенная для связи процессора с оперативной памятью, недолго оставалась в этом качестве. Сегодня она используется только как шина для подключения внешних устройств, а для связи процессора и памяти, начиная с процессора *Intel Pentium Pro*, используется специальная шина, получившая название *Front Side Bus* (*FSB*). Эта шина работает на частоте 100 – 200 МГц. Частота шины *FSB* является одним из основных потребительских параметров, именно он и указывается в спецификации материнской платы.

*AGP*. Видеоадаптер — устройство, требующее особенно высокой скорости передачи данных. Как при внедрении локальной шины *VLB*, так и при внедрении локальной шины *PCI* видеоадаптер всегда

был первым устройством, «врезаемым» в новую шину. Когда параметры шины *PCI* перестали соответствовать требованиям видеоадаптеров, для них была разработана отдельная шина, получившая название *AGP (Advanced Graphic Port* — усовершенствованный графический порт*)*. Частота этой шины соответствует частоте шины *PCI* (33 МГц или 66 МГц), но она имеет много более высокую пропускную способность за счет передачи нескольких сигналов за один такт.

*PCMCIA (Personal Computer Memory Card International Association* — стандарт международной ассоциации производителей плат памяти для персональных компьютеров). Этот стандарт определяет интерфейс подключения плоских карт памяти небольших размеров и используется в портативных персональных компьютерах.

*USB* (*Universal Serial Bus* — универсальная последовательная магистраль). Это одно из последних нововведений в архитектурах материнских плат. Этот стандарт определяет способ взаимодействия компьютера с периферийным оборудованием. Он позволяет подключать до 256 различных устройств, имеющих последовательный интерфейс. Устройства могут включаться цепочками (каждое следующее устройство подключается к предыдущему). Производительность шины *USB* относительно невелика, но вполне достаточна для таких устройств, как клавиатура, мышь, модем джойстик, принтер и т. п. Удобство шины состоит в том, что она практически исключает конфликты между различным оборудованием, позволяет подключать и отключать устройства в «горячем режиме» (не выключая компьютер) и объединять несколько компьютеров в простейшую локальную сеть без применения специального оборудования и программного обеспечения.

### Функции микропроцессорного комплекта (чипсета)

Параметры микропроцессорного комплекта (чипсета) в наибольшей степени определяют свойства и функции материнской платы. В настоящее время большинство чипсетов материнских плат выпускаются на базе двух микросхем, исторически получивших название «северный мост» и «южный мост».

«Северный мост» обычно управляет взаимосвязью процессора, оперативной памяти и порта *AGP*.

«Южный мост» называют также функциональным контроллером. Он выполняет функции контроллера жестких и гибких дисков, функции контроллера шины *PCI,* моста *ISA-PCI*, контроллера клавиатуры, мыши, шины *USB* и т. п.

У предыдущих поколений материнских плат связь между северным и южным мостом обеспечивала шина *PCI*, контроллер которой располагался в северном мосте. У современных материнских плат мосты соединены новой шиной повышенной производительности, а контроллер шины *PCI* находится в южном мосте вместе с контроллерами всех прочих устройств.

## 2.7 Периферийные устройства персонального компьютера

Периферийные устройства персонального компьютера подключаются к его интерфейсам и предназначены для выполнения вспомогательных операций. Благодаря им компьютерная система приобретает гибкость и универсальность. По назначению периферийные устройства можно подразделить:

– устройства ввода данных;
– устройства вывода данных;
– устройства хранения данных;
– устройства обмена данными.

## 2.8 Устройства ввода знаковых данных

**Специальные клавиатуры**. Клавиатура является основным устройством ввода данных. Специальные клавиатуры предназначены для повышения эффективности процесса ввода данных. Это достигается путем изменения формы клавиатуры, раскладки ее клавиш или метода подключения к системному блоку.

Клавиатуры, имеющие специальную форму, рассчитанную с учетом требований эргономики, называют эргономичными клавиатурами. Их целесообразно применять на рабочих местах, предназначенных для ввода большого количества знаковой информации. Эргономичные клавиатуры не только повышают производительность

наборщика и снижают общее утомление в течение рабочего дня, но и снижают вероятность и степень развития ряда заболеваний, например туннельного синдрома кистей рук и остеохондроза верхних отделов позвоночника.

Раскладка клавиш стандартных клавиатур далека от оптимальной. Она сохранилась со времен ранних образцов механических пишущих машин. В настоящее время существует техническая возможность изготовления клавиатур с оптимизированной раскладкой и существуют образцы таких устройств (в частности, к ним относится клавиатура Дворака). Однако практическое внедрение клавиатур с нестандартной раскладкой находится под вопросом в связи с тем, что работе с ними надо учиться специально.

## 2.9 Устройства командного управления

### Специальные манипуляторы

Кроме обычной мыши существуют и другие типы манипуляторов, например: трекболы, пенмаусы, инфракрасные мыши.

**Трекбол** в отличие от мыши устанавливается стационарно, и его шарик приводится в движение ладонью руки. Преимущество трекбола состоит в том, что он не нуждается в гладкой рабочей поверхности, поэтому трекболы нашли широкое применение в портативных персональных компьютерах.

**Тачпады** — сенсорные пластины, реагирующие на движение пальца пользователя по поверхности. Удар пальцем по поверхности тачпада воспринимается как нажатие кнопки. Недостатком тачпадов является невысокая точность.

**Пенмаус** представляет собой аналог шариковой авторучки, на конце которой вместо пишущего узла установлен узел, регистрирующий величину перемещения.

**Инфракрасная мышь** отличается от обычной наличием устройства беспроводной связи с системным блоком.

## 2.10 Устройства ввода графических данных

Для ввода графической информации используют сканеры, графические планшеты (дигитайзеры) и цифровые фотокамеры. Интересно отметить, что с помощью сканеров можно вводить и знаковую информацию. В этом случае исходный материал вводится в графическом виде, после чего обрабатывается специальными программными средствами (программами распознавания образов).

**Планшетные сканеры**. Планшетные сканеры предназначены для ввода графической информации с прозрачного или непрозрачного листового материала. Принцип действия этих устройств состоит в том, что луч света, отраженный от поверхности материала (или прошедший сквозь прозрачный материал), фиксируется специальными элементами, называемыми приборами с зарядовой связью (ПЗС).

**Ручные сканеры**. Принцип действия ручных сканеров в основном соответствует планшетным. Разница заключается в том, что протягивание линейки ПЗС в данном случае выполняется вручную. Равномерность и точность сканирования при этом обеспечиваются неудовлетворительно, и разрешающая способность ручного сканера составляет 150–300 *dpi*.

**Барабанные сканеры**. В сканерах этого типа исходный материал закрепляется на цилиндрической поверхности барабана, вращающегося с высокой скоростью. Устройства этого типа обеспечивают наивысшее разрешение (2400 – 5000 *dpi*) благодаря применению не ПЗС, а фотоэлектронных умножителей. Их используют для сканирования исходных изображений, имеющих высокое качество, но недостаточные линейные размеры (фотонегативов, слайдов и т. п.).

**Сканеры форм**. Предназначены для ввода данных со стандартных форм, заполненных механически или «от руки». Необходимость в этом возникает при проведении переписей населения, обработке результатов выборов и анализе анкетных данных. От сканеров форм не требуется высокой точности сканирования, но быстродействие играет повышенную роль и является основным потребительским параметром.

**Штрих-сканеры**. Эта разновидность ручных сканеров предназначена для ввода данных, закодированных в виде штрих-кода. Такие устройства имеют применение в розничной торговой сети.

**Графические планшеты** (дигитайзеры) предназначены для ввода художественной графической информации. Существует несколько различных принципов действия графических планшетов, но в основе всех лежит фиксация перемещения специального пера относительно планшета. Устройства удобны для художников и иллюстраторов, поскольку позволяют создавать экранные изображения привычными приемами, наработанными для традиционных инструментов (карандаш, перо, кисть).

**Цифровые фотокамеры**. Как и сканеры, эти устройства воспринимают графические данные с помощью приборов с зарядовой связью, объединенных в прямоугольную матрицу. Основным параметром цифровых фотоаппаратов является разрешающая способность, которая напрямую связана с количеством ячеек ПЗС в матрице.

## 2.11 Устройства вывода данных

В качестве устройств вывода данных, дополнительных к монитору, используют печатающие устройства (принтеры), позволяющие получать копии документов на бумаге или прозрачном носителе. По принципу действия различают матричные, лазерные, светодиодные и струйные принтеры.

**Матричные принтеры**. Это простейшие печатающие устройства. Данные выводятся на бумагу в виде оттиска, образующегося при ударе цилиндрических стержней «иголок» через красящую ленту. Качество печати матричных принтеров напрямую зависит от количества иголок в печатающей головке.

**Лазерные принтеры** обеспечивают высокое качество печати, не уступающее, а во многих случаях и превосходящее полиграфическое. Они отличаются также высокой скоростью печати, которая измеряется в страницах в минуту (*ppm-page per minute*). Как и в матричных принтерах, итоговое изображение формируется из отдельных точек.

**Светодиодные принтеры**. Принцип действия светодиодных принтеров похож на принцип действия лазерных принтеров. Разница заключается в том, что источником света является не лазерная головка, а линейка светодиодов. Поскольку эта линейка расположена по всей ширине печатаемой страницы, отпадает необходимость в механизме формирования горизонтальной развертки и вся конструкция получается проще, надежнее и дешевле. Типичная величина разрешения печати для светодиодных принтеров составляет порядка 600 *dpi*.

**Струйные принтеры**. В струйных печатающих устройствах изображение на бумаге формируется из пятен, образующихся при попадании капель красителя на бумагу. Выброс микрокапель красителя происходит под давлением, которое развивается в печатающей головке за счет парообразования. В некоторых моделях капля выбрасывается щелчком в результате пьезоэлектрического эффекта — этот метод позволяет обеспечить более стабильную форму капли, близкую к сферической. Качество печати изображения во многом зависит от формы капли и ее размера, а также от характера впитывания жидкого красителя поверхностью бумаги. В этих условиях особую роль играют вязкостные свойства красителя и свойства бумаги.

## 2.12 Устройства хранения данных

В настоящее время для внешнего хранения данных применяют несколько типов устройств, использующих магнитные или магнитооптические носители.

**Стримеры** — это накопители на магнитной ленте. Их отличает сравнительно низкая цена. К недостаткам стримеров относят малую производительность (она связана прежде всего с тем, что магнитная лента — это устройство последовательного доступа) и недостаточную надежность (кроме электромагнитных наводок, ленты стримеров испытывают повышенные механические нагрузки и могут физически выходить из строя). Емкость магнитных кассет (картриджей) для стримеров достигает нескольких десятков гигабайт.

**Накопители на съемных магнитных дисках**. К этой категории относится несколько разных типов устройств, ни одно из

которых так и не стало общепринятым стандартом. Например, *ZIP*-накопители выпускаются компанией *Iomega*, специализирующейся на создании внешних устройств для хранения данных. Устройство работает с дисковыми носителями, которые по размеру незначительно превышают стандартные гибкие диски и имеют емкость 100/250/750 Мбайт.

**Магнитооптические устройства**. Эти устройства получили широкое распространение в компьютерных системах высокого уровня благодаря своей универсальности. С их помощью решаются задачи резервного копирования, обмена данными и их накопления. Однако достаточно высокая стоимость приводов и носителей не позволяет отнести их к устройствам массового спроса. В этом секторе параллельно развиваются 5,25- и 3,5-дюймовые накопители, носители для которых отличаются в основном форм-фактором и емкостью.

**Флэш-диски.** Это современное устройство хранения данных на основе энергонезависимой флэш-памяти. Устройство имеет минимальные размеры и допускает «горячее» подключение в разъем *USB*, после чего распознается как жесткий диск, причем не требует установки драйвера. Объем флэш-дисков может составлять от 32 Мбайт до 1 Гбайт, их распространение сдерживает относительно высокая цена.

## 2.13 Устройства обмена данными

**Модем**. Устройство, предназначенное для обмена информацией между удаленными компьютерами по каналам связи, принято называть модемом. При этом под каналом связи понимают физические линии (проводные, оптоволоконные, кабельные, радиочастотные), способ их использования (коммутируемые и выделенные) и способ передачи данных (цифровые или аналоговые сигналы). В зависимости от типа канала связи устройства приема-передачи подразделяют на радиомодемы, кабельные модемы и прочие. Наиболее широкое применение нашли модемы, ориентированные на подключение к коммутируемым телефонным каналам связи.

Цифровые данные, поступающие в модем из компьютера, преобразуются в нем путем модуляции (по амплитуде, частоте, фазе) в соответствии с избранным стандартом (протоколом) и направляются в телефонную линию. Модем-приемник, понимающий данный протокол, осуществляет обратное преобразование (демодуляцию) и пересылает восстановленные цифровые данные в свой компьютер. Таким образом обеспечивается удаленная связь между компьютерами и обмен данными между ними.

# 3 АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Компьютерные программы создают программисты — люди, обученные процессу их составления (программированию). Знаем, что программа — это логически упорядоченная последовательность команд, необходимых для управления компьютером (выполнения им конкретных операций), поэтому программирование сводится к созданию последовательности команд, которая необходима для решения определенной задачи.

Алгоритм — это точно определенное описание способа решения задачи в виде конечной (по времени) последовательности действий. Такое описание еще называется формальным. Для представления алгоритма в виде, понятном компьютеру, служат языки программирования. Сначала всегда разрабатывается алгоритм действий, а потом он записывается на одном из таких языков. В итоге получается текст программы — это полное, законченное и детальное описание алгоритма на языке программирования. Затем этот текст программы специальными служебными приложениями, которые называются трансляторами, либо переводится в машинный код, либо исполняется.

## 3.1 Уровни языков программирования

Разные типы процессоров имеют разные наборы команд. Если язык программирования ориентирован на конкретный тип процессора и учитывает его особенности, то он называется языком программирования низкого уровня. Языком самого низкого уровня является язык ассемблера, который просто представляет каждую команду машинного кода, но не в виде чисел, а с помощью символьных условных обозначений, называемых мнемониками. Однозначное преобразование одной машинной инструкции в одну команду ассемблера называется транслитерацией. Языки программирования высокого уровня значительно ближе и понятнее человеку, нежели компьютеру. Особенности конкретных компьютерных архитектур в них не учитываются, поэтому создаваемые программы на уровне исходных текстов легко переносимы на другие платформы, для которых создан

транслятор этого языка. Разрабатывать программы на языках высокого уровня с помощью понятных и мощных команд значительно проще, а ошибок при создании программ допускается гораздо меньше.

## 3.2 Языки программирования высокого уровня

***FORTRAN*** **(Фортран)**. Это первый компилируемый язык, созданный Джимом Бэкусом в 50-е годы XX века. Программисты, которые разрабатывали программы исключительно на ассемблере, выражали серьезное сомнение в возможности появления высокопроизводительного языка высокого уровня, поэтому основным критерием при разработке компиляторов Фортрана являлась эффективность исполняемого кода. Хотя в Фортране впервые был реализован ряд важнейших понятий программирования, удобство создания программ было принесено в жертву возможности получения эффективного машинного кода. Однако для этого языка было создано огромное количество библиотек, начиная от статистических комплексов и кончая пакетами управления спутниками, поэтому Фортран продолжает активно использоваться во многих организациях. Имеется стандартная версия Фортрана *HPF* (*High Performance Fortran*) для параллельных суперкомпьютеров со множеством процессоров.

***COBOL*** **(Кобол)**. Это компилируемый язык для применения в экономической области и решения бизнес-задач, разработанный в начале 60-х годов XX века. Он отличается большой «многословностью», его операторы иногда выглядят как обычные английские фразы. В Коболе были реализованы очень мощные средства работы с большими объемами данных, хранящимися на различных внешних носителях. На этом языке создано очень много приложений, которые активно эксплуатируются и сегодня. Достаточно сказать, что наибольшую зарплату в США получают программисты на Коболе.

***Algol*** **(Алгол)**. Компилируемый язык, созданный в 1960 году. Он был призван заменить Фортран, но из-за более сложной структуры не получил широкого распространения. В 1968 году была создана версия Алгол 68, по своим возможностям и сегодня опережающая многие языки программирования, однако из-за отсутствия

достаточно эффективных компьютеров для нее не удалось своевременно создать хорошие компиляторы.

***Pascal* (Паскаль)**. Язык Паскаль, созданный в конце 70-х годов прошлого века основоположником множества идей современного программирования Никлаусом Виртом, во многом напоминает Алгол, но в нем ужесточен ряд требований к структуре программы и имеются возможности, позволяющие успешно применять его при создании крупных проектов.

***Basic* (Бейсик)**. Для этого языка есть и компиляторы, и интерпретаторы, а по популярности он занимает первое место в мире. Он создавался в 60-х годах XX века в качестве учебного языка и очень прост в изучении.

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

***C++* (Си++)**. Си++ — это объектно-ориентированное расширение языка Си, созданное Бьярном Страуструпом в 1980 году. Множество новых мощных возможностей, позволивших резко повысить производительность программистов, наложилось на унаследованную от языка Си определенную низкоуровневость, в результате чего создание сложных и надежных программ потребовало от разработчиков высокого уровня профессиональной подготовки.

***Java* (Джава, Ява)**. Этот язык был создан компанией *Sun* в начале 90-х годов прошлого века на основе Си++. Он призван упростить разработку приложений на основе Си++ путем исключения из него всех низкоуровневых возможностей. Но главная особенность этого языка — компиляция не в машинный код, а в платформно-независимый байт-код (каждая команда занимает один байт). Этот байт-код может выполняться с помощью интерпретатора — виртуальной *java*-машины *JVM* (*Java Virtual Machine*), версии которой

созданы сегодня для любых платформ. Благодаря наличию множества java-машин программы Java можно переносить не только на уровне исходных текстов, но и на уровне двоичного байт-кода, поэтому по популярности язык Ява сегодня занимает второе место в мире после Бейсика. Особое внимание в развитии этого языка уделяется двум направлениям: поддержке всевозможных мобильных устройств и микрокомпьютеров, встраиваемых в бытовую технику (технология *jini*), и созданию платформно-независимых программных модулей, способных работать на серверах в глобальных и локальных сетях с различными операционными системами (технология *Java Beans*). Пока основной недостаток этого языка — невысокое быстродействие, так как язык Ява интерпретируемый.

***JavaScript* (ДжаваСкрипт)**. Один из самых популярных языков программирования в мире. Он, созданный более 20 лет назад, прошёл в своём развитии огромный путь. JavaScript (JS) задумывался как скриптовый язык для браузеров. В самом начале он обладал куда более скромными возможностями, чем сейчас. Его в основном использовали для создания несложных анимаций, вроде выпадающих меню, о нем знали как о части технологии Dynamic HTML (DHTML, динамический HTML). Со временем потребности веб-среды росли, в частности появлялись новые API, и JavaScript для поддержки веб-разработки нужно было не отставать от других технологий. В наши дни JS используется не только в традиционных браузерах, но и за их пределами. В частности, речь идёт о серверной платформе Node.js, о возможностях по использованию JavaScript в разработке встраиваемых и мобильных приложений, о решении широкого спектра задач, для решения которых раньше JavaScript не использовался.

***C#* (Си Шарп)**. В конце 90-х годов XX века в компании *Microsoft* под руководством Андерса Хейльсберга был разработан язык *C#*. В нем воплотились лучшие идеи Си и Си++, а также достоинства *Java*. Правда, *C#*, как и другие технологии *Microsoft*, ориентирован на платформу *Windows*. Однако формально он не отличается от прочих универсальных языков, а корпорация даже планирует его стандартизацию. Язык C# предназначен для быстрой разработки *NET*-приложений, и его реализация в системе *Microsoft Visual Studio .NET* содержит множество особенностей, привязывающих *C#* к внутренней архитектуре *Windows* и платформы *.NET*.

***Python* (Пайтон).** Высокоуровневый язык программирования, который используется в различных сферах IT, таких как машинное обучение, разработка приложений, web, парсинг и другие. Язык начал разрабатывать программист Гвидо ван Россум в конце 1980-х. На тот момент он работал в центре математики и информатике в Нидерландах. Россум работал над Python в свободное время, в качестве основы он взял язык программирования ABC, в разработке которого когда-то участвовал. В 2019 году Python стал самым популярным языком программирования, обогнав Java на 10 %. Это обусловлено многими причинами, одна из которых — высокая оплата труда квалифицированных специалистов (около 100 тысяч долларов в год).

## 3.3 Этапы решения задач на ЭВМ

Первоначально ЭВМ были созданы для вычислений, но постепенно на ней стали решать задачи по физике, химии, биологии, управлению технологическими процессами, рисованию мультфильмов и т. д., т. е. для решения задач с математикой непосредственно не связанных. В общем случае выделяют несколько этапов в подготовке и решении задач на ЭВМ.

**На первом этапе** анализируется условие задачи, определяются исходные данные и результаты, устанавливается зависимость между величинами, рассматриваемыми в задаче. Некоторые задачи имеют множество способов решения, поэтому необходимо выбрать способ решения (осуществить постановку задачи, составить модель задачи). Для этого необходимо определить математические соотношения между исходными данными и результатом. Выполнив перевод задачи на язык математики, получают математическую модель.

**Второй этап** заключается в составлении алгоритма решения задачи по выбранной модели.

На **третьем этапе** алгоритм записывается на языке программирования и полученная программа вводится в ЭВМ. Далее проводится отладка программы, т. е. поиск ошибок. Различают логические и семантические ошибки. Семантические ошибки возникают, когда программист неправильно записывает конструкции языка программирования. Семантические ошибки отыскать легче, т. к. современ-

ные трансляторы языков программирования способны их выявить. Логические ошибки возникают, когда инструкции записаны правильно, но последовательность их выполнения дает неверный результат.

Далее проводится тестирование, которое заключается в запуске программы с использованием контрольных примеров — тестов. Тесты выбирают таким образом, чтобы при работе с ними программа прошла все возможные ветви алгоритма, поскольку на каждом из них могут быть свои ошибки.

После отладки и тестирования программа выполняется с реальными исходными данными и проводится анализ полученных результатов, т. е. сопоставление их с экспериментальными фактами, теоретическими воззрениями и другой информацией об изучаемом объекте. Если результаты работы программы не удовлетворяют пользователей по каким-либо параметрам, то производится уточнение модели. При уточнении модели правится алгоритм программы, снова проводятся отладка, тестирование, расчеты и анализ результатов. Так продолжается до тех пор, пока результаты работы программы не будут удовлетворять знаниям об изучаемом объекте.

Общая схема решения задач с помощью ЭВМ приведена на рисунке 3.1.

## 3.4 Погрешности при вычислениях на ЭВМ

При решении задачи на ЭВМ практически невозможно получить точное решение. Получаемое численное решение почти всегда содержит погрешность, т. е. является приближенным. Погрешности решения задач на ЭВМ объясняются следующими причинами:

– математическая модель задачи является приближенным описанием реального объекта или процесса. Поэтому получаемые результаты также всегда будут приближенными, а их погрешности зависят от степени адекватности моделей реальному объекту или процессу;

– исходные данные при решении вычислительной задачи, как правило, содержат погрешности. Это объясняется тем, что исходные данные получают в результате экспериментов, наблюдений, измерений или в результате решения вспомогательных задач;

– применяемые для решения вычислительных задач методы в большинстве случаев являются приближенными, так как получить аналитическое решение задачи обычно не удается;

– использование ЭВМ вносит ошибки, которые появляются при вводе-выводе данных в процессе вычислений.


Рисунок 3.1 — Схема решения задач с помощью ЭВМ

С учетом указанных выше причин погрешность решения вычислительной задачи на ЭВМ складывается из трех составляющих:

– неустранимой погрешности;

– погрешности метода;

– вычислительной погрешности.

Неустранимая погрешность соответствует первым двум причинам, и единственный способ уменьшить эту погрешность заключается в переходе к более точной модели или в использовании более точных входных данных.

Погрешность метода определяется третьей причиной, причем появление этой погрешности практически неизбежно при любых вычислениях.

Вычислительная погрешность возникает в основном из-за округления чисел при вводе-выводе, а также при выполнении арифметических операций в ЭВМ. Это обусловлено ограниченной разрядностью ЭВМ и особенностями представления данных в памяти машины.

## 3.5 Основные типы алгоритмов

Алгоритмизация выступает как набор определенных практических приёмов, особых специфических навыков рационального мышления в рамках заданных языковых средств. Алгоритмизация вычислений предполагает решение задачи в виде последовательности действий, т. е. решение, представленное в виде блок-схемы. Можно выделить типичные алгоритмы. К ним относятся: линейные алгоритмы, разветвляющиеся алгоритмы, циклические алгоритмы.

### Линейные алгоритмы

Линейный алгоритм является наиболее простым. В нём предполагается последовательное выполнение операций. В этом алгоритме не предусмотрены проверки условий или повторений.

**Пример.** Вычислить функцию $z = \dfrac{(x - y)}{x + y^2}$.

Составить блок-схему вычисления функции по линейному алгоритму. Значения переменных $x$, $y$ могут быть любые, кроме нуля, вводить их с клавиатуры.

*Решение.* Линейный алгоритм вычисления функции задан в виде блок-схемы (рисунок 3.2). При выполнении линейного алгоритма значения переменных вводятся с клавиатуры, подставляются в заданную функцию, вычисляется, а затем выводится результат.

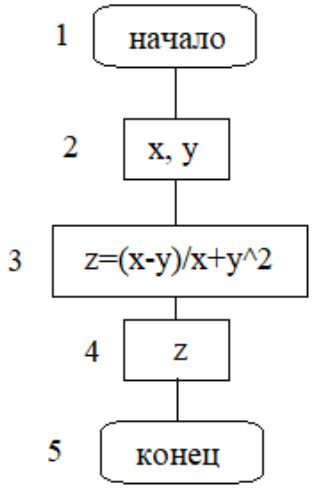



Рисунок 3.2 — Линейный алгоритм:
блок 1 — логическое начала; блок 2 — ввод данных;
блок 3 — арифметическое действие; блок 4 — выводит результат;
блок 5 — логическое завершение

## Алгоритмы ветвлений

Разветвляющийся алгоритм предполагает проверку условий для выбора решения, соответственно, в алгоритме появятся две ветви для каждого условия.

В примере рассматривается разветвляющийся алгоритм, где в зависимости от условия выбирается один из возможных вариантов решений. Алгоритм представляется в виде блок-схемы.

**Пример**. При выполнении условия $x > 0$ вычисляется функция: $z = \ln x + y$, иначе, а именно, когда $x = 0$ или $x < 0$, вычисляется функция: $z = x + y^2$.

Составить блок-схему вычисления функции по алгоритму ветвления. Значения переменных *x, y* могут быть любые, вводить их с клавиатуры.

*Решение.* На рисунке 3.3 представлен разветвляющийся алгоритм, где в зависимости от условия выполнится одна из веток. В блок-схеме появился новый блок 3, который проверяет условие задачи. Остальные блоки знакомы из линейного алгоритма.

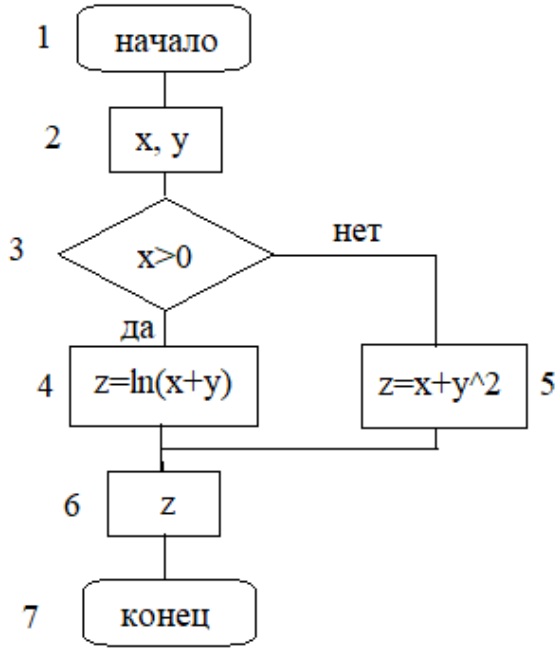



Рисунок 3.3 — Алгоритм ветвления

## Циклические алгоритмы

Циклический алгоритм предусматривает повторение одной операции или нескольких операций в зависимости от условия задачи.

Из циклических алгоритмов выделяют два типа:

– с заданным количеством циклов или со счётчиком циклов;

– количество циклов неизвестно.

### Циклические и рекуррентные вычисления

Многие циклические вычислительные процессы используют рекуррентные зависимости при решении различных математических задач.

В общем виде формулу для рекуррентных вычислений можно представить так:

$$Y_i = F\left(Y_{i-1}, Y_{i-2}, ..., Y_{i-k}\right). \qquad (3.1)$$

В этой рекуррентной формуле для вычисления $i$-го члена последовательности $Y_i$, где $i > k$, используются $k$ предыдущих членов последовательности $Y_{i-1}, Y_{i-2}, ..., Y_{i-k}$. Для вычислений по этой формуле нужно задать $k$ первых членов последовательности — $Y_0, Y_1, ..., Y_{k-i}$. Использование рекуррентных формул, как правило, сокращает текст

программы и время ее выполнения на компьютере. Однако в большинстве случаев рекуррентную формулу нужно написать программисту, что в ряде случаев вызывает определенные трудности.

**Алгоритмы со структурой вложенных циклов**

Внутри алгоритма циклической структуры может быть помещен другой цикл — вложенный (внутренний) цикл. Вложенный цикл должен полностью находиться в области внешнего цикла. Вложенный цикл может быть один, но может быть и несколько вложенных циклов. Второй вложен в первый, третий во второй и т. д. Ниже с помощью псевдокода представлена структура циклического алгоритма, содержащего несколько вложенных циклов:

```
Начало цикла 1;
  Начало цикла 2;
    Начало цикла 3;
    Тело цикла 3;
    Конец цикла 3;
  Конец цикла 2;
Конец цикла 1.
```

Для организации и внутреннего, и внешнего циклов могут использоваться разные типы алгоритмических структур (цикл с параметром, цикл с предусловием, цикл с постусловием).

На рисунке 3.4 представлена блок-схема алгоритма с внутренним циклом. В данном случае и внешний и внутренний циклы организованы на базе алгоритмической структуры «цикл с параметром».

На каждом шаге по внешнему циклу внутренний цикл выполняется несколько раз. Количество внутренних циклов на каждом внешнем цикле зависит от параметра внутреннего цикла.

Пусть, например, задано, что параметр внешнего цикла меняется от 1 до 5 с шагом 1, а параметр внутреннего цикла — от 1 до 10 с шагом 1.

Это означает, что на каждом шаге по внешнему циклу внутренний цикл будет выполняться 10 раз. Так как внешний цикл должен выполниться 5 раз, то внутренний цикл выполнится при этом 50 раз.

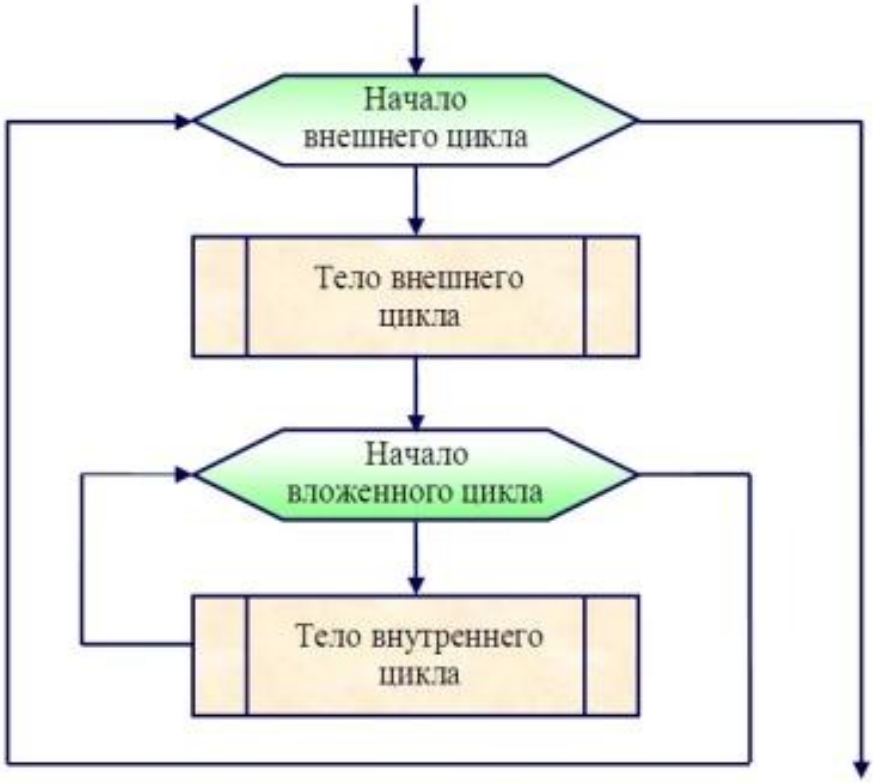



Рисунок 3.4 — Блок-схема алгоритма с внутренним циклом на базе алгоритмической структуры «цикл с параметром»

# 4 МОДЕЛИ РЕШЕНИЙ ФУНКЦИОНАЛЬНЫХ И ВЫЧИСЛИТЕЛЬНЫХ ЗАДАЧ

## 4.1 Программирование задач выбора и сортировки

Сортировка выбором заключается в поиске в каждом проходе по массиву максимального элемента из оставшихся и перемещении его на последнее место в просматриваемой части массива.

В первом проходе просматривается весь массив, во втором — нет смысла просматривать последний элемент, на третьем — два последних опускаются из области поиска и т. д.

Переменной — счетчику внешнего цикла — сначала присваивается индекс последнего элемента массива. Цикл выполняется до тех пор, пока индекс не дойдет до первого элемента (не включая его).

Во вложенном цикле осуществляется поиск максимального элемента. Достаточно сохранить в переменной лишь его индекс, а не значение. После вложенного цикла происходит обмен последнего элемента просматриваемой части массива (на него указывает счетчик внешнего цикла) на найденный максимум.

Метод сортировки прямым выбором основан на следующих правилах:

– выбирается элемент с наименьшим ключом;

– он меняется местами с первым элементом $a_0$;

– затем эти операции повторяются с оставшимися $n-1$ элементами, $n-2$ элементами и так далее до тех пор, пока не останется один, самый большой элемент.

### Найти максимальный и минимальный элемент массива

Максимальный элемент массива — это элемент, который имеет самое большое числовое значение, а минимальный элемент массива — это элемент, имеющий самое маленькое значение.

**Пример**. В массиве, состоящем из таких элементов: 3, 1, 0, минус 4, 16, 2 — максимальный элемент равен 16, т. к. это число больше других, а минимальный элемент равен 4, т. к. оно меньше остальных.

Алгоритм решения задачи:

– инициализация массива, переменных, хранящих минимальное и максимальное значение;

– заполнение массива случайными числами при помощи цикла и функции, возвращающей случайные числа;

– вывод массива;

– сравнение каждого элемента массива: если элемент больше переменной с максимальным значением, то значение записывается в переменную; если элемент меньше переменной с минимальным значением, то значение записывается в переменную;

– вывод переменных с максимальным и минимальным элементом.

## Поиск экстремума функции

Задачи поиска экстремума функции означают нахождение ее максимума (наибольшего значения) или минимума (наименьшего значения) в некоторой области ее аргументов. Ограничения значений аргументов, задающих эту область, как и прочие дополнительные условия, должны быть определены в виде системы неравенств и (или) уравнений. В таком случае говорят о задаче на условный экстремум. Поиск экстремума функции включает в себя задачи нахождения локального и глобального экстремума. Последние называют еще задачами оптимизации.

Для решения задач поиска максимума и минимума чаще всего применяются те же самые итерационные градиентные численные методы, что и для решения нелинейных уравнений. Для решения задачи на экстремум функции также следует задать начальное приближение — нулевую итерацию. Отличием является критерий, согласно которому строятся следующие итерации. В случае решения нелинейных уравнений и систем он заключается в поиске точки $x$, максимально близкой к нулю $f(x)$, а в случае задачи на экстремум — точки, увеличивающей (или уменьшающей) значение функции $f(x)$ от шага к шагу либо приближающей к нулю ее производную $f'(x)$ (в последнем случае, подразумевающем дифференцируемость функции $f$, задача поиска экстремума сводится к задаче решения нелинейного уравнения).

Значение функции в точке максимума называется локальным максимумом, значение функции в точке минимума — локальным минимумом данной функции. Локальные максимум и минимум функции называются локальными экстремумами.

Точка $x_0$ называется точкой строгого локального максимума функции $y = f(x)$, если для всех $x$ из окрестности этой точки будет справедливо строгое неравенство $f(x) < f(x_0)$.

Наибольшее или наименьшее значение функции на промежутке называется глобальным экстремумом.

**Необходимое условие экстремума**

Если функция $y = f(x)$ имеет экстремум в точке $x_0$, то ее производная $f'(x_0)$ либо равна нулю, либо не существует.

Поиск и сортировка в одномерных массивах

**Поиск**

Задачи поиска элемента, обладающего заданными свойствами, достаточно часто приходится решать применительно к массивам. Следует заметить, что если массив не отсортирован, то поиск неэффективен. Вообще с развитием информационных и информационно-поисковых систем рационализация алгоритмов поиска является актуальной задачей. Рассмотрим простейший алгоритм поиска для отсортированного массива (для определенности пусть элементы массива расположены по возрастанию).

Алгоритм носит название двоичного (бинарного) поиска, т. к. на каждом шаге область поиска уменьшается вдвое.

Пусть в отсортированном массиве требуется найти элемент со значением $x$ или указать, что такого элемента там нет. Выберем средний элемент. Для этого элемента относительно значения $x$ возможны три случая:

1) элемент равен $x$ (поиск завершен);

2) элемент больше $x$ (поиск необходимо продолжить в левой части массива);

3) элемент меньше $x$ (поиск необходимо продолжить в правой части массива).

В случаях 2–3 поиск (если это ещё возможно) продолжается. Для этого в выделенной части массива вновь выбирается средний элемент и проводятся аналогичные рассуждения. И так далее, до тех пор, пока поиск не будет завершен.

Поиск завершается в одном из двух случаев:

– элемент найден;

– элемент не найден (это констатируется в том случае, когда длина области поиска уменьшилась до нуля, т. е. левая и правая границы области поиска сомкнулись).

### Сортировка

Обычно сортировку подразделяют на два класса: внутреннюю и внешнюю. При внутренней сортировке все элементы хранятся в оперативной памяти, как правило, это сортировка массивов. При внешней сортировке элементы хранятся на внешнем запоминающем устройстве, это сортировка файлов.

Одно из основных требований к методам сортировки — экономное использование памяти. Это означает, что переупорядочение нужно выполнять «на том же месте», то есть методы пересылки элементов из одного массива в другой не представляют интереса.

Удобной мерой эффективности является подсчет чисел $C$ — сравнений элементов, и $M$ — присваиваний элементов. Достаточно хороший алгоритм затрачивает на сортировку $N$ элементов время порядка $N \cdot \log_2(N)$. Простейшие алгоритмы сортировки, которые мы рассмотрим в этом разделе, обладают характеристикой порядка $N_2$. Если $N$ достаточно мало, то простые алгоритмы выгодно использовать в силу простоты их реализации.

### Сортировка выбором

Алгоритм сортировки. Пусть часть массива до $i-1$ элемента включительно отсортирована. Выбираем в несортированной части минимальный элемент и меняем его местами с $i$-м.

Изначально отсортированная часть состоит из нулевого количества элементов.

**Сортировка обменом (пузырьковая)**

Массив просматривается $N - 1$ раз. При каждом просмотре сравниваются каждые два соседних элемента. Если элемент с меньшим индексом оказывается больше, производится их обмен.

**Сортировка вставками**

Пусть часть массива отсортирована. Выбираем в неотсортированной части очередной элемент и вставляем его в отсортированную так, чтобы упорядоченность элементов сохранилась. При поиске места вставки осуществляем сдвиг элементов в сторону возрастания номеров.

**Метод простого ранжирования**

Метод представляет собой процедуру упорядочения объектов, выполняемую экспертом. На основе знаний и опыта эксперт располагает объекты в порядке предпочтения, руководствуясь одним или несколькими выбранными показателями сравнения. В зависимости от вида отношений между объектами возможны различные варианты упорядочения объектов.

## 4.2 Машинное преобразование матриц

### Метод Гаусса приведения матрицы к ступенчатому виду

Элементарными преобразованиями матрицы называются следующие ее преобразования:

– перестановка двух столбцов (строк) матрицы;

– умножение всех элементов одного столбца (строки) матрицы на одно и то же число, отличное от нуля;

– прибавление к элементам одного столбца (строки) соответствующих элементов другого столбца (строки), умноженных на одно и то же число.

Матрица $B$, полученная из исходной матрицы $A$ конечным числом элементарных преобразований, называется эквивалентной. Это обозначается $A \sim B$.

Элементарные преобразования применяются для упрощения матриц, что будет в дальнейшем использоваться для решения разных задач.

Покажем, как при помощи элементарных преобразований можно привести матрицу к ступенчатому виду. Здесь высота каждой «ступеньки» составляет одну строку, символом 1 (единицей) обозначены единичные элементы матрицы, символом «*» обозначены элементы с произвольными значениями, остальные элементы матрицы нулевые. К ступенчатому виду можно привести любую матрицу, причем достаточно использовать только элементарные преобразования строк матрицы (рисунок 4.1).

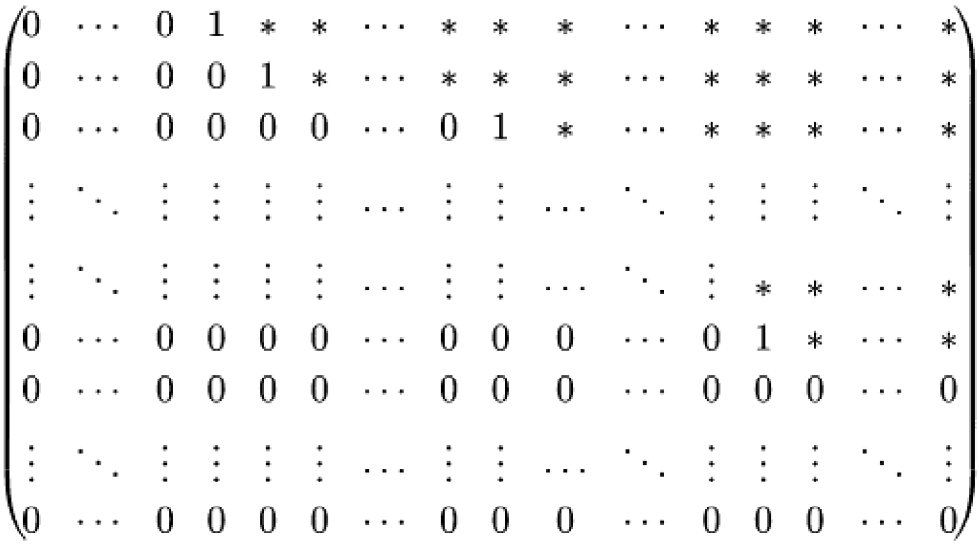


Рисунок 4.1 — Приведение матрицы к ступенчатому виду

### Алгоритм приведения матрицы к ступенчатому виду

Чтобы привести матрицу к ступенчатому виду, нужно выполнить определенные действия.

В первом столбце выбрать элемент отличный от нуля (ведущий элемент). Строку с ведущим элементом (ведущая строка), если она не первая, переставить на место первой строки (преобразование I типа). Если в первом столбце нет ведущего (все элементы равны нулю), то исключаем этот столбец, и продолжаем поиск ведущего элемента в оставшейся части матрицы. Преобразования заканчиваются, если исключены все столбцы или в оставшейся части матрицы все элементы нулевые.

Разделить все элементы ведущей строки на ведущий элемент (преобразование II типа). Если ведущая строка последняя, то на этом преобразования следует закончить.

К каждой строке, расположенной ниже ведущей, прибавить ведущую строку, умноженную соответственно на такое число, чтобы элементы, стоящие под ведущим оказались равными нулю (преобразование III типа).

Исключив из рассмотрения строку и столбец, на пересечении которых стоит ведущий элемент, перейти к пункту 1, в котором все описанные действия применяются к оставшейся части матрицы.

### Вычисление определителей

Вычисления определителей второго порядка

Чтобы вычислить определитель матрицы $A$ второго порядка (рисунок 4.2), надо от произведения элементов главной диагонали отнять произведение элементов побочной диагонали.

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11} \cdot a_{22} - a_{12} \cdot a_{21}.$$

Рисунок 4.2 — Вычисление определителя матрицы A

Методы вычисления определителей третьего порядка

### Правило треугольника

Схематически это правило изображено на рисунке 4.3.

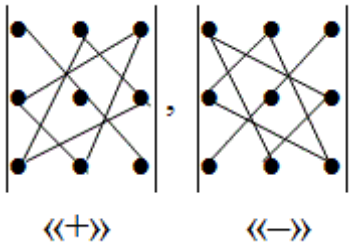



Рисунок 4.3 — Правило треугольника

Произведение элементов в первом определителе, которые соединены прямыми, берется со знаком «плюс», аналогично для второго определителя — соответствующие произведения берутся со знаком «минус» (рисунок 4.4).

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} -$$

$$-a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33} - a_{13}a_{22}a_{31}.$$

Рисунок 4.4 — Решение матрицы «методом треугольника»

**Правило Саррюса**

Справа от определителя дописывают первых два столбца, произведения элементов на главной диагонали и на диагоналях, ей параллельных, берут со знаком «плюс», а произведения элементов побочной диагонали и диагоналей, ей параллельных — со знаком «минус» (рисунок 4.5).

$$\begin{array}{|ccc|cc} a_{11} & a_{12} & a_{13} & a_{11} & a_{12} \\ a_{21} & a_{22} & a_{23} & a_{21} & a_{22} \\ a_{31} & a_{32} & a_{33} & a_{31} & a_{32} \end{array} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} -$$

$$-a_{13}a_{22}a_{31} - a_{11}a_{23}a_{32} - a_{12}a_{21}a_{33}.$$

Рисунок 4.5 — Решение матрицы с применением правила Саррюса

## Нахождение обратной матрицы

Матрица $A^{-1}$ называется обратной матрицей по отношению к матрице $A$, если $A \cdot A^{-1} = E$, где $E$ — единичная матрица $n$-го порядка. Обратная матрица может существовать только для квадратных матриц.

**Алгоритм нахождения обратной матрицы:**

– определение, является ли матрица квадратной. Если нет, то обратной матрицы для нее не существует;

– вычисление определителя матрицы А. Если он не равен нулю, продолжаем решение, иначе — обратной матрицы не существует;

– нахождение транспонированной матрицы $A^{\mathrm{T}}$;

– определение алгебраических дополнений. Заменяют каждый элемент матрицы его алгебраическим дополнением;

– составление обратной матрицы из алгебраических дополнений: каждый элемент полученной матрицы делят на определитель исходной матрицы. Результирующая матрица является обратной для исходной матрицы;

– делают проверку: перемножают исходную и полученную матрицы. В результате должна получиться единичная матрица.

## Вычисление собственных значений матриц

***Определение 1.*** Собственным значением (или характеристическим числом) квадратной матрицы $A$ называется такое число $\lambda$, что для некоторого ненулевого вектора $X$ имеет место равенство $A \cdot X = \lambda \cdot X$.

***Определение 2.*** Любой ненулевой вектор $X$, удовлетворяющий этому равенству, называется собственным вектором матрицы $A$, соответствующим (или принадлежащим) собственному значению $\lambda$.

Очевидно, что все собственные векторы матрицы $A$ определяются с точностью до числового множителя.

Ценность сведений о собственных значениях матрицы $A$ не вызывает сомнений. В самом деле сходимость и скорость сходимости метода простой итерации, применяемого для приближенного решения системы линейных алгебраических уравнений, существенно зависят от величины максимального по модулю собственного значения матрицы $B$: $X = B \cdot X + b$.

Заметим, что задача нахождения собственных значений и собственных векторов матрицы важна не только как вспомогательная. Многие прикладные задачи физики, механики, астрономии, радиофизики приводят исследователя к проблеме определения нетривиального решения однородной системы линейных алгебраических уравнений и тех значений числового параметра $\lambda$, при которых такое решение существует.

Проблема собственных значений играет существенную роль во всех явлениях неустойчивых колебаний и вибраций, т. к. частота колебаний определяется собственными значениями некоторой матрицы, форму же этих колебаний указывают собственные векторы этой матрицы. Анализ собственных значений матриц является важной темой научно-технических исследований.

Условием существования нетривиального решения $y$ системы (или, что то же, $(A-\lambda\cdot E)X=0$, где $E$ — единичная матрица) является требование, приведенное на рисунке 4.6.

$$\det|A-\lambda E| = \det\begin{pmatrix} a_{11}-\lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22}-\lambda & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn}-\lambda \end{pmatrix} = 0$$

Рисунок 4.6 — Решение матрицы

Обычно это уравнение (рисунок 4.7) называется характеристическим (или вековым) уравнением матрицы $A$.

$$\det|A-\lambda E| = (-1)^n\left(\lambda^n - P_1\lambda^{n-1} - P_2\lambda^{n-2} - \ldots - P_n\right)$$

Рисунок 4.7 — Характеристическое уравнение матрицы А

Левая часть этого уравнения называется характеристическим полиномом матрицы (рисунок 4.8). Вместо полинома вида рассматривают полином, отличающийся от характеристического множителем $(-1)^n$. Этот полином имеет вид и обычно называется собственным многочленом матрицы. Собственные значения матрицы есть корни её собственного многочлена $P(\lambda)$.

$$P(\lambda) = \lambda^n - P_1\lambda^{n-1} - P_2\lambda^{n-2} - \ldots - P_n$$

Рисунок 4.8 — Характеристический полином матрицы

Совокупность всех собственных значений $\lambda_1 \ldots \lambda_n$ матрицы $A$, где каждое $\lambda_i$ выписано столько раз, какова его кратность как корня $P(\lambda)$, называется спектром этой матрицы.

## 4.3 Решение систем линейных алгебраических уравнений

В практической деятельности человека в различных областях наук широко применяются системы линейных уравнений. Без них не обходятся ни в метеорологии, ни в медицине, ни в технике. Этим и обусловливается интерес к этой теме.

Система $m$ линейных алгебраических уравнений с $n$ неизвестными в линейной алгебре — это система уравнений вида, приведенного на рисунке 4.9.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

Рисунок 4.9 — Система $m$ линейных алгебраических уравнений с $n$ неизвестными

Здесь $m$ — количество уравнений; $n$ — количество неизвестных; $x_1, x_2, \ldots, x_n$ — неизвестные, которые надо определить; $a_{11}, a_{12}, \ldots, a_{mn}$ — коэффициенты при неизвестных системы; $b_1, b_2, \ldots, b_m$ — свободные члены. Индексы коэффициентов $\left(a_{ij}\right)$ системы обозначают номера уравнения ($i$) и неизвестного ($j$), при котором стоит этот коэффициент соответственно.

Решение системы $m$ линейных алгебраических уравнений с $n$ неизвестными — это совокупность $n$ чисел $\left(c_1, c_2, \ldots, c_n\right)$ таких, что подстановка каждого $c_i$ вместо $x_i$ в систему обращает все ее уравнения в тождества.

Система линейных уравнений может быть представлена в матричной форме (рисунок 4.10).

Системы $m$ линейных алгебраических уравнений с $n$ неизвестными решаются различными методами.

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_m \end{pmatrix}$$

Рисунок 4.10 — Система линейных уравнений в матричной форме

Прямые (или точные) методы позволяют найти решение за определенное количество шагов. Итерационные методы основаны на использовании повторяющегося процесса и позволяют получить решение в результате последовательных приближений.

### Метод Гаусса-Жордана

Метод Гаусса-Жордана (метод полного исключения неизвестных) — это метод, который используется для решения систем линейных алгебраических уравнений (СЛАУ), нахождения обратной матрицы, нахождения координат вектора в заданном базисе или отыскания ранга матрицы. Метод является модификацией метода Гаусса. Назван в честь К. Ф. Гаусса и немецкого геодезиста и математика Вильгельма Йордана.

Алгоритм этого метода таков:

– выбирают первый слева столбец матрицы, в котором есть хоть одно отличное от нуля значение;

– если самое верхнее число в этом столбце есть ноль, то меняют всю первую строку матрицы с другой строкой матрицы, где в этой колонке нет нуля;

– все элементы первой строки делят на верхний элемент выбранного столбца;

– из оставшихся строк вычитают первую строку, умноженную на первый элемент соответствующей строки, с целью получить первым элементом каждой строки (кроме первой) ноль;

– далее проводят такую же процедуру с матрицей, получающейся из исходной матрицы после вычёркивания первой строки и первого столбца;

– после повторения этой процедуры один раз получают верхнюю треугольную матрицу;

– вычитают из предпоследней строки последнюю строку, умноженную на соответствующий коэффициент, с тем чтобы в предпоследней строке осталась только 1 на главной диагонали;

– повторяют предыдущий шаг для последующих строк. В итоге получают единичную матрицу и решение на месте свободного вектора (с ним необходимо проводить все те же преобразования);

– чтобы получить обратную матрицу, нужно применить все операции в том же порядке к единичной матрице.

Методом Гаусса можно решать системы линейных алгебраических уравнений любого вида без предварительного их исследования на совместность. Процесс последовательного исключения неизвестных переменных позволяет сделать вывод как о совместности, так и о несовместности СЛАУ, а в случае существования решения дает возможность отыскать его.

С точки зрения вычислительной работы метод Гаусса является предпочтительным.

## Метод Крамера

Метод Крамера — это способ решения квадратных систем линейных алгебраических уравнений с ненулевым определителем основной матрицы (причем для таких уравнений решение существует и единственное). Назван по имени Габриэля Крамера (1704–1752), придумавшего метод.

### Описание метода

Для системы $n$ линейных уравнений с $n$ неизвестными (над произвольным полем) с определителем $\Delta$ основной матрицы системы, отличным от нуля (рисунок 4.11).

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

Рисунок 4.11 — Система $n$ линейных уравнений с $n$ неизвестными

Решение данной матрицы записывается в виде, приведенном на рисунке 4.12, где *i*-й столбец матрицы системы заменен столбцом свободных членов.

В высшей алгебре описаны и другие методы решения систем линейных уравнений. Но все они имеют ограниченную сферу использования.

$$x_i = \frac{1}{\Delta}\begin{vmatrix} a_{11} & \dots & a_{1,i-1} & b_1 & a_{1,i+1} & \dots & a_{1n} \\ a_{21} & \dots & a_{2,i-1} & b_2 & a_{2,i+1} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n-1,1} & \dots & a_{n-1,i-1} & b_{n-1} & a_{n-1,i+1} & \dots & a_{n-1,n} \\ a_{n1} & \dots & a_{n,i-1} & b_n & a_{n,i+1} & \dots & a_{nn} \end{vmatrix}$$

Рисунок 4.12 — Решение системы n линейных уравнений

При решении практических задач приходится находить решения систем линейных уравнений с большой степенью точности, потому что целочисленные решения встречаются довольно редко. Кроме того, математические модели некоторых явлений, процессов приводят к системам линейных уравнений с большим числом уравнений и неизвестных (например, при составлении прогноза погоды специалисты получают системы, в которых до пятидесяти уравнений с таким же числом неизвестных). Это приводит к громоздким вычислениям или вообще к невозможности решить систему линейных уравнений точными методами.

Поэтому в таких случаях используют итерационные методы, которые позволяют для их реализации использовать компьютерную технику и информационные технологии.

**Итерационные методы** устанавливают процедуру уточнения определённого начального приближения к решению. При выполнении условий сходимости они позволяют достичь любой точности просто повторением итераций. Преимущество этих методов в том, что часто они позволяют достичь решения с заранее заданной точностью быстрее, а также решать большие системы уравнений.

В численных методах разработан целый ряд итерационных методов: метод Якоби, метод Зейделя, метод минимальных невязок и другие.

Все итерационные методы были созданы давно, однако период их бурного развития и внедрения в практику начался с появлением и развитием электронно-вычислительной техники. И сейчас под итерационным методом понимается такая интерпретация математической модели, которая доступна для компьютерной реализации.

### Метод Гаусса с выбором главного элемента

Заметим, что в методе последовательного исключения Гаусса вычисления возможны, если ведущие элементы системы $a_{kk}^{(k-1)} \neq 0$. Добиться выполнения этого условия можно, переставляя элементы строк и столбцов матрицы. Но среди ведущих элементов могут оказаться очень маленькие по абсолютной величине. При делении на такие ведущие элементы получается большая погрешность округления (вычислительная погрешность).

Чтобы избежать сильного влияния вычислительной погрешности на решение, применяется метод Гаусса с выбором главного элемента.

Рассмотрим систему (рисунок 4.13).

$$S = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1i}^{(0)} & \dots & a_{1n}^{(0)} & a_{1,n+1}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & \dots & a_{2i}^{(0)} & \dots & a_{2n}^{(0)} & a_{2,n+1}^{(0)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{m1}^{(0)} & a_{m2}^{(0)} & \dots & a_{mi}^{(0)} & \dots & a_{mn}^{(0)} & a_{m,n+1}^{(0)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & \dots & a_{ni}^{(0)} & \dots & a_{nn}^{(0)} & a_{n,n+1}^{(0)} \end{pmatrix}$$

Рисунок 4.13 — Прямоугольная матрица

Среди элементов матрицы $a_{ij}^{(0)}\left(i, j = \overline{1,n}\right)$ выбираем наибольший по модулю элемент, называемый главным. Например, пусть им будет элемент $a_{ml}^{(0)}$. Строка с номером $m$, содержащая главный элемент, называется главной строкой.

Далее вычисляем множители $m_i = -\dfrac{a_{il}^{(0)}}{a_{ml}^{(0)}}$ для всех $i \neq m$. Затем матрица $S$ преобразуется так: к каждой $i$-й неглавной строке, прибавим почленно главную строку, умножив её на $m_i$. В результате получим матрицу, у которой все элементы $l$-го столбца, за исключением $a_{ml}^{(0)}$, равны 0. Отбрасывая этот столбец и главную строку, получаем новую матрицу $S_1$ с меньшим на единицу числом строк и столбцов.

Над матрицей $S_1$ повторяем те же операции, после чего получаем матрицу $S_2$ и т. д. Эти преобразования продолжаются до тех пор, пока не получится матрица, содержащая одну строку из двух элементов, которая тоже считается главной. Затем объединяем все главные строки, начиная с последней. После некоторой перестановки они образуют треугольную матрицу, эквивалентную исходной. На этом заканчивается прямой ход метода Гаусса с выбором главного элемента.

Далее находим $x_i$, $i = \overline{1,n}$, решая систему с треугольной матрицей. Это обратный ход.

Контроль правильности вычислений в методе Гаусса с выбором главного элемента осуществляется аналогично ранее описанному в методе последовательного исключения неизвестных.

## 4.4 Численное решение нелинейных уравнений

Пусть имеется уравнение вида:

$$f(x) = 0, \tag{4.1}$$

где $f(x)$ — заданная алгебраическая или трансцендентная функция.

Решить уравнение — значит найти все его корни, то есть те значения $x$, которые обращают уравнение в тождество. Если уравнение достаточно сложно, то задача точного определения корней является в некоторых случаях нерешаемой. Поэтому ставится задача найти такое приближенное значение корня $x_{\text{пр}}$, которое отличается от точного значения корня $x^*$ на величину, по модулю не превышающую

указанной точности (малой положительной величины) ε, то есть $\left|x^* - x_{\text{пр}}\right| < \varepsilon$.

Величину ε также называют допустимой ошибкой, которую можно задать по своему усмотрению.

**Этапы приближенного решения нелинейных уравнений:**

– отделение корней, то есть нахождение интервалов из области определения функции $f(x)$, в каждом из которых содержится только один корень уравнения $f(x) = 0$;

– уточнение корней до заданной точности.

**Отделение корней**

Отделение корней можно проводить графически и аналитически. Для того чтобы графически отделить корни уравнения, необходимо построить график функции $f(x)$. Абсциссы точек его пересечения с осью *Ox* являются действительными корнями уравнения.

Для примера рассмотрим задачу решения уравнения:

$$\sin(x) = \frac{1}{x}, \tag{4.2}$$

где угол *x* задан в градусах. Указанное уравнение можно переписать в виде:

$$\sin\left(\frac{\pi}{180}x\right) - \frac{1}{x} = 0. \tag{4.3}$$

Для графического отсечения корней достаточно построить график функции (рисунок 4.14).

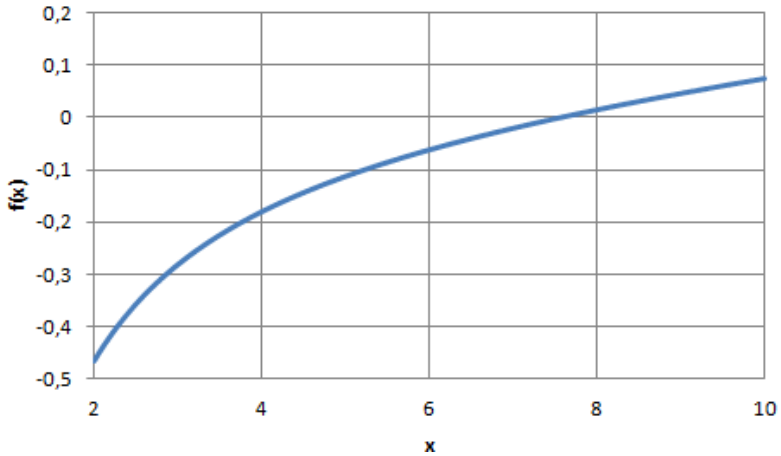



Рисунок 4.14 — График функции

Из рисунка видно, что корень уравнения лежит в промежутке $x \in (6;8)$.

**Аналитическое отделение корней**

Аналитическое отделение корней основано на следующих теоремах.

**Теорема 1.** Если непрерывная функция $f(x)$ принимает на концах отрезка $[a;b]$ значения разных знаков, т. е.

$$f(a) \cdot f(b) < 0, \tag{4.4}$$

то на этом отрезке содержится по крайней мере один корень уравнения.

**Теорема 2.** Если непрерывная на отрезке $[a;b]$ функция $f(x)$ принимает на концах отрезка значения разных знаков, а производная $f'(x)$ сохраняет знак внутри указанного отрезка, то внутри отрезка существует единственный корень уравнения $f(x) = 0$.

## Метод дихотомии

Метод дихотомии получил свое название от древнегреческого слова, что в переводе означает деление надвое. Именно поэтому данный метод имеет еще второе название: метод половинного деления. Допустим, играя в игру «Угадай число», один игрок загадывает число от 1 до 100, а другой пытается его отгадать, руководствуясь подсказками «больше» или «меньше». Логично предположить, что первым числом будет названо 50, а вторым, в случае если оно меньше, 25, если больше — 75. Таким образом, на каждом этапе (итерации) неопределенность неизвестного уменьшается в 2 раза. Таким образом, даже самый невезучий в мире человек отгадает загаданное число в данном диапазоне за 7 предположений вместо 100 случайных утверждений.

**Метод половинного деления в решении уравнения**

Правильное решение уравнения методом половинного деления возможно лишь в том случае, если известно, что на заданном

интервале имеется корень и он является единственным. Это совсем не означает, что метод дихотомии может использоваться только для решения линейных уравнений. Для нахождения корней уравнений более высокого порядка методом половинного деления необходимо сначала отделить корни по отрезкам. Процесс отделения корней осуществляется путем отыскания первой и второй производной от функции и приравнивания их к нулю $f'(x)$ и $f''(x)$. Далее определяются знаки $f(x)$ в критических и граничных точках. Интервал, где функция меняет знак $|a,b|$, где $f(a)\cdot f(b)<0$.

**Алгоритм метода дихотомии**

Алгоритм метода дихотомии очень прост. Рассмотрим отрезок $|a,b|$ в пределах которого имеется один корень $x_1$ (рисунок 4.15).

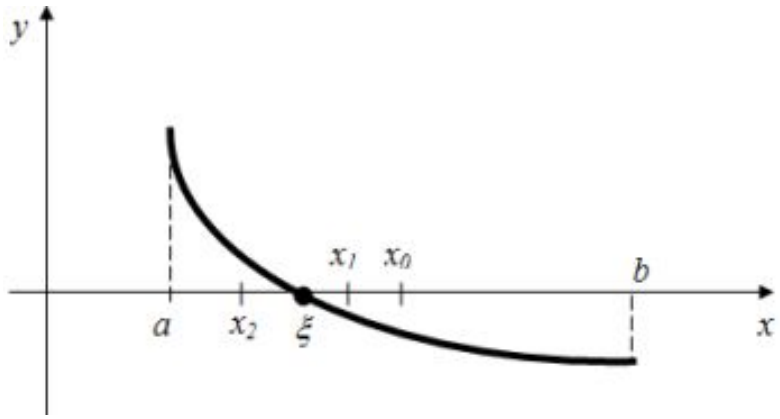



Рисунок 4.15 — Отрезок $|a,b|$

На первом этапе вычисляется

$$x_0=\frac{(a+b)}{2}. \qquad (4.5)$$

Далее определяется значение функции в этой точке: если $f(x_0)<0$, то$[a,x_0]$, если наоборот, то $[x_0,b]$, т. е происходит сужение интервала. Таким образом, в результате формируется последовательность $x_i$, где $i$ — номер итерации.

Вычисления прекращаются, когда разность $b-a$ меньше требуемой погрешности.

Графическая интерпретация метода дихотомии представлена на рисунке 4.16

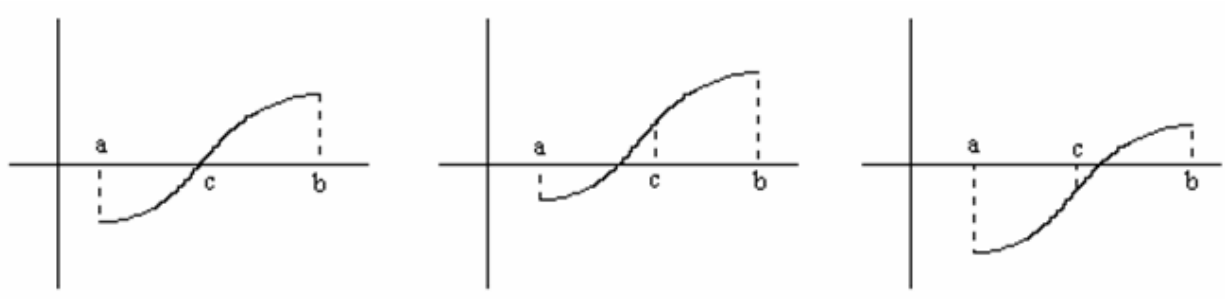



Рисунок 4.16 — Графическое представление метода дихотомии

### Комбинированный метод хорд и касательных

Методы хорд и касательных дают приближения корня с разных сторон. Поэтому их часто применяют в сочетании друг с другом, тогда уточнение корня происходит быстрее.

Пусть дано уравнение $f(x)=0$, корень отделен на отрезке $[a,b]$.

Рассмотрим случай, когда $f'(x)f''(x)>0$ (рисунок 4.17).

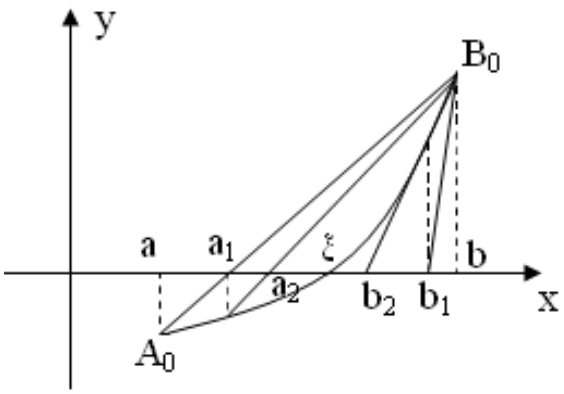



Рисунок 4.17 — График функции $f'(x)f''(x)>0$

В этом случае метод хорд дает приближенное значение корня с недостатком (конец $b$ неподвижен), а метод касательных — с избытком (за начальное приближение берем точку $b$).

Тогда вычисления следует проводить по формулам:

$$a_1 = a - \frac{f(a)(b-a)}{f(b)-f(a)}, \tag{4.6}$$

$$b_1 = b - \frac{f(b)}{f'(b)}. \tag{4.7}$$

Теперь корень $\xi$ заключен в интервале $[a_1, b_1]$. Применяя к этому отрезку комбинированный метод, получим:

$$a_2 = a_1 - \frac{f(a_1)(b_1 - a_1)}{f(b_1) - f(a)}, \tag{4.8}$$

$$b_2 = b_1 - \frac{f(b_1)}{f'(b_1)}, \tag{4.9}$$

и т. д.

$$a_{n+1} = a_n - \frac{f(a_n)(b_n - a_n)}{f(b_n) - f(a_n)}, \tag{4.10}$$

$$b_{n+1} = b_n - \frac{f(b_n)}{f'(b_n)}. \tag{4.11}$$

Если же $f'(x)f''(x) < 0$ (рисунок 4.18), то, рассуждая аналогично, получим следующие формулы для уточнения корня уравнения:

$$a_{n+1} = a_n - \frac{f(a_n)}{f'(a_n)}, \tag{4.12}$$

$$b_{n+1} = b_n - \frac{f(b_n)(b_n - a_n)}{f(b_n) - f(a_n)}. \tag{4.13}$$

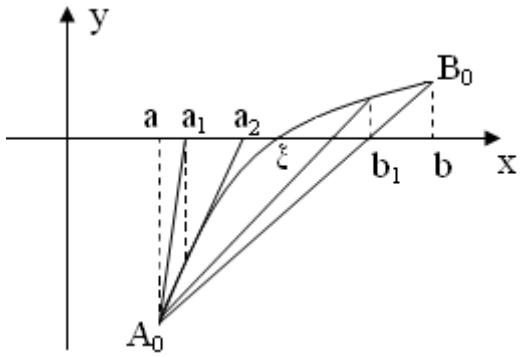



Рисунок 4.18 — График функции $f'(x)f''(x) < 0$

Вычислительный процесс прекращается, как только выполнится условие:

$$|b_{n+1} - a_{n+1}| < \xi, \tag{4.14}$$

### Метод последовательных приближений (метод итераций)

**Метод итерации** — численный метод решения математических задач, используемый для приближённого решения алгебраических уравнений и систем. Суть метода заключается в нахождении по приближённому значению величины следующего приближения (являющегося более точным). Метод позволяет получить решение с заданной точностью в виде предела последовательности итераций. Характер сходимости и сам факт сходимости метода зависит от выбора начального приближения решения. Функциональное уравнение может быть записано в виде:

$$x = f(x). \tag{4.15}$$

Функцию $f(x)$ называют сжимающим отображением. Последовательность чисел $x_0, x_1, \ldots, x_n$ называется итерационной, если для любого номера $n > 0$ элемент $x_n$ выражается через элемент $x_{n-1}$ по рекуррентной формуле $x_n = f(x_{n-1})$, а в качестве $x_0$ взято любое число из области задания функции $f(x)$.

## 4.5 Численные методы интегрирования

В ряде задач возникает необходимость вычисления определенного интеграла от некоторой функции:

$$I = \int_a^b f(x) \cdot dx, \tag{4.16}$$

где $f(x)$ — подынтегральная функция, непрерывная на отрезке $[a,b]$.

Геометрический смысл интеграла заключается в том, что если $f(x) \geq 0$ на отрезке $[a,b]$, то интеграл $\int_a^b f(x) \cdot dx$ численно равен площади фигуры, ограниченной графиком функции $y = f(x)$, отрезком оси абсцисс, прямой $x = a$ и прямой $x = b$ (рисунок 4.19). Таким образом, вычисление интеграла равносильно вычислению площади криволинейной трапеции.

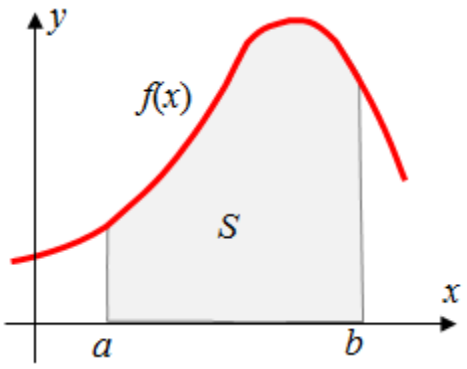



Рисунок 4.19 — Геометрический смысл интеграла

Задача численного интегрирования состоит в замене исходной подынтегральной функции некоторой аппроксимирующей функцией (обычно полиномом).

Численное интегрирование применяется, когда:

– сама подынтегральная функция не задана аналитически, а, например, представлена в виде таблицы значений;

– аналитическое представление подынтегральной функции известно, но ее первообразная не выражается через аналитические функции.

### Метод прямоугольников

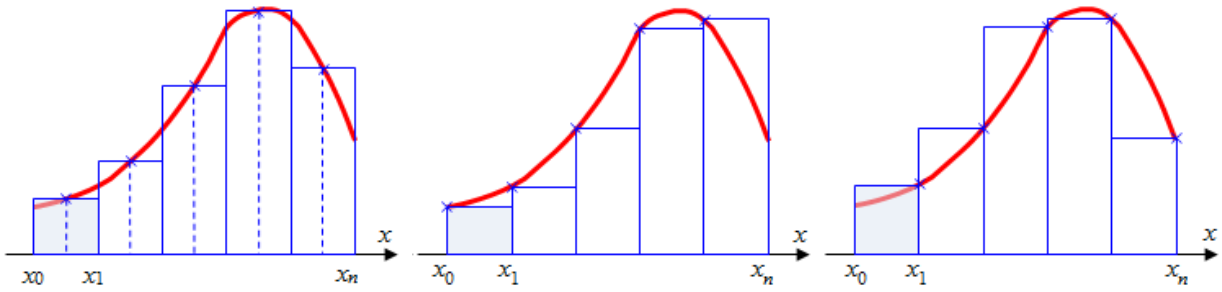
Одним из простейших методов численного интегрирования является метод прямоугольников. На частичном отрезке $\left[x_{j-1}, x_j\right]$ подынтегральную функцию заменяют полиномом Лагранжа нулевого порядка, построенным в одной точке.

Графическая иллюстрация метода средних прямоугольников представлена на рисунке 4.20,*а*. Из рисунка видно, что площадь криволинейной трапеции приближенно заменяется площадью многоугольника, составленного из $N$ прямоугольников. Таким образом, вычисление определенного интеграла сводится к нахождению суммы площадей $N$ элементарных прямоугольников.

$$\int_a^b f\left(x\right)\cdot dx \approx \sum_{j=1}^{N} h\cdot f\left(x_{j-1}\right), \tag{4.17}$$

$$\int_a^b f\left(x\right)\cdot dx \approx \sum_{j=1}^{N} h\cdot f\left(x_{j}\right). \tag{4.18}$$

Формулы (4.17) и (4.18) называются формулой левых и правых прямоугольников соответственно. Графически метод левых и правых прямоугольников представлен на рисунках 4.20,*б*,*в*. Однако из-за нарушения симметрии в формулах правых и левых прямоугольников их погрешность значительно больше, чем в методе средних прямоугольников.




*а* *б* *в*

Рисунок 4.20 — Интегрирование методом прямоугольников:
*а* — средние прямоугольники; *б* — левые прямоугольники;
*в* — правые прямоугольники

### Метод трапеций

Если на частичном отрезке $\left[x_{j-1},x_j\right]$ подынтегральную функцию заменить полиномом Лагранжа первой степени

$$f\left(x\right)=L_{1,j}\left(x\right)=\frac{1}{h}\left[\left(x-x_{j-1}\right)f\left(x_j\right)-\left(x-x_j\right)f\left(x_{j-1}\right)\right], \quad (4.19)$$

то искомый интеграл на частичном отрезке запишется следующим образом:

$$\int_{x_{j-1}}^{x_j} f\left(x\right)dx \approx \frac{1}{h}\left[f\left(x_j\right)\int_{x_{j-1}}^{x_j}\left(x-x_{j-1}\right)dx - f\left(x_{j-1}\right)\int_{x_{j-1}}^{x_j}\left(x-x_j\right)dx\right]=$$

$$=\frac{f\left(x_{j-1}\right)+f\left(x_j\right)}{2}h\,. \quad (4.20)$$

Тогда составная формула трапеций на всем отрезке интегрирования $\left[a,b\right]$ примет вид:

$$\int_a^b f(x)dx \approx \sum_{j=1}^{N} \frac{f(x_j)+f(x_{j-1})}{2} h = h\left[\frac{1}{2}(f_1+f_N)+f_2+\ldots+f_{N-1}\right]. \quad (4.21)$$

Графически метод трапеций представлен на рисунке 4.21. Площадь криволинейной трапеции заменяется площадью многоугольника, составленного из $N$ трапеций, при этом кривая заменяется вписанной в нее ломаной. На каждом из частичных отрезков функция аппроксимируется прямой, проходящей через конечные значения, при этом площадь трапеции на каждом отрезке определяется по формуле (4.21).

Погрешность метода трапеций выше, чем у метода средних прямоугольников. Однако на практике найти среднее значение на элементарном интервале можно только у функций, заданных аналитически (а не таблично), поэтому использовать метод средних прямоугольников удается далеко не всегда.

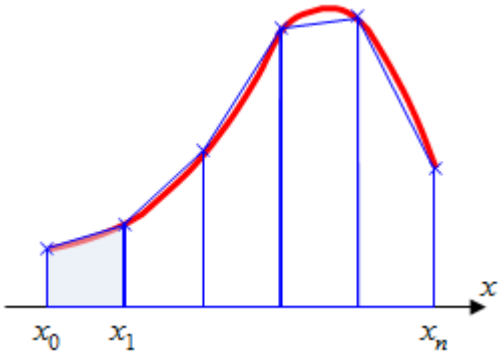



Рисунок 4.21 — Интегрирование методом трапеций

### Метод Симпсона

В этом методе подынтегральная функция на частичном отрезке $\left[x_{j-1}, x_j\right]$ аппроксимируется параболой, проходящей через три точки $x_{j-1}, x_{j-0,5}, x_j$, то есть интерполяционным многочленом Лагранжа второй степени:

$$f(x) = L_{2,j}(x) = \frac{2}{h^2}\left[\left(x-x_{j-0,5}\right)\left(x-x_j\right)f\left(x_{j-1}\right) - \right.$$

$$\left. -2\cdot\left(x-x_{j-1}\right)\left(x-x_j\right)f\left(x_{j-0,5}\right)+\left(x-x_{j-1}\right)\left(x-x_{j-0,5}\right)f\left(x_j\right)\right]. \quad (4.22)$$

Проведя интегрирование, получим:

$$\int_{x_{j-1}}^{x_j} f(x)dx \approx \frac{h}{6}\left(f_{j-1} + 4f_{j-0,5} + f_j\right). \qquad (4.23)$$

Это и есть формула Симпсона или формула парабол. На отрезке $[a,b]$ формула Симпсона примет вид:

$$\int_a^b f(x)dx \approx$$

$$\approx \frac{h}{6}\left[f_0 + f_N + 2\left(f_1 + f_2 + \ldots + f_{N-1}\right) + 4\left(f_{0,5} + f_{1,5} + f_{2,5} + \ldots + f_{N-0,5}\right)\right] =$$

$$= \frac{h}{6}\left[f_0 + f_N + 2\cdot\sum_{j=1}^{N-1} f_j + 4\cdot\sum_{j=0,5}^{N-0,5} f_j\right]. \qquad (4.24)$$

Если разбить отрезок интегрирования $[a,b]$ на четное количество $2N$ равных частей с шагом $h = \frac{b-a}{2N}$, то можно построить параболу на каждом сдвоенном частичном отрезке $\left[x_{j-1}, x_j\right]$ и переписать выражения (4.22)–(4.24) без дробных индексов. Тогда формула Симпсона примет вид:

$$\int_a^b f(x)dx \approx$$

$$\approx \frac{h}{3}\left[f_0 + f_{2N} + 2\left(f_2 + f_4 + \ldots + f_{2N-2}\right) + 4\left(f_1 + f_3 + f_5 + \ldots + f_{2N-1}\right)\right] =$$

$$= \frac{h}{3}\left[f_0 + f_{2N} + 2\cdot\sum_{j=2,2}^{2N-2} f_j + 4\cdot\sum_{j=1,2}^{2N-1} f_j\right]. \qquad (4.25)$$

Графическое представление метода Симпсона показано на рисунке 4.22. На каждом из сдвоенных частичных отрезков заменяем дугу данной кривой параболой.

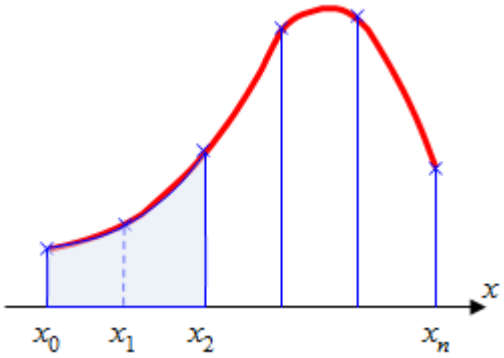


Рисунок 4.22 — Метод Симпсона

### Адаптивный алгоритм

Основная идея адаптивного кодирования заключается в том, что компрессор и декомпрессор начинают работать с «пустого» дерева Хаффмана, а потом модифицируют его по мере чтения и обработки символов. Соответственно, и кодер и декодер должны модифицировать дерево одинаково, чтобы все время использовать один и тот же код, который может меняться по ходу процесса.

Итак, вначале кодер строит пустое дерево Хаффмана, т.е. никакому символу коды еще не присвоены. Поэтому первый символ просто записывается в выходной поток в незакодированной форме, что обычно соответствует 8-битному коду *ASCII*. Затем этот символ помещается в дерево, и ему присваивается код, например 0. После этого кодируется следующий символ во входном потоке, и если этот символ встретился впервые, то он также записывается в выходной поток в виде 8-битного *ASCII* символа и помещается в дерево Хаффмана, где ему присваивается определенный код в соответствии с его текущей частотой появления равной 1. По мере того как поступают символы на вход кодера, происходит подсчет числа их появления и их количества и в соответствии с этой информацией выполняется перестройка дерева Хаффмана. Но здесь есть один нюанс: как отличить 8-битный *ASCII* символ от кода переменной длины в момент декодирования последовательности. Чтобы разрешить эту коллизию используют специальный *esc*- (*escape*) символ, который показывает, что за ним следует незакодированный символ. Соответственно, код самого *esc*-символа должен находиться в дереве Хаффмана и меняться каждый раз по мере кодирования информации (перестройки дерева).

## Метод Гаусса

Классический метод решения системы линейных алгебраических уравнений (СЛАУ). Назван в честь немецкого математика Карла Фридриха Гаусса. Это метод последовательного исключения переменных, когда с помощью элементарных преобразований система уравнений приводится к равносильной системе треугольного вида, из которой последовательно, начиная с последних (по номеру), находятся все переменные систем (рисунок 4.23).

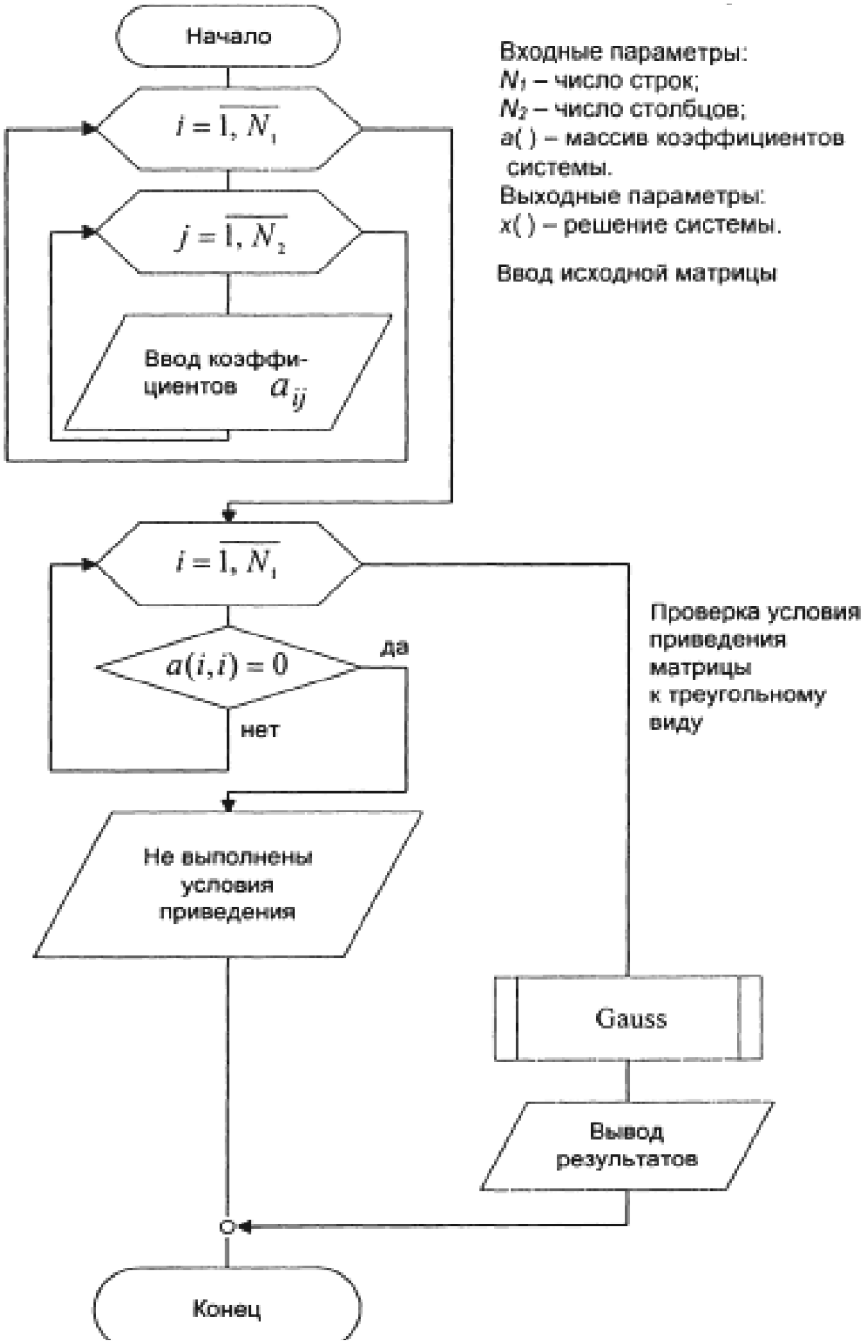



Рисунок 4.23 — Структурная схема алгоритма метода Гаусса

Метод Гаусса включает в себя прямой (приведение расширенной матрицы к ступенчатому виду, то есть получение нулей под главной диагональю) и обратный (получение нулей над главной диагональю расширенной матрицы) ходы. Прямой ход и называется методом Гаусса, обратный — методом Гаусса-Жордана, который отличается от первого только последовательностью исключения переменных.

Метод Гаусса идеально подходит для решения систем, содержащих больше трех линейных уравнений, для решения систем уравнений, которые не являются квадратными (чего не скажешь про метод Крамера и матричный метод). То есть метод Гаусса наиболее универсальный метод для нахождения решения любой системы линейных уравнений, он работает в случае, когда система имеет бесконечно много решений или несовместна (рисунок 4.24, 4.25).

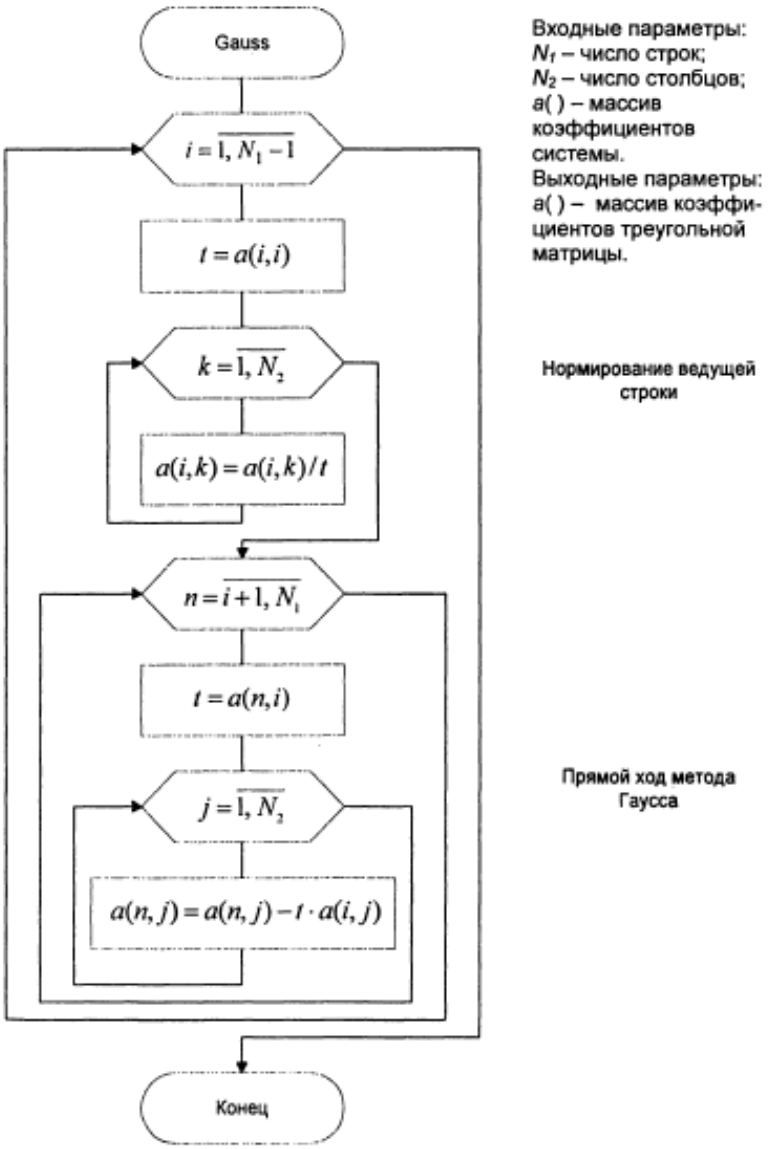
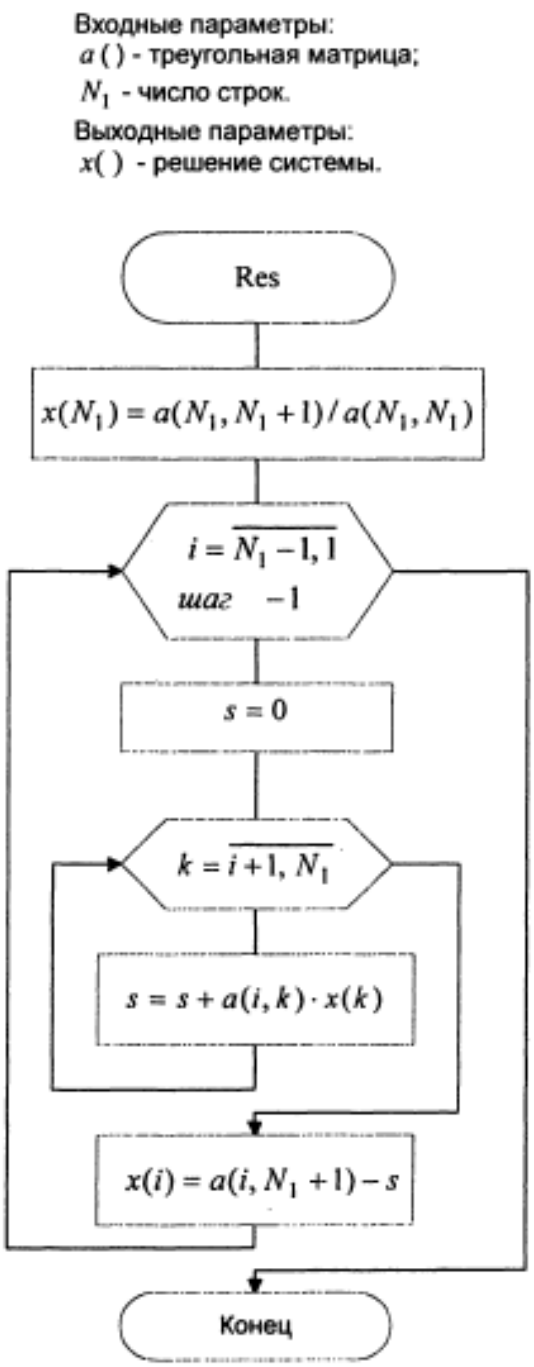



Рисунок 4.24 — Процедура приведения матрицы к треугольному виду




Обратный ход метода Гаусса

Рисунок 4.25 — Процедура решения системы уравнений, приведенной к треугольному виду

## 4.6 Решение обыкновенных дифференциальных уравнений

### Постановка задачи. Типы задач для обыкновенных дифференциальных уравнений (ОДУ)

Известно, что с помощью дифференциальных уравнений можно описать задачи движения системы взаимодействующих материальных точек, химической кинетики, электрических цепей и др. Конкретная прикладная задача может приводить к дифференциальному уравнению любого порядка или к системе уравнений любого порядка. Так как любое ОДУ порядка $p$ $U^p(x) = f\left(x, u, u^1, u^2 \ldots, u^{(p+1)}\right)$ заменой $U^k(x) = y_k(x)$ можно свести к эквивалентной системе из $p$ уравнений первого порядка, представленных в каноническом виде:

$$y_k'(x) = y_{k+1}(x), \quad 0 \le k \le p-2, \tag{4.26}$$

$$y_{p-1}(x) = f\left(x, y_0, y_1, \ldots, y_{p-1}\right), \quad y_0(x) = u(x). \tag{4.27}$$

Покажем такое преобразование на примере уравнения Бесселя:

$$y'' + \frac{y'}{x} + \left(1 - \frac{p^2}{x^2}\right) y = 0. \tag{4.28}$$

Предполагая тождественную замену $y_1(x) = y(x)$, представим систему ОДУ в следующем виде:

$$\begin{cases} y_1'(x) = y_2(x) \\ y_2'(x) = \left(\dfrac{p^2}{x^2} - 1\right) y_1'(x) - \dfrac{y_2(x)}{x}. \end{cases} \tag{4.29}$$

Аналогично произвольную систему дифференциальных уравнений любого порядка можно заменить некоторой эквивалентной системой уравнений первого порядка. Следовательно, алгоритмы численного решения достаточно реализовать для решения системы дифференциальных уравнений первого порядка.

Известно, что система $p$-го порядка имеет множество решений, которые в общем случае зависят от $p$ параметров $\{C_1;C_2;\ldots;C_p\}$. Для определения значений этих параметров, т. е. для выделения единственного решения, необходимо наложить $p$ дополнительных условий на $U_k(x)$. По способу задания условий различают три вида задач, для которых доказано существование и единственность решения.

**Задача Коши.** Задается координата $U_k(x_0)=U_{k0}$ начальной точки интегральной кривой в $(p+1)$-мерном пространстве $(k=1,\ldots,p)$. Решение при этом требуется найти на некотором отрезке $x_0 \le x \le x_{\max}$.

**Краевая задача.** Это задача отыскания частного решения системы ОДУ на отрезке $a \le x \le b$, в которой дополнительные условия налагаются на значения функции $U_k(x)$ более чем в одной точке этого отрезка.

**Задача на собственные значения.** Кроме искомых функций и их производных в уравнение входят дополнительно $m$ неизвестных параметров $\lambda_1,\lambda_2,\ldots,\lambda_m$, которые называются собственными значениями. Для единственности решения на некотором интервале необходимо задать $p + m$ граничных условий.

В большинстве случаев необходимость численного решения систем ОДУ возникает в случае, когда аналитическое решение найти либо невозможно, либо нерационально, а приближенное решение (в виде набора интерполирующих функций) не дает требуемой точности. Численное решение системы ОДУ, в отличие от двух вышеприведенных случаев, никогда не покажет общего решения системы, так как даст только таблицу значений неизвестных функций, удовлетворяющих начальным условиям. По этим значениям можно построить графики данных функций или рассчитать для некоторого $x > x_0$ соотвествующие $U_k(x)$, что обычно и требуется в физических или инженерных задачах. При этом требуется, чтобы соблюдались условия корректно поставленной задачи.

**Метод Эйлера (метод ломаных)**

Рассмотрим задачу Коши для уравнения первого порядка:

$$U(x) = f\left(x, u(x),\ x_0 \le x \ge x_{\max}\right),\ u(x_0) = u_0, \qquad (4.30)$$

В окрестности точки $x_0$ функцию $U(x)$ разложим в ряд Тейлора:

$$U(x) = u(x_0) + u'(x_0)(x - x_0) + \frac{1}{2}u''(x_0)(x - x_0)^2 + \ldots \qquad (4.31)$$

Идея этого и последующих методов основывается на том, что если $f(x,u)$ имеет $q$ непрерывных производных, то в разложении можно удержать члены вплоть до $O\left(h^{q+1}\right)$, при этом стоящие в правой части производные можно найти, дифференцируя требуемое число раз. В случае метода Эйлера ограничимся только двумя членами разложения.

Пусть $h$ — малое приращение аргумента. Тогда выражение (4.31) превратится в

$$U(x_0 + h) = u(x_0) + h \cdot u'(x_0) + O\left(h^2\right). \qquad (4.32)$$

Так как $U'(x_0) = f(x_0, u_0)$, то $U(x_0 + h) \approx u_0 + h \cdot f(x_0, u_0)$.

Теперь приближенное решение в точке $x_1 = x_0 + h$ можно вновь рассматривать как начальное условие, т. е. организуется расчет по следующей рекуррентной формуле:

$$y_{k+1} = y_k + h \cdot f(x_k, y_k), \qquad (4.33)$$

где $y_0 = u_0$, а все $y_k$ — приближенные значения искомой функции.

В методе Эйлера происходит движение не по интегральной кривой, а по касательной к ней. На каждом шаге касательная находится уже для новой интегральной кривой (что и дало название методу — метод ломаных), таким образом ошибка будет возрастать с отдалением $x$ от $x_0$.

При $h \to 0$ приближенное решение сходится к точному равномерно с первым порядком точности, то есть метод дает весьма низкую точность вычислений: погрешность на элементарном шаге $h$

составляет $\frac{1}{2}h^2 y''\left(\frac{1}{2}(x_k + x_{k+1})\right)$, а для всей интегральной кривой порядка $h^1$. При $h = \text{const}$ для оценки апостериорной погрешности может быть применена первая формула Рунге, хотя для работы метода обеспечивать равномерность шага в принципе не требуется.

Метод Эйлера легко обобщается для систем ОДУ. При этом общая схема процесса (4.33) может быть записана так:

$$y_{k+1} = y_k + h \cdot f_i(x_k, y_k, y_k, \ldots, y_k), \tag{4.34}$$

где $i = 1, \ldots, m$ — число уравнений; $k$ — номер предыдущей вычисленной точки.

### Метод Рунге-Кутты II порядка

Увеличение точности решения ОДУ из предыдущей задачи при заданном шаге *h* может быть достигнуто учетом большего количества членов разложения функции в ряд Тейлора. Для метода Рунге-Кутты второго порядка следует взять три первых коэффициента, т. е. обеспечить:

$$U_{k+1} = u_k + hu'_k + \frac{h^2}{2}u''_k + O\left(h^3\right). \tag{4.35}$$

Переходя к приближенному решению $y \approx u$ и заменяя производные в формуле (4.35) конечными разностями, получаем в итоге следующее выражение:

$$y_{k+1} = y_k + h \cdot \left((1-\alpha) f(x_k, y_k) + \alpha \cdot f\left(x_k + \frac{h}{2} y_k + \frac{h}{2\alpha} f(x_k, y_k)\right)\right), \tag{4.36}$$

где $0 \le \alpha \le 1$ — свободный параметр.

Можно показать, что если *f*(*x*,*u*) непрерывна и ограничена вместе со своими вторыми производными, то решение, полученное по данной схеме, равномерно сходится к точному решению с погрешностью порядка $h^2$.

Для параметра $\alpha$ наиболее часто используют значения $\alpha = 1$ и $\alpha = \frac{1}{2}$. В случае $\alpha = 1$

$$y_{k+1} = y_k + h \cdot f\left( x_k + \frac{h}{2} y_k + \frac{h}{2\alpha} f\left(x_k, y_k\right) \right). \qquad (4.37)$$

Графически это уточнение можно интерпретировать так: сначала по схеме ломаных делается шаг $\frac{h}{2}$ и находится значение $y\left(x_k + \frac{h}{2}\right) = y_k + \frac{h}{2} f_k$. В найденной точке определяется наклон касательной к интегральной кривой, который и будет определять приращение функции для целого шага, т. е. отрезок $[AB]$ (рисунок 4.26) будет параллелен касательной, проведенной в точке $\left(x_k + \frac{h}{2}\right), y\left(x_k + \frac{h}{2}\right)$ к интегральной кривой.

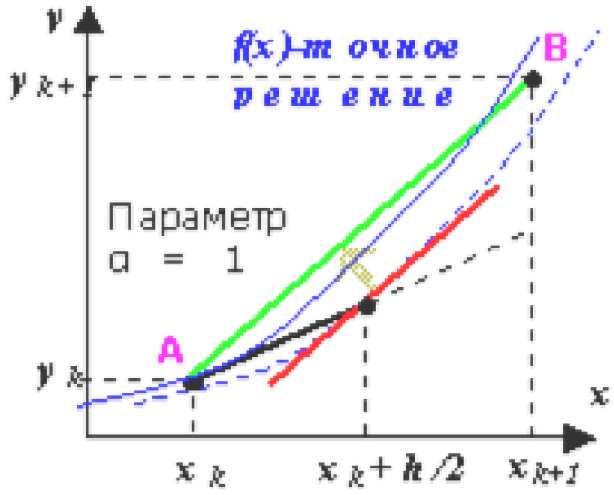



Рисунок 4.26 — График приращения функции

При $\alpha = \frac{1}{2}$ получим:

$$y_{k+1} = y_k + \frac{h}{2} \cdot \left( f\left(x_k, y_k\right) + f\left(x_k + h, y_k + h \cdot f_k\right) \right). \qquad (4.38)$$

Можно представить, что в этом случае по методу Эйлера сначала вычисляется значение функции и наклон касательной к интегральной кривой в точке $x_{k+1}$. Затем находится среднее положение касательной из сравнения соответствующих наклонов в точках $x_k$ и $x_{k+1}$, которое и будет использоваться для расчета точки $y_{k+1}$.

## 4.7 Методы обработки экспериментальных данных

Любому специалисту в своей практической деятельности приходится изучать зависимости между различными параметрами исследуемых объектов, процессов и систем. Например: зависимость числа оборотов двигателя от нагрузки, т. е. $n = f$; зависимость силы резания при обработке детали на металлорежущем станке от глубины резания, т. е. $P = f(t)$, и т. д. Из всех способов задания зависимостей наиболее удобным является аналитический способ задания зависимости в виде функции $n = f, P = f(t), y = f(t)$. Однако на практике специалист чаще всего получает зависимости между исследуемыми параметрами экспериментально. В этом случае ставится натурный эксперимент, изменяются значения параметров на входе системы, измеряются значения параметров на выходе системы. Результаты измерений заносятся в таблицу. Таким образом, в результате проведения натурного эксперимента получаем зависимости между исследуемыми параметрами в виде таблицы, т. е. получаем так называемую табличную функцию.

Далее с этой табличной функцией необходимо вести научно-исследовательские расчеты. Например, необходимо проинтегрировать или продифференцировать табличную функцию и т. д. При такой постановке задачи моделирования нужно заменить табличную функцию аналитической. Для этой цели используются методы аппроксимации и интерполяции.

**Аппроксимация** — это замена исходной функции $f(x)$ функцией $\varphi(x)$ так, чтобы отклонение $f(x)$ от $\varphi(x)$ в заданной области было наименьшим. Функция $\varphi(x)$ называется аппроксимирующей.

Если исходная функция $f(x)$ задана таблично (дискретным набором точек), то аппроксимация называется дискретной. Если исходная функция $f(x)$ задана аналитически (на отрезке), то аппроксимация называется непрерывной, или интегральной.

**Интерполяция** — это замена исходной функции $f(x)$ функцией $\varphi(x)$ так, чтобы $\varphi(x)$ точно проходила через точки исходной

функции $f(x)$. Интерполяция еще называется точечной аппроксимацией. Точки исходной функции $f(x)$ называются узлами интерполяции. Для интерполирующей функции справедливо: $\varphi(x_i) = f_i$, $i = 0,1,2,\ldots,n$.

Графическая задача интерполирования заключается в том, чтобы построить такую интерполирующую функцию, которая бы проходила через все узла интерполирования.

**Экстраполяцией** называется аппроксимация вне заданной области определения исходной функции, т. е $x < x_0$ и $x > x_n$. Найдя интерполяционную функцию, мы можем вычислить ее значения между узлами интерполяции, а также определить значение функции за пределами заданного интервала (провести экстраполяцию). Основной мерой отклонения функции $y(x)$ от функции $f(x)$ при аппроксимации является величина равная сумме квадратов разностей между значениями аппроксимирующей и исходной функции.

Простейшими видами интерполяции является линейная и квадратичная. При линейной интерполяции точки заданной функции соединяются линейными отрезками, и функция $f(x)$ приближается ломаной с вершинами в данных точках. В качестве уравнения интерполяционного многочлена используются уравнения прямой, проходящей через две точки. При квадратичной интерполяции в качестве приближающей функции, соединяющей соседние точки, принимается квадратный трехчлен. Такая интерполяция называется параболической. Распространенным видом интерполяции является интерполяция с использованием кубических сплайн-функций. Сплайн представляет собой модель гибкого тонкого стержня из упругого материала, закрепленного в двух соседних узлах интерполяции с заданными углами наклона $\alpha$ и $\beta$ так, чтобы потенциальная энергия стержня была минимальна.

Интерполяция может выполняться с помощью многочленов Ньютона, Эрмита, Лагранжа и т. д. Наиболее известными методами аппроксимации являются метод наименьших квадратов, метод многочленов Чебышева, рядов Тейлор и т. д. При решении задач аппроксимации часто используются функции регрессии. Регрессия — пред-

ставление совокупности данных некоторой функцией $f(x)$. Задачей регрессии является вычисление параметров функции $f(x)$ таким образом, чтобы функция приближала последовательность исходных точек с наименьшей погрешностью. При этом функция $f(x)$ называется уравнением регрессии. При регрессии не требуется, чтобы функция проходила через все заданные точки, это особенно важно при аппроксимации данных, заведомо содержащих ошибки.

### Интерполяция каноническим многочленом

Интерполяция каноническим многочленом — это определение коэффициентов многочлена *n*-й степени, проходящего через заданную $(n+1)$ точку. Значения в точке определяются по формуле многочлена:

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n, \quad \begin{Vmatrix} a_0 \\ a_1 \\ a_2 \\ \ldots \\ a_3 \end{Vmatrix} = \begin{Vmatrix} 1 & x_0 & x_0^2 & x_0^n \\ 1 & x_1 & x_1^2 & x_1^n \\ 1 & x_2 & x_2^2 & x_2^n \\ \ldots & \ldots & \ldots & \ldots \\ 1 & x_n & x_n^2 & x_n^n \end{Vmatrix}^{-1} \cdot \begin{Vmatrix} y_0 \\ y_1 \\ y_2 \\ \ldots \\ y_n \end{Vmatrix}. \tag{4.39}$$

Заметим, что канонический многочлен — это многочлен *n*-й степени, как и формула Лагранжа. В случае, когда необходимо многократное вычисление многочлена *n*-й степени в различных точках, предпочтительнее использование формулы канонического многочлена.

### Линейная интерполяция

При $n = 1$ канонический многочлен имеет вид:

$$y = a_0 + a_1 x, \begin{Vmatrix} a_0 \\ a_1 \end{Vmatrix} = \begin{Vmatrix} 1 & x_0 \\ 1 & x_1 \end{Vmatrix}^{-1} \cdot \begin{Vmatrix} y_0 \\ y_1 \end{Vmatrix},$$

$$a_0 = \frac{y_0 x_1 - y_1 x_0}{x_1 - x_0}, \quad a_0 = \frac{y_1 - y_0}{x_1 - x_0}. \tag{4.40}$$

### Квадратическая интерполяция

При $n = 2$ канонический многочлен имеет вид:

$$y = a_0 + a_1 x + a_2 x^2, \quad \begin{Vmatrix} a_0 \\ a_1 \\ a_2 \end{Vmatrix} = \begin{Vmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{Vmatrix}^{-1} \cdot \begin{Vmatrix} y_0 \\ y_1 \\ y_2 \end{Vmatrix}. \qquad (4.41)$$

### Кубическая интерполяция

При $n = 3$ канонический многочлен имеет вид:

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3, \quad \begin{Vmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Vmatrix} = \begin{Vmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ 1 & x_3 & x_3^2 & x_3^3 \end{Vmatrix}^{-1} \cdot \begin{Vmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{Vmatrix}. \quad (4.42)$$

### Сплайновая интерполяция

**Сплайном** (*spline*) называли гибкую металлическую линейку — универсальное лекало, которое использовали чертежники для того, чтобы гладко соединить отдельные точки на чертеже, т. е. для графического исполнения интерполяции. Более того, кривая, описывающая деформацию гибкой линейки, зафиксированной в отдельных точках, является сплайном. Другими словами, сплайн используется для определения функции профиля между точками данных.

Сплайны обладают исключительно хорошими аппроксимативными свойствами, универсальностью, обеспечивают простоту реализации вычислительных алгоритмов, полученных на их основе. При этом алгоритмы построения сплайнов совпадают с алгоритмом метода конечных элементов, который является основным промышленным методом прочностного анализа в САПР. И несмотря на то, что в настоящее время традиционной прикладной сферой использования интерполяционных сплайнов стали САПР, потенциальные возможности сплайнов значительно шире, чем просто описание некоторых кривых. В реальном мире большое количество физических процессов по самой своей природе являются сплайнами. То есть сплайн —

не выдуманная математическая абстракция, а во многих случаях он является решением дифференциальных уравнений, описывающих вполне реальные физические процессы (например, в механике — это деформация гибкой пластины или стержня, зафиксированного в отдельных точках; в термодинамике — это теплообмен в стержне, составленном из фрагментов с различной теплопередачей; в химии — диффузия через слои различных веществ; в электричестве — распространение электромагнитных полей через разнородные среды и т. д.).

Рассмотрим более подробно некоторые виды сплайнов.

### Кубический сплайн

Метод интерполяции кубического сплайна создает глобальную посадку. Глобальные методы используют все существующие точки при расчете коэффициентов для всех интервалов интерполяции одновременно. Следовательно, каждая точка данных влияет на весь кубический сплайн. При перемещении одной точки весь сплайн соответственно изменяется, что делает его более неровным, и тем сложнее заставить его принять желаемую форму. Это особенно очевидно для функций с линейными частями или с резкими изменениями в кривой.

Метод интерполяции кубического сплайна работает не так быстро, как интерполяция сплайна Акима, но дает хорошие результаты для значений аппроксимированной функции, а также ее первой и второй производных. Точки данных не должны быть равномерно расположены. Процесс решения часто требует оценки производных определяемых функций. Чем более гладкая производная, тем проще достичь сходимости решения.

### Линейный

Метод линейной интерполяции создает местную посадку путем определения кусочно-непрерывной линейной функции между смежными точками данных.

Метод линейной интерполяции дает более быстрое схождение, чем два других метода. Результирующая функция представляет собой кусочно-непрерывную линейную функцию, имеющую

разрывную производную в заданных точках данных. Вторая производная равняется нулю, за исключением заданных точек данных, где она равна бесконечности.

### *B*-сплайн

В последнее время в вычислительной практике широкое распространение получили *B*-сплайны (от англ. *Bell* — колокол) — сплайны с локальным носителем. Эти сплайны сосредоточены на конечном носителе. Они используются как для интерполяции функций, так и в качестве базисных функций при построении методов типа конечных элементов.

Линия, составленная из *B*-сплайнов, не будет проходить точно через заданные точки. Подобную кривую составляют из дуг полиномов третьей степени, так как такой полином обеспечивает необходимую непрерывность. Построение линии происходит с помощью итерационной процедуры.

## Метод наименьших квадратов при построении аппроксимирующей функции

Аппроксимация опытных данных — это метод, основанный на замене экспериментально полученных данных аналитической функцией, наиболее близко проходящей или совпадающей в узловых точках с исходными значениями (данными, полученными в ходе опыта или эксперимента).

В настоящее время существуют два способа определения аналитической функции:

– с помощью построения интерполяционного многочлена *n*-степени, который проходит непосредственно через все точки заданного массива данных. В таком случае аппроксимирующая функция представляется в виде интерполяционного многочлена в форме Лагранжа или интерполяционного многочлена в форме Ньютона;

– с помощью построения аппроксимирующего многочлена *n*-степени, который проходит в максимальной близости от точек из заданного массива данных. Таким образом, аппроксимирующая функция сглаживает все случайные помехи (или погрешности), которые могут возникать при выполнении эксперимента: измеряемые значения в ходе опыта зависят от случайных факторов, которые ко-

леблются по своим собственным случайным законам (погрешности измерений или приборов, неточность или ошибки опыта). В данном случае аппроксимирующая функция определяется по методу наименьших квадратов.

**Метод наименьших квадратов** (в англоязычной литературе *Ordinary Least Squares*, *OLS*) — математический метод, основанный на определении аппроксимирующей функции, которая строится в максимальной близости от точек из заданного массива экспериментальных данных (рисунок 4.27). Близость исходной и аппроксимирующей функции $F(x)$ определяется числовой мерой, а именно: сумма квадратов отклонений экспериментальных данных от аппроксимирующей кривой $F(x)$ должна быть наименьшей.

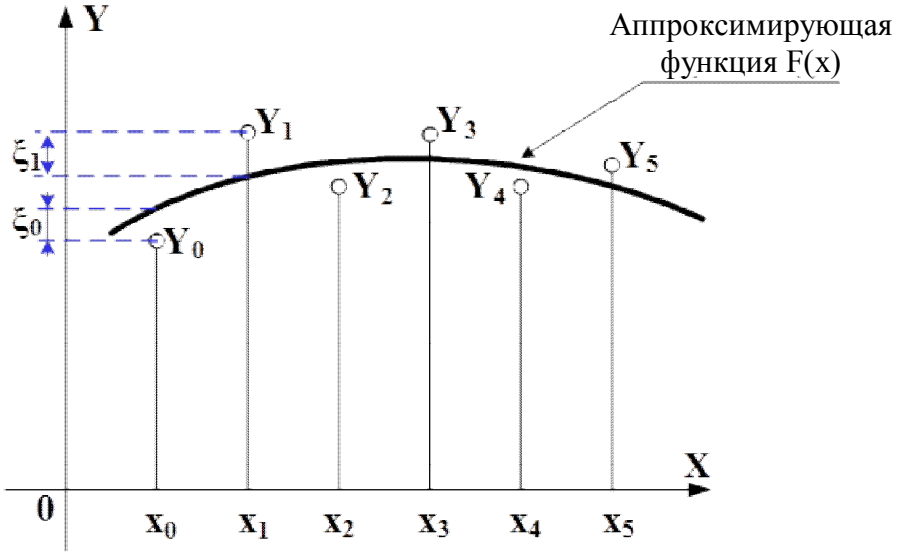



Рисунок 4.27 — Аппроксимирующая кривая,
построенная по методу наименьших квадратов

Метод наименьших квадратов используется:

– для решения переопределенных систем уравнений, когда количество уравнений превышает количество неизвестных;

– для поиска решения в случае обычных (не переопределенных) нелинейных систем уравнений;

– для аппроксимации точечных значений некоторой аппроксимирующей функцией.

Аппроксимирующая функция по методу наименьших квадратов определяется из условия минимума суммы квадратов отклонений

$(\varepsilon_i)$ расчетной аппроксимирующей функции от заданного массива экспериментальных данных. Этот критерий метода наименьших квадратов записывается в виде следующей формулы:

$$\sum_{i=1}^{N} \xi_i^2 = \sum_{i=1}^{N} \left( F(x_i) - y_i \right)^2 \to \min, \qquad (4.43)$$

где $F(x_i)$ — значения расчетной аппроксимирующей функции в узловых точках $x_i$; $y_i$ — заданный массив экспериментальных данных в узловых точках $x_i$.

Квадратичный критерий обладает рядом «хороших» свойств, таких как дифференцируемость, обеспечение единственного решения задачи аппроксимации при полиномиальных аппроксимирующих функциях.

В зависимости от условий задачи аппроксимирующая функция представляет собой многочлен степени *m*:

$$F_m(x) = a_0 + a_1 \cdot x + \ldots + a_{m-1} \cdot x^{m-1} + a_m \cdot x^m. \qquad (4.44)$$

Степень аппроксимирующей функции $m$ не зависит от числа узловых точек, но ее размерность должна быть всегда меньше размерности (количества точек) заданного массива экспериментальных данных

$$1 \le m \ge N - 1. \qquad (4.45)$$

В случае, если степень аппроксимирующей функции $m = 1$, то аппроксимируем табличную функцию прямой линией (линейная регрессия).

В случае, если степень аппроксимирующей функции $m = 2$, то аппроксимируем табличную функцию квадратичной параболой (квадратичная аппроксимация).

В случае, если степень аппроксимирующей функции $m = 3$, то аппроксимируем табличную функцию кубической параболой (кубическая аппроксимация).

В общем случае, когда требуется построить аппроксимирующий многочлен степени $m$ для заданных табличных значений, условие минимума суммы квадратов отклонений по всем узловым точкам переписывается в следующем виде:

$$S = \sum_{i=1}^{N} \xi_i^2 = \sum_{i=1}^{N} \left( a_0 + a_1 \cdot x_i + \ldots + a_{m-1} \cdot x^{m-1} + a_m \cdot x^m - y_i \right)^2 \to \min, \tag{4.46}$$

где $x_i, y_i$ — координаты узловых точек таблицы; $a_j \left( j = 0,\ldots,m \right)$ — неизвестные коэффициенты аппроксимирующего многочлена степени $m$; $N$ — количество заданных табличных значений.

Необходимым условием существования минимума функции является равенство нулю ее частных производных по неизвестным переменным $a_j \left( j = 0,\ldots,m \right)$. В результате получим систему уравнений:

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = 2 \cdot \sum_{i=1}^{N} \left( a_0 + a_1 \cdot x_i + \ldots + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i \right) = 0; \\ \dfrac{\partial S}{\partial a_0} = 2 \cdot \sum_{i=1}^{N} \left( a_0 + a_1 \cdot x_i + \ldots + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i \right) \cdot x_i = 0; \\ \ldots \\ \dfrac{\partial S}{\partial a_0} = 2 \cdot \sum_{i=1}^{N} \left( a_0 + a_1 \cdot x_i + \ldots + a_{m-1} \cdot x_i^{m-1} + a_m \cdot x_i^m - y_i \right) \cdot x_i^m = 0. \end{cases} \tag{4.47}$$

Преобразуем полученную линейную систему уравнений: раскроем скобки и перенесем свободные слагаемые в правую часть выражения:

$$\begin{cases} a_0 \cdot N + a_1 \cdot \sum_{i=1}^{N} x_i + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^{m-1} + a_m \cdot \sum_{i=1}^{N} x_i^m = \sum_{i=1}^{N} y_i; \\ a_0 \cdot \sum_{i=1}^{N} x_i + a_1 \cdot \sum_{i=1}^{N} x_i^{\,2} + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^m + a_m \cdot \sum_{i=1}^{N} x_i^{m+1} = \\ = \sum_{i=1}^{N} y_i \cdot x_i; \\ \ldots \\ a_0 \cdot \sum_{i=1}^{N} x_i^{\,m} + a_1 \cdot \sum_{i=1}^{N} x_i^{\,m+1} + \ldots + a_{m-1} \cdot \sum_{i=1}^{N} x_i^{2m-1} + a_m \cdot \sum_{i=1}^{N} x_i^{2m} = \\ = \sum_{i=1}^{N} y_i \cdot x_i^{\,m}. \end{cases} \tag{4.48}$$

Данная система линейных алгебраических выражений может быть переписана в матричном виде:

$$\begin{vmatrix} N & \sum_{i=1}^{N} x_i & \cdots & \sum_{i=1}^{N} x_i^m \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \cdots & \sum_{i=1}^{N} x_i^{m+1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} x_i^m & \sum_{i=1}^{N} x_i^{m+1} & \cdots & \sum_{i=1}^{N} x_i^{2m} \end{vmatrix} \cdot \begin{vmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{vmatrix} = \begin{vmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} y_i \cdot x_i \\ \vdots \\ \sum_{i=1}^{N} y_i \cdot x_i^m \end{vmatrix}. \quad (4.49)$$

В результате была получена система линейных уравнений размерностью $m+1$, которая состоит из $m+1$ неизвестных. Данная система может быть решена с помощью любого метода решения линейных алгебраических уравнений (например, методом Гаусса). В результате решения будут найдены неизвестные параметры аппроксимирующей функции, обеспечивающие минимальную сумму квадратов отклонений аппроксимирующей функции от исходных данных, т. е. наилучшее возможное квадратичное приближение. Следует помнить, что при изменении даже одного значения исходных данных все коэффициенты изменят свои значения, так как они полностью определяются исходными данными.

# 5 ОПЕРАЦИОННЫЕ СИСТЕМЫ, БАЗЫ ДАННЫХ И ЛОКАЛЬНЫЕ СЕТИ

## 5.1 Введение в архитектуру вычислительных систем и операционные системы

Операционная система представляет собой комплекс системных и служебных программных средств. С одной стороны, она опирается на базовое программное обеспечение компьютера, входящее в его систему *BIOS* (базовая система ввода-вывода); с другой стороны, она сама является опорой для программного обеспечения более высоких уровней — прикладных и большинства служебных приложений. Приложениями операционной системы принято называть программы, предназначенные для работы под управлением данной системы. Основная функция всех операционных систем — посредническая. Она заключается в обеспечении нескольких видов интерфейса:

– интерфейса между пользователем и программно-аппаратными средствами компьютера (интерфейс пользователя);

– интерфейса между программным и аппаратным обеспечением (аппаратно-программный интерфейс);

– интерфейса между разными видами программного обеспечения (программный интерфейс).

Даже для одной аппаратной платформы, например, такой как *IBM PC*, существует несколько операционных систем. Различия между ними рассматривают в двух категориях: внутренние и внешние. Внутренние различия характеризуются методами реализации основных функций. Внешние различия определяются наличием и доступностью приложений данной системы, необходимых для удовлетворения технических требований, предъявляемых к конкретному рабочему месту.

### Обеспечение интерфейса пользователя

#### Режимы работы с компьютером

Все операционные системы способны обеспечивать как пакетный, так и диалоговый режим работы с пользователем. В пакетном режиме операционная система автоматически исполняет заданную

последовательность команд. Суть диалогового режима состоит в том, что операционная система находится в ожидании команды пользователя и, получив ее, приступает к исполнению, а исполнив, возвращает отклик и ждет очередной команды. Диалоговый режим работы основан на использовании прерываний процессора и прерываний *BIOS* (которые в свою очередь также основаны на использовании прерываний процессора). Опираясь на эти аппаратные прерывания, операционная система создает свой комплекс системных прерываний. Способность операционной системы прервать текущую работу и отреагировать на события, вызванные пользователем с помощью управляющих устройств, воспринимается нами как диалоговый режим работы.

**Виды интерфейсов пользователя**

**Интерфейс командной строки.** По реализации интерфейса пользователя различают неграфические и графические операционные системы. Неграфические операционные системы реализуют интерфейс командной строки. Основным устройством управления в данном случае является клавиатура. Управляющие команды вводят в поле командной строки, где их можно и редактировать. Исполнение команды начинается после ее утверждения, например, нажатием клавиши *ENTER*. Для компьютеров платформы *IBM PC* интерфейс командной строки обеспечивается семейством операционных систем под общим названием *MS-DOS*.

**Графический интерфейс.** Графические операционные системы реализуют более сложный тип интерфейса, в котором в качестве органа управления кроме клавиатуры может использоваться мышь или адекватное устройство позиционирования. Работа с графической операционной системой основана на взаимодействии активных и пассивных экранных элементов управления. В качестве активного элемента управления выступает указатель мыши — графический объект, перемещение которого на экране синхронизировано с перемещением мыши. В качестве пассивных элементов управления выступают графические элементы управления приложений (экранные кнопки, значки, переключатели, флажки, раскрывающиеся списки, строки меню и многие другие). Характер взаимодействия между

активными и пассивными элементами управления выбирает сам пользователь. В его распоряжении приемы наведения указателя мыши на элемент управления, щелчки кнопками мыши и другие средства.

### Обеспечение автоматического запуска

Все операционные системы обеспечивают свой автоматический запуск. Для дисковых операционных систем в специальной (системной) области диска создается запись программного кода. Обращение к этому коду выполняют программы, находящиеся в базовой системе ввода-вывода (*BIOS*). Завершая свою работу, они дают команду на загрузку и исполнение содержимого системной области диска. Бездисковые операционные системы характерны для специализированных вычислительных систем, в частности для компьютеризированных устройств автоматического управления. Математическое обеспечение, содержащееся в микросхемах ПЗУ таких компьютеров, можно условно рассматривать как аналог операционной системы. Автоматический запуск такой системы осуществляется аппаратно. При подаче питания процессор обращается к фиксированному физическому адресу ПЗУ (его можно изменять аппаратно с использованием логических микросхем), с которого начинается запись программы инициализации операционной системы.

### Организация файловой системы

Все современные дисковые операционные системы обеспечивают создание файловой системы, предназначенной для хранения данных на дисках и обеспечения доступа к ним. Принцип организации файловой системы — табличный. Поверхность жесткого диска рассматривается как трехмерная матрица, измерениями которой являются номера поверхности, цилиндра и сектора. Под цилиндром понимается совокупность всех дорожек, принадлежащих разным поверхностям и находящихся на равном удалении от оси вращения. Данные о том, в каком месте диска записан тот или иной файл, хранятся в системной области диска. Формат служебных данных определяется конкретной файловой системой. Нарушение целостности служебных сведений приводит к невозможности воспользоваться

данными, записанными на диске. Поэтому к системной области предъявляются особые требования по надежности. Целостность, непротиворечивость и надежность этих данных регулярно контролируется средствами операционной системы. Наименьшей физической единицей хранения данных является сектор.

Размер сектора равен 512 байт. Теоретически возможна самостоятельная адресация для каждого сектора. Но для дисков большого объема такой подход неэффективен, а для некоторых файловых систем и просто невозможен. В связи с этим группы секторов объединяются в кластеры. Кластер является наименьшей единицей адресации при обращении к данным. Размер кластера, в отличие от размера сектора, строго не фиксирован.

Обычно он зависит от емкости диска. Операционные системы *MS-DOS*, *OS*/2, *Windows* и другие используют файловую систему на основе таблиц размещения файлов (*FAT*-таблицы), состоящих из 16-разрядных полей. Такая файловая система называется *FAT*16. Она позволяет разместить в *FAT*-таблицах не более 65 536 записей о местоположении единиц хранения данных. Для дисков объемом от 1 до 2 Гбайт длина кластера составляет 32 Кбайт (64 сектора). Это не вполне рациональный расход рабочего пространства, поскольку любой файл (даже очень маленький) полностью оккупирует весь кластер, которому соответствует только одна адресная запись в таблице размещения файлов. Даже если файл достаточно велик и располагается в нескольких кластерах, все равно в его конце образуется некий остаток, нерационально расходующий целый кластер. Для жестких дисков, объем которых приближается к 2 Гбайт, потери, связанные с неэффективностью этой файловой системы, весьма значительны и могут составлять от 25 % до 40 % полной емкости диска в зависимости от среднего размера хранящихся файлов. С дисками же размером более 2 Гбайт файловая система *FAT*16 вообще работать не может.

Начиная с *Windows* 98 операционные системы семейства *Windows* поддерживают более совершенную версию файловой системы на основе *ivir*-таблиц — *FAT*32 с 32-разрядными полями в таблице размещения файлов. Для дисков размером до 8 Гбайт эта система обеспечивает размер кластера 4 Кбайт (8 секторов). Операционные системы *Windows NTFS* Windows XP способны поддерживать совер-

шенно другую файловую систему — *NTFS*. В ней хранение файлов организовано иначе — служебная информация хранится в главной таблице файлов (*MFT*). В системе *NTFS* размер кластера не зависит от размера диска, и, потенциально, для очень больших дисков эта система должна работать эффективнее, чем *FAT*32. Однако с учетом типичных характеристик современных компьютеров можно говорить о том, что в настоящее время эффективность *FAT*32 и *MF*5 примерно одинакова.

### Обслуживание файловой структуры

Несмотря на то что данные о местоположении файлов хранятся в табличной структуре, пользователю они представляются в виде иерархической структуры — людям так удобнее, а все необходимые преобразования берет на себя операционная система. К функции обслуживания файловой структуры относятся следующие операции, происходящие под управлением операционной системы:

– создание файлов и присвоение им имен;

– создание каталогов (папок) и присвоение им имен;

– переименование файлов и каталогов (папок);

– копирование и перемещение файлов между дисками компьютера и между каталогами (папками) одного диска;

– удаление файлов и каталогов (папок);

– навигация по файловой структуре с целью доступа к заданному файлу, каталогу (папке);

– управление атрибутами файлов.

### Создание и именование файлов

Файл — это именованная последовательность байтов произвольной длины. Поскольку из этого определения вытекает, что файл может иметь нулевую длину, то фактически создание файла состоит в присвоении ему имени и регистрации его в файловой системе — это одна из функций операционной системы. Даже когда нужно создать файл, работая в какой-то прикладной программе, в общем случае для этой операции привлекаются средства операционной системы.

По способам именования файлов различают «короткое» и «длинное» имя. До появления операционной системы *Windows* 95 общепринятым способом именования файлов на компьютерах *IBM PC* было соглашение 83. Согласно этому соглашению, принятому в *MS-DOS,* имя файла состоит из двух частей: собственно имени и расширения имени. На имя файла отводится 8 символов, а на его расширение — 3 символа. Имя от расширения отделяется точкой. Как имя, так и расширение могут включать только алфавитно-цифровые символы латинского алфавита.

Соглашение 83 не является стандартом, и потому в ряде случаев отклонения от правильной формы записи допускаются как операционной системой, так и ее приложениями. Так, например, в большинстве случаев система «не возражает» против использования некоторых специальных символов (восклицательный знак, символ подчеркивания, дефис, тильда и т. п.), а некоторые версии *M5-D05* даже допускают использование в именах файлов символов русского и других алфавитов. Сегодня имена файлов, записанные в соответствии с соглашением 83, считаются «короткими». Основным недостатком «коротких» имен является их низкая содержательность. Далеко не всегда удается выразить несколькими символами характеристику файла, поэтому с появлением операционной системы *Windows* 95 было введено понятие «длинного» имени. Такое имя может содержать до 256 символов. Этого вполне достаточно для создания содержательных имен файлов. «Длинное» имя может содержать любые символы, кроме девяти специальных: «\, /, :, *, ?, «, <, >, |».

В имени разрешается использовать пробелы и несколько точек. Расширением имени считаются все символы, идущие после последней точки, их может быть и больше трех. Введение длинных имен потребовало внесения изменений в организацию файловых систем на основе *FAT*. Появился термин *VFAT*, обозначающий файловую систему на основе *FAT* с поддержкой длинных имен. Файловая система *NTFS* поддерживает длинные имена с самого начала. Наряду с «длинным» именем операционные системы семейства *Windows* создают также и короткое имя файла — оно необходимо для возможности работы с данным файлом на рабочих местах с устаревшими операционными системами.

**Особенности использования длинных имен**. Использование «длинных» имен файлов в операционных системах семейства *Windows* имеет ряд особенностей.

1. Если «длинное» имя файла включает пробелы, то в служебных операциях его надо заключать в кавычки. Рекомендуется не использовать пробелы, а заменять их символами подчеркивания.

2. В корневой папке диска (на верхнем уровне иерархической файловой структуры) нежелательно хранить файлы с длинными именами. В файловых системах на основе *FAT* количество единиц хранения в этой папке ограничено. Чем длиннее имена, тем меньше файлов можно разместить в корневой папке.

3. Кроме ограничения на длину имени файла (256 символов) существует гораздо более жесткое ограничение на длину полного имени файла (в него входит путь доступа к файлу, начиная от вершины иерархической структуры). Полное имя не может быть длиннее 260 символов.

4. В длинных именах файлов разрешается использовать символы любых алфавитов, в том числе и русского, но если документ готовится для передачи, с заказчиком (потребителем документа) необходимо согласовать возможность воспроизведения файлов с такими именами на его оборудовании.

5. Прописные и строчные буквы в именах не различаются операционной системой. Для нее имена Письмо.txt и письмо.txt соответствуют одному и тому же файлу. Однако отображаются символы разных регистров операционной системой исправно. Если для наглядности желательно использовать прописные буквы, это можно делать.

6. Программисты давно научились использовать расширение имени файла для передачи операционной системе, исполняющей программе или пользователю информации о том, к какому типу относятся данные, содержащиеся в файле, и о формате, в котором они записаны. В ранних операционных системах этот факт использовался мало. По существу, операционные системы *MS-DOS* анализировали только расширения *.BAT* (пакетные файлы с командами *MS-DOS*), *.EXE*, *.COM* (исполнимые файлы программ) и *.SYS* (системные файлы конфигурации). В современных операционных системах любое

расширение имени файла может нести информацию для операционной системы. Операционные системы семейства *Windows* имеют средства для регистрации свойств типов файлов по расширению их имени, поэтому во многих случаях выбор расширения имени файла не является частным делом пользователя. Приложения этих систем предлагают выбрать только основную часть имени и указать тип файла, а соответствующее расширение имени приписывают автоматически.

### Создание каталогов (папок)

Каталоги (папки) — важные элементы иерархической структуры, необходимые для обеспечения удобного доступа к файлам, если файлов на носителе слишком много. Файлы объединяются в каталоги по любому общему признаку, заданному их создателем (по типу, по принадлежности, по назначению, по времени создания и т. п.). Каталоги низких уровней вкладываются в каталоги более высоких уровней и являются для них вложенными. Верхним уровнем вложенности иерархической структуры является корневой каталог диска. Все современные операционные системы позволяют создавать каталоги.

Правила присвоения имени каталогу ничем не отличаются от правил присвоения имени файлу, хотя негласно для каталогов не принято задавать расширения имен. Знаем, что в иерархических структурах данных адрес объекта задается маршрутом (путем доступа), ведущим от вершины структуры к объекту. При записи пути доступа к файлу, проходящего через систему вложенных каталогов, все промежуточные каталоги разделяются между собой определенным символом. Во многих операционных системах в качестве такого символа используется «\» (обратная косая черта).

### Каталоги и папки

До появления операционной системы *Windows* 95 при описании иерархической файловой структуры использовался введенный выше термин каталог. С появлением этой системы был введен новый термин — папка. В том что касается обслуживания файловой структуры носителя данных, эти термины равнозначны: каждому каталогу фай-

лов на диске соответствует одноименная папка операционной системы. Основное отличие понятий папка и каталог проявляется не в организации хранения файлов, а в организации хранения объектов иной природы. Так, например, в операционных системах семейства *Windows* существуют специальные папки, представляющие собой удобные логические структуры, которым не соответствует ни один каталог диска.

### Копирование и перемещение файлов

В неграфических операционных системах операции копирования и перемещения файлов выполняются вводом прямой команды в поле командной строки. При этом указывается имя команды, путь доступа к каталогу-источнику и путь доступа к каталогу-приемнику. В графических операционных системах существуют приемы работы с устройством позиционирования, позволяющие выполнять эти команды наглядными методами.

### Удаление файлов и каталогов (папок)

Средства удаления данных не менее важны для операционной системы, чем средства их создания, поскольку ни один носитель данных не обладает бесконечной емкостью. Существует как минимум три режима удаления данных: удаление, уничтожение и стирание, хотя операционные системы обеспечивают только два первых режима (режим надежного стирания данных можно обеспечить лишь специальными программными средствами). Удаление файлов является временным. В операционных системах семейства *Windows* оно организовано с помощью специальной папки, которая называется Корзина. При удалении файлов и папок они перемещаются в Корзину. Эта операция происходит на уровне файловой структуры операционной системы (изменяется только путь доступа к файлам). На уровне файловой системы жесткого диска ничего не происходит — файлы остаются в тех же секторах, где и были записаны. Уничтожение файлов происходит при их удалении в операционной системе *MS-DOS* или при очистке Корзины в операционных системах семейства *Windows*. В этом случае файл полностью удаляется из файловой структуры операционной системы, но на уровне файловой системы

диска с ним происходят лишь незначительные изменения. В таблице размещения файлов он помечается как удаленный, хотя физически остается там же, где и был. Это сделано для минимизации времени операции. При этом открывается возможность записи новых файлов в кластеры, помеченные как «свободные». Для справки укажем, что операция стирания файлов, выполняемая специальными служебными программами, состоит именно в том, чтобы заполнить якобы свободные кластеры, оставшиеся после уничтоженного файла, случайными данными. Поскольку даже после перезаписи данных их еще можно восстановить специальными аппаратными средствами (путем анализа остаточного магнитного гистерезиса), для надежного стирания файлов требуется провести не менее пяти актов случайной перезаписи в одни и те же сектора. Эта операция весьма продолжительна, и, поскольку массовому потребителю она не нужна, ее не включают в стандартные функции операционных систем.

### Навигация по файловой структуре

Навигация по файловой структуре является одной из наиболее используемых функций операционной системы. Удобство этой операции часто воспринимают как удобство работы с операционной системой. В операционных системах, имеющих интерфейс командной строки, навигацию осуществляют путем ввода команд перехода с диска на диск или из каталога в каталог.

В связи с крайним неудобством такой навигации широкое применение нашли специальные служебные программы, называемые файловыми оболочками. Как и операционные системы, файловые оболочки бывают неграфическими и графическими. Наиболее известная неграфическая файловая оболочка для *MS-DOS* — диспетчер файлов *Norton Commander*.

### Управление атрибутами файлов

Кроме имени и расширения имени файла операционная система хранит для каждого файла дату его создания (изменения) и несколько флаговых величин, называемых атрибутами файла. Атрибуты — это дополнительные параметры, определяющие свойства файлов. Операционная система позволяет их контролировать и изменять;

состояние атрибутов учитывается при проведении автоматических операций с файлами. Основных атрибутов четыре:

– «Только для чтения» (*Readonly*);

– «Скрытый» (*Hidden*);

– «Системный» (*System*);

– «Архивный» (*Archive*).

Атрибут «Только для чтения» — ограничивает возможности работы с файлом. Его установка означает, что файл не предназначен для внесения изменений.

Атрибут «Скрытый» — сигнализирует операционной системе о том, что данный файл не следует отображать на экране при проведении файловых операций. Это мера защиты против случайного (умышленного или неумышленного) повреждения файла.

Атрибутом «Системный» — помечаются файлы, обладающие важными функциями для работы самой операционной системы. Его отличительная особенность в том, что средствами операционной системы его изменить нельзя. Как правило, большинство файлов, имеющих установленный атрибут «Системный», имеют также и установленный атрибут «Скрытый».

Атрибут «Архивный» — в прошлом использовался для работы программ резервного копирования. Предполагалось, что любая программа, изменяющая файл, должна автоматически устанавливать этот атрибут, а средство резервного копирования должно его сбрасывать. Таким образом, очередному резервному копированию подлежали только те файлы, у которых этот атрибут был установлен. Современные программы резервного копирования используют другие средства для установления факта изменения файла, и данный атрибут во внимание не принимается, а его изменение вручную средствами операционной системы не имеет практического значения.

## Управление установкой, исполнением и удалением приложений

### Понятие многозадачности

Работа с приложениями составляет наиболее важную часть работы операционной системы. Это очевидно, если вспомнить, что основная функция операционной системы состоит в обеспечении интерфейса приложений с аппаратными и программными средствами вычислительной системы, а также с пользователем. С точки зрения управления исполнением приложений различают однозадачные и многозадачные операционные системы.

Однозадачные операционные системы (например, *MS-DOS*) передают все ресурсы вычислительной системы одному исполняемому приложению и не допускают ни параллельного выполнения другого приложения (полная многозадачность), ни его приостановки и запуска другого приложения (вытесняющая многозадачность). В то же время, параллельно с однозадачными операционными системами возможна работа специальных программ, называемых резидентными. Такие программы не опираются на операционную систему, а непосредственно работают с процессором, используя его систему прерываний.

Большинство современных графических операционных систем — многозадачные. Они управляют распределением ресурсов вычислительной системы между задачами и обеспечивают:

– возможность одновременной или поочередной работы нескольких приложений;

– возможность обмена данными между приложениями;

– возможность совместного использования программных, аппаратных, сетевых и прочих ресурсов вычислительной системы несколькими приложениями.

## Взаимодействие с аппаратным обеспечением

Средства аппаратного обеспечения вычислительной техники отличаются гигантским многообразием. Существуют сотни различных моделей видеоадаптеров, звуковых карт, мониторов, принтеров, сканеров и прочего оборудования. Ни один разработчик программного

обеспечения не в состоянии предусмотреть все варианты взаимодействия своей программы, например, с печатающим устройством. Гибкость аппаратных и программных конфигураций вычислительных систем поддерживается за счет того, что каждый разработчик оборудования прикладывает к нему специальные программные средства управления — драйверы.

Драйверы имеют точки входа для взаимодействия с прикладными программами, а диспетчеризация обращений прикладных программ к драйверам устройств — это одна из функций операционной системы. Строго говоря, выпуская устройство, например модем, его разработчик прикладывает к нему несколько драйверов, предназначенных для основных операционных систем, как-то: *MS-DOS, Windows XP, Linux* и т. п. В операционных системах *MS-DOS* драйверы устройств загружают резидентные программы, напрямую работающие с процессором и другими устройствами материнской платы.

Здесь участие операционной системы сводится лишь к тому, чтобы предоставить пользователю возможность загрузки драйвера, далее он сам перехватывает прерывания, используемые для обращения к устройству, и управляет его взаимодействием с вызывающей программой. Загрузка драйверов устройств может быть ручной или автоматической.

При ручной загрузке после первоначальной загрузки компьютера пользователь сам выдает команды на загрузку драйверов. В автоматическом режиме команды на загрузку и настройку драйверов включаются в состав файлов, автоматически читаемых при загрузке компьютера. В *MS-DOS* такие файлы называются файлами конфигурации, их всего два — это файлы *autoexec.bat* и *config.sys*. В них прежде всего включают команды загрузки драйвера мыши, дисковода *CD-ROM*, звуковой карты, расширенной памяти (оперативная память, лежащая за пределами 1 Мбайт, рассматривается в *MS-DOS* как дополнительное устройство и требует специального драйвера), а также прочих устройств.

В операционных системах семейства *Windows* операционная система берет на себя все функции по установке драйверов устройств и передаче им управления от приложений. Во многих случаях операционная система даже не нуждается в драйверах, полученных

от разработчика устройства, а использует драйверы из собственной базы данных.

Наиболее современные операционные системы позволяют управлять не только установкой и регистрацией программных драйверов устройств, но и процессом аппаратно-логического подключения. Каждое подключенное устройство может использовать до трех аппаратных ресурсов устройств материнской платы: адреса внешних портов процессора, прерывания процессора и каналы прямого доступа к памяти. При некоторых способах подключения устройства к материнской плате (например, через шину PC) есть техническая возможность организовать между ним и материнской платой обратную связь. Это позволяет операционной системе анализировать требования устройств о выделении им ресурсов и гибко реагировать на них, исключая захват одних и тех же ресурсов разными устройствами.

Такой принцип динамического распределения ресурсов операционной системой получил название *plug-andplay*, а устройства, удовлетворяющие этому принципу, называются самоустанавливающимися. Устройства, подключаемые по устаревшим шинам, не являются самоустанавливающимися.

В этом случае операционная система не может выделять им ресурсы динамически, но тем не менее при распределении ресурсов для самоустанавливающихся устройств она учитывает ресурсы, захваченные ими.

### Обслуживание компьютера

Предоставление основных средств обслуживания компьютера — одна из функций операционной системы. Обычно она решается внешним образом — включением в базовый состав операционной системы первоочередных служебных приложений.

### Средства проверки дисков

Надежность работы дисков (особенно жесткого диска) определяет не только надежность работы компьютера в целом, но и безопасность хранения данных, ценность которых может намного превышать стоимость самого компьютера. Поэтому наличие средств для

проверки дисков является обязательным требованием к любой операционной системе. Средства проверки принято рассматривать в двух категориях: средства логической проверки, то есть проверки целостности файловой структуры, и средства физической диагностики поверхности.

Логические ошибки, как правило, устраняются средствами самой операционной системы, а физические дефекты поверхности только локализуются — операционная система принимает во внимание факт повреждения магнитного слоя в определенных секторах и исключает их из активной работы. Возможность возникновения логических ошибок зависит от типа файловой системы.

Например, схема организации работы в системе *NTFS* вообще исключает возникновение внутренних несоответствий в логической структуре, если не принимать во внимание возможность физического сбоя в процессе записи. В системе на основе *FAT* логические ошибки файловой структуры имеют два характерных проявления: это потерянные кластеры или общие кластеры.

Потерянные кластеры образуются в результате неправильного (или аварийного) завершения работы с компьютером. Так, например, ни в одной операционной системе нельзя выключать компьютер, если на нем запущены приложения, осуществляющие обмен информацией с дисками. Кроме того, в операционных системах *Windows* также нельзя выключать компьютер, если не исполнена специальная процедура завершения работы с операционной системой. Механизм образования потерянных кластеров выглядит следующим образом:

– во время работы с файлом приложение манипулирует с кластерами, занимая или освобождая их, и регистрирует сведения об этом в *FAT*-таблице, но не записывает полные сведения о файле в каталог;

– если при завершении работы с приложением происходит сохранение результатов деятельности, оно вносит окончательные изменения в *FAT*-таблицы и регистрирует данные, записанные в кластерах, как файл в каталоге;

– если при завершении работы с приложением файл уничтожается, информация не фиксируется в каталоге, а использованные кластеры освобождаются;

– если компьютер выключается до завершения работы с приложением, кластеры остаются помеченными как «занятые», но ссылки на них в каталоге не создается, так что согласно данным *FAT*-таблицы этим кластерам не соответствует ни один файл.

Ошибка, связанная с потерянными кластерами, легко парируется средствами операционной системы. При этом можно либо полностью освободить данные кластеры, либо превратить их в полноценные файлы, которые можно просмотреть в поисках ценной информации, утраченной во время сбоя.

Ошибка, проявляющаяся как общие кластеры, характеризуется тем, что согласно данным *FAT*-таблиц два или более файлов претендуют на то, что их данные находятся в одном и том же месте диска.

При нормальной работе такой ситуации быть не может, и это свидетельствует об ошибке в *FAT*-таблицах. Причиной появления общих кластеров может стать самопроизвольное изменение данных в *FAT*-таблицах или некорректное восстановление ранее удаленных данных с помощью внесистемных средств. Некорректность может быть обусловлена нарушением порядка операций восстановления данных или неадекватностью средств восстановления данных (например, использованием средств *MS-DOS* для восстановления файлов, записанных средствами *Windows*).

Ошибка, связанная с общими кластерами, парируется повторной записью обоих конфликтующих файлов. Один из них обязательно испорчен и подлежит последующему удалению, но велика вероятность того, что испорчены оба файла. Дополнительно к вышеуказанным логическим ошибкам операционные системы семейства *Windows* определяют логические ошибки, связанные с некорректной записью даты создания файла и с представлением «короткого» имени файла для заданного «длинного» имени. В операционной системе *Windows XP* проверка дисков, содержащих системную или служебную информацию, рассматривается как потенциально опасная операция, способная поставить дальнейшую работу компьютера под угрозу. В этом случае проверка не выполняется немедленно, а назначается на время очередной перезагрузки системы. Такая же проверка системных дисков обычно производится и в случае аварийного отключения или аварийной перезагрузки компьютера.

### Средства управления виртуальной памятью

Ранние операционные системы ограничивали возможность использования приложений по объему необходимой для их работы оперативной памяти. Так, например, без специальных драйверов (менеджеров оперативной памяти) операционные системы *MS-DOS* ограничивали предельный размер исполняемых программ величиной около 640 Кбайт. Современные операционные системы не только обеспечивают непосредственный доступ ко всему полю оперативной памяти, установленной в компьютере, но и позволяют ее расширить за счет создания так называемой виртуальной памяти на жестком диске.

Виртуальная память реализуется в виде так называемого файла подкачки. В случае недостаточности оперативной памяти для работы приложения часть ее временно опорожняется с сохранением образа на жестком диске. В процессе работы приложений происходит многократный обмен между основной установленной оперативной памятью и файлом подкачки.

Поскольку электронные операции в оперативной памяти происходят намного быстрее, чем механические операции взаимодействия с диском, увеличение размера оперативной памяти компьютера всегда благоприятно сказывается на ускорении операций и повышении производительности всей вычислительной системы.

Операционная система не только берет на себя весь необходимый обмен данными между ОЗУ и диском, но и позволяет в определенной степени управлять размером файла подкачки вручную.

### Средства кэширования дисков

Поскольку, как уже было отмечено, взаимодействие процессора с дисками компьютера происходит намного медленнее операций обмена с оперативной памятью, операционная система принимает специальные меры по сохранению части прочитанных с диска данных в оперативной памяти. В случае, если по ходу работы процессору вновь потребуется обратиться к ранее считанным данным или программному коду, он может найти их в специальной области ОЗУ, называемой дисковым кэшем.

В ранних операционных системах функции кэширования диска возлагались на специальное внешнее программное средство, подключаемое через файлы конфигурации. В современных операционных системах эту функцию включают в ядро системы и она работает автоматически, без участия пользователя, хотя определенная возможность настройки размера кэша за ним сохраняется.

### Средства резервного копирования данных

Если на компьютере выполняется практическая работа, объем ценных (а зачастую и уникальных) данных нарастает с каждым днем. Ценность данных, размещенных на компьютере, принято измерять совокупностью затрат, которые может понести владелец в случае их утраты. Важным средством защиты данных является регулярное резервное копирование на внешний носитель. В связи с особой важностью этой задачи операционные системы обычно содержат базовые средства для выполнения резервного копирования.

### Прочие функции операционных систем

Кроме основных (базовых) функций операционные системы могут предоставлять различные дополнительные функции. Конкретный выбор операционной системы определяется совокупностью предоставляемых функций и конкретными требованиями к рабочему месту. Прочие функции операционных систем могут включать:

– возможность поддерживать функционирование локальной компьютерной сети без специального программного обеспечения;

– доступ к основным службам Интернета средствами, интегрированными в состав операционной системы;

– возможность создания системными средствами сервера Интернета, его обслуживание и управление, в том числе дистанционное посредством удаленного соединения;

– наличие средств защиты данных от несанкционированного доступа, просмотра и внесения изменений;

– возможность оформления рабочей среды операционной системы, в том числе и средствами, относящимися к категории мультимедиа;

– возможность обеспечения комфортной поочередной работы различных пользователей на одном персональном компьютере с сохранением персональных настроек рабочей среды каждого из них и ограничением доступа к конфиденциальной информации;

– возможность автоматического исполнения операций по обслуживанию компьютера и операционной системы в соответствии с заданным расписанием или под управлением удаленного сервера;

– возможность работы с компьютером для лиц, имеющих физические недостатки, связанные с органами зрения, слуха и другими.

Кроме всего вышеперечисленного, современные операционные системы могут включать минимальный набор прикладного программного обеспечения, которое можно использовать для исполнения простейших практических задач:

– чтение, редактирование и печать текстовых документов;

– создание и редактирование простейших рисунков;

– выполнение арифметических и математических расчетов;

– ведение дневников и служебных блокнотов;

– создание, передача и прием сообщений электронной почты;

– создание и редактирование факсимильных сообщений;

– воспроизведение и редактирование звукозаписи;

– воспроизведение видеозаписи;

– разработка и воспроизведение комплексных электронных документов, включающих текст, графику, звукозапись и видеозапись.

Этим возможности операционных систем не исчерпываются. По мере развития аппаратных средств вычислительной техники и средств связи функции операционных систем непрерывно расширяются, а средства их исполнения совершенствуются.

## 5.2 Базы данных и системы управления базами данных

База данных — это организованная структура, предназначенная для хранения информации. Внимательный читатель, знающий из первого раздела этого пособия о том, что данные и информация понятия взаимосвязанные, но не тождественные, должен заметить некоторое несоответствие в этом определении. Его причины чисто

исторические. В те годы, когда формировалось понятие баз данных, в них действительно хранились только данные. Однако сегодня большинство систем управления базами данных (СУБД) позволяют размещать в своих структурах не только данные, но и методы (то есть программный код), с помощью которых происходит взаимодействие с потребителем или с другими программно-аппаратными комплексами. Таким образом, мы можем говорить, что в современных базах данных хранятся отнюдь не только данные, но и информация. Это утверждение легко пояснить, если, например, рассмотреть базу данных крупного банка. В ней есть все необходимые сведения о клиентах, об их адресах, кредитной истории, состоянии расчетных счетов, финансовых операциях и т. д. Доступ к этой базе имеется у достаточно большого количества сотрудников банка, но среди них вряд ли найдется такое лицо, которое имеет доступ ко всей базе полностью и при этом способно единолично вносить в нее произвольные изменения. Кроме данных, база содержит методы и средства, позволяющие каждому из сотрудников оперировать только с теми данными, которые входят в его компетенцию. В результате взаимодействия данных, содержащихся в базе, с методами, доступными конкретным сотрудникам, образуется информация, которую они потребляют и на основании которой в пределах собственной компетенции производят ввод и редактирование данных. С понятием базы данных тесно связано понятие системы управления базой данных. Это комплекс программных средств, предназначенных для создания структуры новой базы, наполнения ее содержимым, редактирования содержимого и визуализации информации. Под визуализацией информации базы понимается отбор отображаемых данных в соответствии с заданным критерием, их упорядочение, оформление и последующая выдача на устройство вывода или передача по каналам связи.

### Структура простейшей базы данных

Сразу поясним, что если в базе нет никаких данных (пустая база), то это все равно полноценная база данных. Этот факт имеет методическое значение. Хотя данных в базе и нет, но информация в ней все-таки есть — это структура базы. Она определяет методы занесения данных и хранения их в базе. Простейший «некомпьютерный» вариант базы данных — деловой ежедневник, в котором каж-

дому календарному дню выделено по странице. Даже если в нем не записано ни строки, он не перестает быть ежедневником, поскольку имеет структуру, четко отличающую его от записных книжек, рабочих тетрадей и прочей писчебумажной продукции. Базы данных могут содержать различные объекты, но, забегая вперед, скажем, что основными объектами любой базы данных являются ее таблицы. Простейшая база данных имеет хотя бы одну таблицу (рисунок 5.1). Соответственно, структура простейшей базы данных тождественно равна структуре ее таблицы. Знаем, что структуру двумерной таблицы образуют столбцы и строки. Их аналогами в структуре простейшей базы данных являются поля и записи. Если записей в таблице пока нет, значит, ее структура образована только набором полей. Изменив состав полей базовой таблицы (или их свойства), мы изменяем структуру базы данных и, соответственно, получаем новую базу данных.


Рисунок 5.1 — Простейшая таблица базы данных

## Свойства полей базы данных

Поля базы данных не просто определяют структуру базы — они еще определяют групповые свойства данных, записываемых в ячейки, принадлежащие каждому из полей. Ниже перечислены основные свойства полей таблиц баз данных на примере 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 — то значение, которое вводится в ячейки поля автоматически (средство автоматизации ввода данных);

Условие на значение — ограничение, используемое для проверки правильности ввода данных (средство автоматизации ввода, которое используется, как правило, для данных, имеющих числовой тип, денежный тип или тип даты).

Сообщение об ошибке — текстовое сообщение, которое выдается автоматически при попытке ввода в поле ошибочных данных (проверка ошибочности выполняется автоматически, если задано свойство «Условие на значение»).

Обязательное поле — свойство, определяющее обязательность заполнения данного поля при наполнении базы.

Пустые строки — свойство, разрешающее ввод пустых строковых данных (от свойства «Обязательное поле» отличается тем, что относится не ко всем типам данных, а лишь к некоторым, например к текстовым).

Индексированное поле — если поле обладает этим свойством, все операции, связанные с поиском или сортировкой записей по значению, хранящемуся в данном поле, существенно ускоряются. Кроме того, для индексированных полей можно сделать так, что значения в записях будут проверяться по этому полю на наличие повторов, что позволяет автоматически исключить дублирование данных.

Здесь нужно обратить особое внимание на то, что поскольку в разных полях могут содержаться данные разного типа, то и свойства у полей могут различаться в зависимости от типа данных. Так, например, список вышеуказанных свойств полей относится в основном к полям текстового типа. Поля других типов могут иметь или не

иметь эти свойства, но могут добавлять к ним и свои. Например, для данных, представляющих действительные числа, важным свойством является количество знаков после десятичной запятой. С другой стороны, для полей, используемых для хранения рисунков, звукозаписей, видеоклипов и других объектов *OLE*, большинство приведенных выше свойств не имеет смысла.

### Типы данных

С основными типами данных мы уже знакомы. Так, например, при изучении электронных таблиц *Microsoft Excel* мы видели, что они работают с тремя типами данных: текстами, числами и формулами. Таблицы баз данных, как правило, допускают работу с гораздо большим количеством разных типов данных. Так, например, базы данных *Microsoft Access* работают со следующими типами данных (рисунок 5.2).


Рисунок 5.2 — Таблица с полями некоторых типов

Текстовое поле — тип данных, используемый для хранения обычного неформатированного текста ограниченного размера (до 255 символов).

Поле «Мемо» — специальный тип данных для хранения больших объемов текста (до 65 535 символов). Физически текст не хранится в поле. Он хранится в другом месте базы данных, а в поле хранится указатель на него, но для пользователя такое разделение заметно не всегда.

Числовое поле — тип данных для хранения действительных чисел.

Дата/время — тип данных для хранения календарных дат и текущего времени.

Денежное поле — тип данных для хранения денежных сумм. Теоретически для их записи можно было бы пользоваться и полями числового типа, но для денежных сумм есть некоторые особенности (например, связанные с правилами округления), которые делают более удобным использование специального типа данных, а не настройку числового типа.

Счетчик — специальный тип данных для уникальных (не повторяющихся в поле) натуральных чисел с автоматическим наращиванием. Естественное использование для порядковой нумерации записей.

Логическое поле — 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. Реально, конечно, такие объекты в таблице не хранятся. Как и в случае полей *MEMO*, они хранятся в другом месте внутренней структуры файла базы данных, а в таблице хранятся только указатели на них (иначе работа с таблицами была бы чрезвычайно замедленной).

Гиперссылка — специальное поле для хранения адресов *URL* для *Web*-объектов Интернета. При щелчке на ссылке автоматически происходит запуск браузера и воспроизведение объекта в его окне.

Мастер подстановок — это не специальный тип данных. Это объект, настройкой которого можно автоматизировать ввод данных в поле так, чтобы не вводить их вручную, а выбирать из раскрывающегося списка.

### Безопасность баз данных

Базы данных — это тоже файлы, но работа с ними отличается от работы с файлами других типов, создаваемых прочими приложениями. Выше мы видели, что всю работу по обслуживанию файловой структуры берет на себя операционная система. Для баз данных предъявляются особые требования с точки зрения безопасности, поэтому в них реализован другой подход к сохранению данных. При работе с обычными приложениями для сохранения данных мы выда-

ем соответствующую команду, задаем имя файла и доверяемся операционной системе. Если мы закроем файл, не сохранив его, то вся работа по созданию или редактированию файла пропадет безвозвратно. Базы данных — это особые структуры. Информация, которая в них содержится, очень часто имеет общественную ценность. Нередко с одной и той же базой (например, с базой регистрации автомобилей в ГИБДД) работают тысячи людей по всей стране. От информации, которая содержится в некоторых базах, может зависеть благополучие множества людей. Поэтому целостность содержимого базы не может и не должна зависеть ни от конкретных действий некоего пользователя, забывшего сохранить файл перед выключением компьютера, ни от перебоев в электросети. Проблема безопасности баз данных решается тем, что в СУБД для сохранения информации используется двойной подход. В части операций, как обычно, участвует операционная система компьютера, но некоторые операции сохранения происходят в обход операционной системы. Операции изменения структуры базы данных, создания новых таблиц или иных объектов происходят при сохранении файла базы данных. Об этих операциях СУБД предупреждает пользователя. Это, так сказать, глобальные операции. Их никогда не проводят с базой данных, находящейся в коммерческой эксплуатации, только с ее копией. В этом случае любые сбои в работе вычислительных систем не страшны. С другой стороны, операции по изменению содержания данных, не затрагивающие структуру базы, максимально автоматизированы и выполняются без предупреждения. Если, работая с таблицей данных, мы что-то меняем в составе данных, то изменения сохраняются немедленно и автоматически. Обычно, решив отказаться от изменений в документе, его просто закрывают без сохранения и вновь открывают предыдущую копию. Этот прием работает почти во всех приложениях, но только не в СУБД. Все изменения, вносимые в таблицы базы, сохраняются на диске без нашего ведома, поэтому попытка закрыть базу «без сохранения» ничего не даст, так как все уже сохранено. Таким образом, редактируя таблицы баз данных, создавая новые записи и удаляя старые, мы как бы работаем с жестким диском напрямую, минуя операционную систему.

## Формирование баз данных

### Режимы работы с базами данных

Обычно с базами данных работают две категории исполнителей. Первая категория — проектировщики. Их задача состоит в разработке структуры таблиц базы данных и согласовании ее с заказчиком. Кроме таблиц проектировщики разрабатывают и другие объекты базы данных, предназначенные, с одной стороны, для автоматизации работы с базой, а с другой стороны, для ограничения функциональных возможностей работы с базой (если это необходимо из соображений безопасности). Проектировщики не наполняют базу конкретными данными (заказчик может считать их конфиденциальными и не предоставлять посторонним лицам). Исключение составляет экспериментальное наполнение модельными данными на этапе отладки объектов базы. Вторая категория исполнителей, работающих с базами данных, пользователи. Они получают исходную базу данных от проектировщиков и занимаются ее наполнением и обслуживанием. В общем случае пользователи не имеют средств доступа к управлению структурой базы — только к данным, да и то не ко всем, а к тем, работа с которыми предусмотрена на конкретном рабочем месте. Соответственно, система управления базами данных имеет два режима работы: проектировочный и пользовательский. Первый режим предназначен для создания или изменения структуры базы и ее объектов. Во втором режиме происходит использование уже подготовленных объектов для наполнения базы или получения данных из нее.

### Объекты базы данных

Ранее уже упомянули о том, что кроме таблиц база данных может содержать и другие типы объектов. Привести полную классификацию возможных объектов баз данных затруднительно, поскольку каждая система управления базами данных способна реализовать свои типы объектов. Основные типы объектов мы рассмотрим на примере СУБД *Microsoft Access*. Эта СУБД позволяет создавать и использовать объекты семи различных типов.

**Таблицы.** Как уже говорили, это основные объекты любой базы данных. Во-первых, в таблицах хранятся все данные, имеющиеся

в базе, а во-вторых, таблицы хранят и структуру базы (поля, их типы и свойства).

**Запросы.** Эти объекты служат для извлечения данных из таблиц и предоставления их пользователю в удобном виде. С помощью запросов выполняют такие операции, как отбор данных, их сортировку и фильтрацию. С помощью запросов можно осуществлять преобразование данных по заданному алгоритму, создавать новые таблицы, производить автоматическое наполнение таблиц данными, импортированными из других источников, совершать простейшие вычисления в таблицах и многое другое. Начинающие пользователи не сразу понимают роль запросов, поскольку все те же операции можно делать и с таблицами. Да, действительно, это так, но есть соображения удобства (в первую очередь быстродействия) и соображения безопасности. Из соображений безопасности, чем меньше доступа к базовым таблицам имеют конечные пользователи, тем лучше. Во-первых, снижается риск того, что неумелыми действиями они повредят данные в таблицах. Во-вторых, предоставив разным пользователям разные запросы, можно эффективно разграничить их доступ к данным в строгом соответствии с кругом персональных обязанностей. В банках, например, одни сотрудники имеют доступ к таблицам данных о клиентах, другие — к их расчетным счетам, третьи — к таблицам активов банка. Если и есть специальные службы, имеющие доступ ко всем информационным ресурсам банка (с целью контроля и анализа), то они лишены средств для внесения изменений — все сделано так, чтобы один человек не мог совершить фиктивную операцию независимо от того, какую должность он занимает. В базе данных, которая имеет правильно организованную структуру, для совершения противоправных действий необходим сговор нескольких участников, а такие действия пресекаются не программными, а традиционными средствами обеспечения безопасности. Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе временную результирующую таблицу. Если хотят подчеркнуть факт «временности» этой таблицы, то ее еще называют моментальным снимком. Когда мы работаем с основными таблицами базы, мы физически имеем дело с жестким диском, то есть с очень медленным устройством (напомним, что это

связано с особенностью сохранения данных, описанной выше). Когда же на основании запроса мы получаем результирующую таблицу, то имеем дело с электронной таблицей, не имеющей аналога на жестком диске, это только образ отобранных полей и записей. Разумеется, работа с «образом» происходит гораздо быстрее и эффективнее — это еще одно основание для того, чтобы широко использовать запросы.

Основной принцип состоит в том, что от базовых таблиц никакой упорядоченности не требуется. Все записи в основные таблицы вносятся только в естественном порядке по мере их поступления, то есть в неупорядоченном виде. Если же пользователю надо видеть данные, отсортированные или отфильтрованные по тому или иному принципу, он просто использует соответствующий запрос (рисунок 5.3). Если нужного запроса нет, он обращается к проектировщику и просит его такой запрос сделать и предоставить.

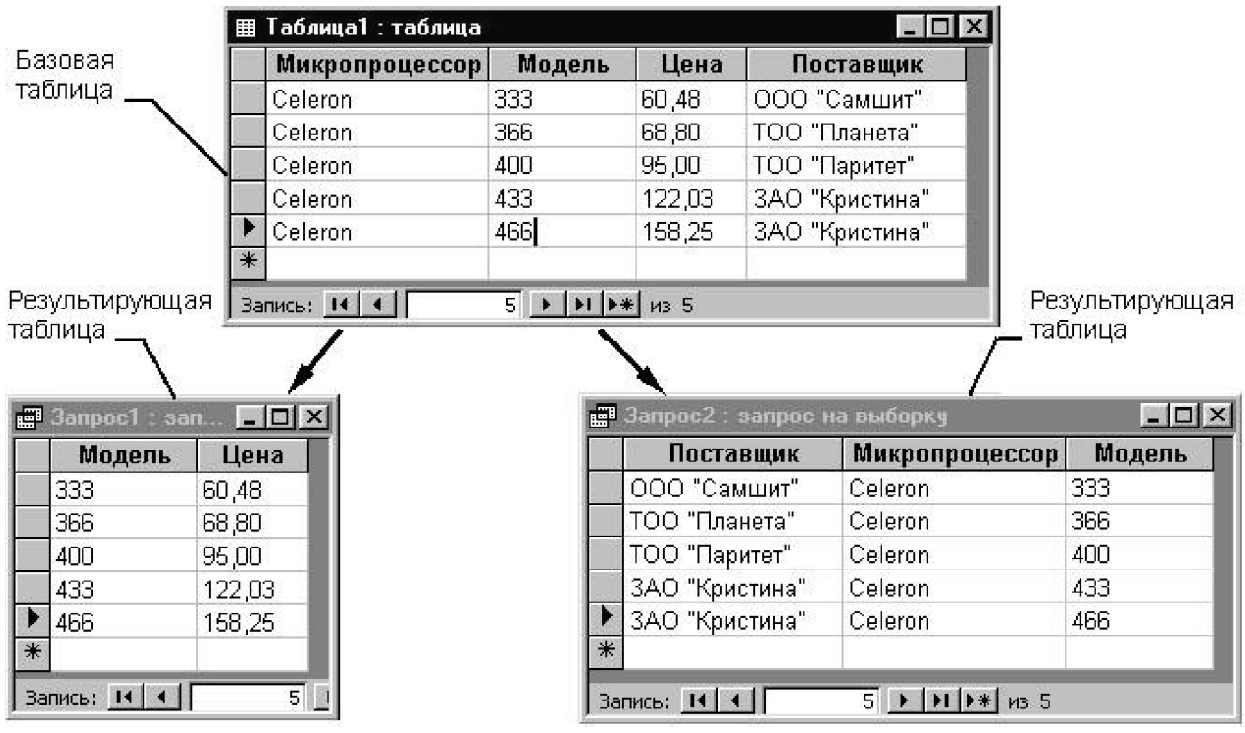



Рисунок 5.3 — Два запроса, сформированные на основе одной таблицы

**Формы**. Если запросы — это специальные средства для отбора и анализа данных, то формы — это средства для ввода данных. Смысл их тот же — предоставить пользователю средства для заполнения только тех полей, которые ему заполнять положено. Одновре-

менно с этим в форме можно разместить специальные элементы управления (счетчики, раскрывающиеся списки, переключатели, флажки и прочие) для автоматизации ввода. Преимущества форм раскрываются особенно наглядно, когда происходит ввод данных с заполненных бланков. В этом случае форму делают графическими средствами так, чтобы она повторяла оформление бланка — это заметно упрощает работу наборщика, снижает его утомление и предотвращает появление печатных ошибок.

С помощью форм данные можно не только вводить, но и отображать. Запросы тоже отображают данные, но делают это в виде результирующей таблицы, не имеющей почти никаких средств оформления. При выводе данных с помощью форм можно применять специальные средства оформления (рисунок 5.4).

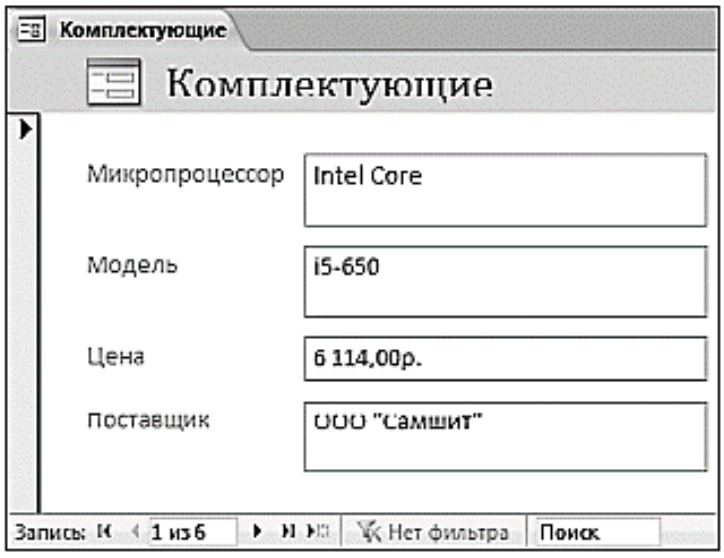



Рисунок 5.4 — Форма для оформленного вывода данных

**Отчеты.** По своим свойствам и структуре отчеты во многом похожи на формы, но предназначены только для вывода данных, причем для вывода не на экран, а на печатающее устройство (принтер). В связи с этим отчеты отличаются тем, что в них приняты специальные меры для группирования выводимых данных и для вывода специальных элементов оформления, характерных для печатных документов (верхний и нижний колонтитулы, номера страниц, служебная информация о времени создания отчета и т. п.) (рисунок 5.5).

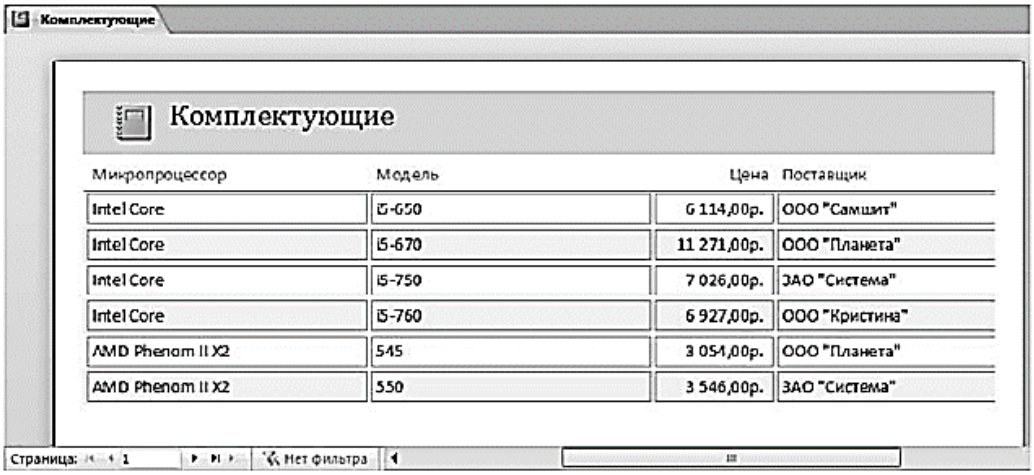



Рисунок 5.5 — Пример простейшего отчета

**Страницы.** Это специальные объекты баз данных, реализованные в последних версиях СУБД *Microsoft Access*, более корректно их называть страницами доступа к данным. Физически это особый объект, выполненный в коде *HTML*, размещаемый на *Web*-странице и передаваемый клиенту вместе с ней. Сам по себе этот объект не является базой данных, но содержит компоненты, через которые осуществляется связь переданной *Web*-страницы с базой данных, остающейся на сервере. Пользуясь этими компонентами, посетитель *Web*-узла может просматривать записи базы в полях страницы доступа (рисунок 5.6). Таким образом, страницы доступа к данным осуществляют интерфейс между клиентом, сервером и базой данных, размещенной на сервере. Эта база данных не обязательно должна быть базой данных *Microsoft Access*. Страницы доступа, созданные средствами *Microsoft Access*, позволяют работать также с базами данных *Microsoft SQL Server*

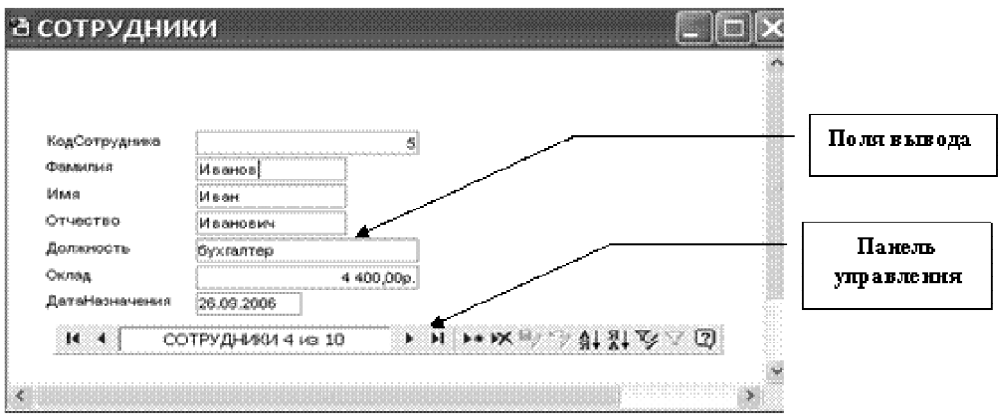



Рисунок 5.6 — Пример простейшей страницы доступа

**Макросы и модули**. Эти категории объектов предназначены как для автоматизации повторяющихся операций при работе с системой управления базами данных, так и для создания новых функций путем программирования. В СУБД *Microsoft Access* макросы состоят из последовательности внутренних команд СУБД и являются одним из средств автоматизации работы с базой. Модули создаются средствами внешнего языка программирования, в данном случае языка *Visual Basic for Applications.* Это одно из средств, с помощью которых разработчик базы может заложить в нее нестандартные функциональные возможности, удовлетворить специфические требования заказчика, повысить быстродействие системы управления, а также уровень ее защищенности.

## Проектирование базы данных

Ранее были рассмотрены основные понятия баз данных и еще не изучались системы управления базами данных. Однако вопрос о проектировании базы уже встает, и это не случайно. Методически правильно начинать работу с карандашом и листом бумаги в руках, не используя компьютер. На данном этапе он просто не нужен. Неоптимальные решения и прямые ошибки, заложенные на этапе проектирования, впоследствии очень трудно устраняются, поэтому этот этап является основополагающим.

**Разработка технического задания.** Техническое задание на проектирование базы данных должен предоставить заказчик. Однако для этого он должен владеть соответствующей терминологией и знать, хотя бы в общих чертах, технические возможности основных систем управления базами данных. К сожалению, на практике такое положение встречается не всегда. Поэтому обычно используют следующие подходы:

– демонстрируют заказчику работу аналогичной базы данных, после чего согласовывают спецификацию отличий;

– если аналога нет, выясняют круг задач и потребностей заказчика, после чего помогают ему подготовить техническое задание.

При подготовке технического задания составляют:

– список выходных данных, которые необходимы заказчику для управления структурой своего предприятия;

– список выходных данных, которые не являются необходимыми для заказчика, но которые он должен предоставлять в другие организации (в вышестоящие структуры, в органы статистического учета, прочие административные и контролирующие организации).

При этом очень важно не ограничиваться взаимодействием с головным подразделением заказчика, а провести обсуждение со всеми службами и подразделениями, которые могут оказаться поставщиками данных в базу или их потребителями. Так, например, при подготовке базы данных для учета абитуриентов и студентов в высшем учебном заведении необходимо не только изучить документооборот ректората и всех деканатов, но и понять, что хотели бы получить от базы данных службы. Следует изучить работу подразделений, распределяющих учебную нагрузку преподавателей, отвечающих за распределение аудиторного фонда, за проживание студентов в общежитии и других. В расчет должны приниматься и такие службы, как библиотека, отдел кадров и прочие. В любой момент может выясниться, например, что администрация библиотеки должна периодически поставлять кому-то отчеты, характеризующие читательскую активность студентов в зависимости от пола, возраста и социального положения. К возможным пожеланиям заказчика следует готовиться на этапе проектирования, до создания базы.

**Разработка схемы данных.** Выяснив основную часть данных, которые заказчик потребляет или поставляет, можно приступать к созданию структуры базы, то есть структуры ее основных таблиц.

Работа начинается с составления генерального списка полей, он может насчитывать десятки и даже сотни позиций.

В соответствии с типом данных, размещаемых в каждом поле, определяют наиболее подходящий тип для каждого поля.

Далее распределяют поля генерального списка по базовым таблицам. На первом этапе распределение производят по функциональному признаку. Цель — обеспечить, чтобы ввод данных в одну таблицу производился, по возможности, в рамках одного подразделения, а еще лучше, на одном рабочем месте. Наметив столько таблиц, сколько подразделений охватывает база данных, приступают к дальнейшему делению таблиц. Критерием необходимости деления является факт множественного повтора данных в соседних записях.

На рисунке 5.7 показана таблица, у которой в поле «Адрес» наблюдается повтор данных. Это явное свидетельство того, что таблицу надо поделить на две взаимосвязанных таблицы.

В каждой из таблиц намечают ключевое поле. В качестве такового выбирают поле, данные в котором повторяться не могут. Например, для таблицы данных о студентах таким полем может служить индивидуальный шифр студента. Для таблицы, в которой содержатся расписания занятий, такого поля можно и не найти, но его можно создать искусственным: комбинированием полей «Время занятия» и «Номер аудитории». Эта комбинация неповторима, так как в одной аудитории в одно и то же время не принято проводить два различных занятия. Если в таблице вообще нет никаких полей, которые можно было бы использовать как ключевые, всегда можно ввести дополнительное поле типа «Счетчик» — оно не может содержать повторяющиеся данные по определению.

С помощью карандаша и бумаги расчерчивают связи между таблицами. На рисунке 5.8 показан пример взаимосвязи между группой таблиц, составляющих одну базу данных. Такой чертеж называется схемой данных.

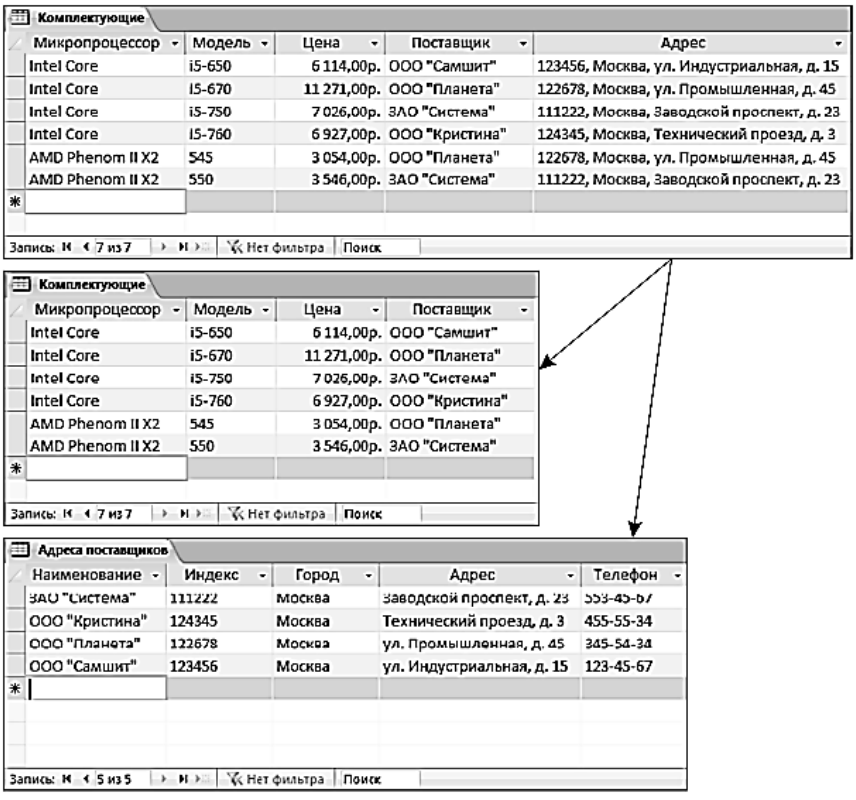



Рисунок 5.7 — Таблица с повтором данных

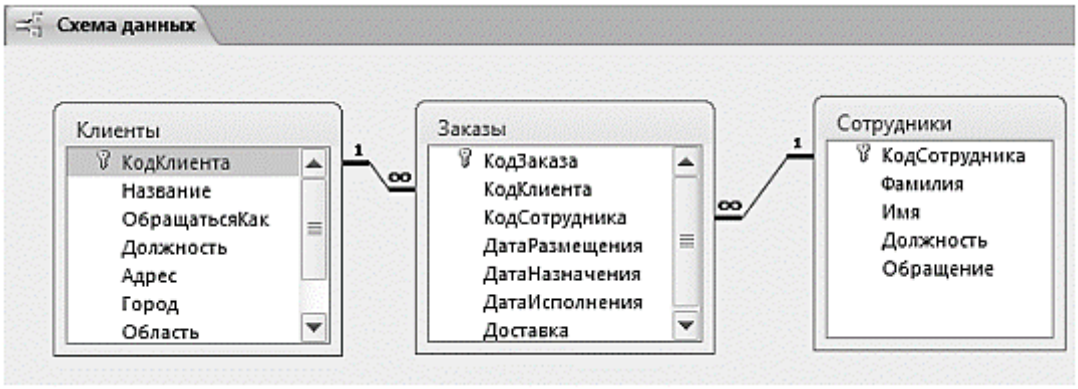



Рисунок 5.8 — Схема связей между таблицами

Существует несколько типов возможных связей между таблицами. Наиболее распространенными являются связи «один ко многим» и «один к одному». Связь между таблицами организуется на основе общего поля, причем в одной из таблиц оно обязательно должно быть ключевым, то есть на стороне «один» должно выступать ключевое поле, содержащее уникальные, неповторяющиеся значения. Значения на стороне «многие» могут повторяться.

Разработкой схемы данных заканчивается «бумажный» этап работы над техническим предложением. Эту схему можно согласовывать с заказчиком, после чего приступать к непосредственному созданию базы данных.

Следует помнить, что по ходу разработки проекта заказчику непременно будут приходить в голову новые идеи. На всех этапах проектирования он стремится охватить единой системой все новые и новые подразделения и службы предприятия. Возможность гибкого исполнения его пожеланий во многом определяется квалификацией разработчика базы данных. Если схема данных составлена правильно, подключать к базе новые таблицы нетрудно. Если структура базы нерациональна, разработчик может испытать серьезные трудности и войти в противоречия с заказчиком. Противоречия исполнителя с заказчиком всегда свидетельствуют о недостаточной квалификации исполнителя. Именно поэтому этап предварительного проектирования базы данных следует считать основным. От его успеха зависит, насколько база данных станет удобной и будут ли с ней работать пользователи. Если отмечается, что пользователи базы «саботируют» ее эксплуатацию и предпочитают работать традиционными методами, это говорит не о низкой квалификации пользователей, а о недос-

таточной квалификации разработчика базы. На этом этапе завершается предварительное проектирование базы данных, и на следующем этапе начинается ее непосредственная разработка. С этого момента следует начать работу с системой управления базами данных.

## 5.3 Локальные и глобальные сети ЭВМ

Основная задача, решаемая при создании компьютерных сетей, — совместимость оборудования по электрическим и механическим характеристикам и совместимость информационного обеспечения (программ и данных) по системе кодирования и формату данных. Решение этой задачи относится к области стандартизации и базируется на так называемой модели *OSI* (модель взаимодействия открытых систем — *Model of Open System Interconnections*). Она создана на основе технических предложений Международного института стандартов *ISO* (*International Standards Organization*). Согласно модели /50/05/архитектуру компьютерных сетей следует рассматривать на разных уровнях (общее число уровней — до семи). Самый верхний уровень — прикладной. На этом уровне пользователь взаимодействует с вычислительной системой. Самый нижний уровень — физический. Он обеспечивает обмен сигналами между устройствами. Обмен данными в системах связи происходит путем их перемещения с верхнего уровня на нижний, затем транспортировки и, наконец, обратным воспроизведением на компьютере клиента в результате перемещения с нижнего уровня на верхний (рисунок 5.9).


Рисунок 5.9 — Простейшая модель обмена данными в компьютерной сети

Для обеспечения необходимой совместимости на каждом из семи возможных уровней архитектуры компьютерной сети действуют специальные стандарты, называемые протоколами. Они определяют характер аппаратного взаимодействия компонентов сети (аппаратные протоколы) и характер взаимодействия программ и данных (программные протоколы). Физически функции поддержки протоколов исполняют аппаратные устройства (интерфейсы) и программные средства (программы поддержки протоколов). Программы, выполняющие поддержку протоколов, также называют протоколами. Так, например, если два компьютера соединены между собой прямым соединением, то на низшем (физическом) уровне протокол их взаимодействия определяют конкретные устройства физического порта (параллельного или последовательного) и механические компоненты (разъемы, кабель и т. п.). На более высоком уровне взаимодействие между компьютерами определяют программные средства, управляющие передачей данных через порты. Для стандартных портов они находятся в базовой системе ввода/вывода (*BIOS*). На самом высоком уровне протокол взаимодействия обеспечивают приложения операционной системы. В соответствии с используемыми протоколами компьютерные сети принято разделять на локальные (*LAN — Local Area Network*) и глобальные (*WAN — Wide Area Network*). Компьютеры локальной сети используют единый комплект протоколов для всех участников. По территориальному признаку локальные сети отличаются компактностью. Они могут объединять компьютеры одного помещения, этажа, здания, группы компактно расположенных сооружений. Глобальные сети имеют, как правило, увеличенные географические размеры. Они могут объединять как отдельные компьютеры, так и отдельные локальные сети, в том числе и использующие различные протоколы. Группы сотрудников, которые работают над одним проектом в рамках локальной сети, называются рабочими группами. В рамках одной локальной сети могут работать несколько рабочих групп. У участников рабочих групп могут быть разные права для доступа к общим ресурсам сети. Совокупность приемов разделения и ограничения прав участников компьютерной сети называется политикой сети. Управление сетевыми политиками (их может быть несколько в одной сети) называется администрированием сети.

Лицо, управляющее организацией работы участников локальной компьютерной сети, называется системным администратором. Создание локальных сетей характерно для отдельных предприятий или отдельных подразделений предприятий. Если предприятие (или отрасль) занимает обширную территорию, то отдельные локальные сети могут объединяться в глобальные сети. В этом случае локальные сети связывают между собой с помощью любых традиционных каналов связи (кабельных, спутниковых, радиорелейных и т. п.). Как мы увидим ниже, при соблюдении специальных условий для этой цели могут быть использованы даже телефонные каналы, хотя они в наименьшей степени удовлетворяют требованиям цифровой связи. Простейшее устройство для соединения между собой двух локальных сетей, использующих одинаковые протоколы, называется мостом. Мост может быть аппаратным (специализированный компьютер) или программным. Цель моста — не выпускать за пределы локальной сети данные, предназначенные для внутреннего потребления. Вне сети такие данные становятся «сетевым мусором», впустую занимающим каналы связи. Для связи между собой нескольких локальных сетей, работающих по разным протоколам, служат специальные средства, называемые шлюзами. Шлюзы могут быть как аппаратными, так и программными. Например, это может быть специальный компьютер (шлюзовый сервер), а может быть и компьютерная программа. В последнем случае компьютер может выполнять не только функцию шлюза, но и какие-то иные функции, типичные для рабочих станций. При подключении локальной сети предприятия к глобальной сети важную роль играет понятие сетевой безопасности. В частности, должен быть ограничен доступ в локальную сеть для посторонних лиц извне, а также ограничен выход за пределы локальной сети для сотрудников предприятия, не имеющих соответствующих прав. Для обеспечения сетевой безопасности между локальной и глобальной сетью устанавливают так называемые брандмауэры. Брандмауэром может быть специальный компьютер или компьютерная программа, препятствующая несанкционированному перемещению данных между сетями.

## Сетевые службы. Основные понятия

### Понятие виртуального соединения

Рассмотрим простой пример взаимодействия двух корреспондентов с помощью обычной почты. Если они регулярно отправляют друг другу письма и, соответственно, получают их, то они могут полагать, что между ними существует соединение на пользовательском (прикладном) уровне. Однако это не совсем так. Такое соединение можно назвать виртуальным. Оно было бы физическим, если бы каждый из корреспондентов лично относил другому письмо и вручал в собственные руки. В реальной жизни он бросает его в почтовый ящик и ждет ответа. Сбором писем из общественных почтовых ящиков и доставкой корреспонденции в личные почтовые ящики занимаются местные почтовые службы. Это другой уровень модели связи, лежащий ниже. Для того чтобы наше письмо достигло адресата в другом городе, должна существовать связь между нашей местной почтовой службой и его местной почтовой службой. Это еще один пример виртуальной связи, поскольку никакой физической связью эти службы не обладают — поступившую почтовую корреспонденцию они только сортируют и передают на уровень федеральной почтовой службы. Федеральная почтовая служба в своей работе опирается на службы очередного уровня, например на почтово-багажную службу железнодорожного ведомства. И только рассмотрев работу этой службы, мы найдем, наконец, признаки физического соединения, например железнодорожный путь, связывающий два города. Это очень простой пример, поскольку в реальности даже доставка обычного письма может затронуть гораздо большее количество служб. Но нам важно обратить внимание на то, что в нашем примере образовалось несколько виртуальных соединений между аналогичными службами, находящимися в пунктах отправки и приема.

Не вступая в прямой контакт, эти службы взаимодействуют между собой. На каком-то уровне письма укладываются в мешки, мешки пломбируют, к ним прикладывают сопроводительные документы, которые где-то в другом городе изучаются и проверяются на аналогичном уровне.

**Модель взаимодействия открытых систем**

Выше мы упомянули о том, что согласно рекомендациям Международного института стандартизации ISO системы компьютерной связи рекомендуется рассматривать на семи разных уровнях (рисунок 5.10).

| Уровень | Аналогия |
|---|---|
| Прикладной уровень | Письмо написано на бумаге. Определено его содержание |
| Уровень представления | Письмо запечатано в конверт. Конверт заполнен. Наклеена марка. Клиентом соблюдены необходимые требования протокола доставки |
| Сеансовый уровень | Письмо опущено в почтовый ящик. Выбрана служба доставки (письмо можно было бы запечатать в бутылку и бросить в реку, но избрана другая служба) |
| Транспортный уровень | Письмо доставлено на почтамт. Оно отделено от писем, с доставкой которых местная почтовая служба справилась бы самостоятельно |
| Сетевой уровень | После сортировки письмо уложено в мешок. Появилась новая единица доставки — мешок |
| Уровень соединения | Мешки писем уложены в вагон. Появилась новая единица доставки — вагон |
| Физический уровень | Вагон прицеплен к локомотиву. Появилась новая единица доставки — состав. За доставку взялось другое ведомство, действующее по другим протоколам |

Рисунок 5.10 — Уровни модели связи

Из рисунка видно, что каждый новый уровень все больше и больше увеличивает функциональность системы связи. Местная почтовая служба работает не только с письмами, но и с бандеролями и посылками. Почтово-багажная служба занимается еще и доставкой грузов. Вагоны перевозят не только почту, но и людей. По рельсам ходят не только почтово-пассажирские поезда, но и грузовые составы и т. д. То есть, чем выше уровень в модели связи, тем больше различных функциональных служб его используют. Возвращаясь к системам компьютерной связи, рассмотрим, как в модели *ISO/OSI* происходит обмен данными между пользователями, находящимися на разных континентах:

– на прикладном уровне с помощью специальных приложений пользователь создает документ (сообщение, рисунок и т. п.);

– на уровне представления операционная система его компьютера фиксирует, где находятся созданные данные (в оперативной памяти, в файле на жестком диске и т. п.), и обеспечивает взаимодействие со следующим уровнем;

– на сеансовом уровне компьютер пользователя взаимодействует с локальной или глобальной сетью. Протоколы этого уровня проверяют права пользователя на «выход в эфир» и передают документ к протоколам транспортного уровня;

– на транспортном уровне документ преобразуется в ту форму, в которой положено передавать данные в используемой сети, например он может нарезаться на небольшие пакеты стандартного размера;

– сетевой уровень определяет маршрут движения данных в сети. Так, например, если на транспортном уровне данные были «нарезаны» на пакеты, то на сетевом уровне каждый пакет должен получить адрес, по которому он должен быть доставлен независимо от прочих пакетов;

– уровень соединения необходим для того, чтобы промодулировать сигналы, циркулирующие на физическом уровне, в соответствии с данными, полученными с сетевого уровня. Например, в компьютере эти функции выполняет сетевая карта или модем;

– реальная передача данных происходит на физическом уровне. Здесь нет ни документов, ни пакетов, ни даже байтов — только биты, то есть элементарные единицы представления данных. Восстановление документа из них произойдет постепенно, при переходе с нижнего на верхний уровень на компьютере клиента.

Средства физического уровня лежат за пределами компьютера. В локальных сетях это оборудование самой сети. При удаленной связи с использованием телефонных модемов — это линии телефонной связи, коммутационное оборудование телефонных станций и т. п.

На компьютере получателя информации происходит обратный процесс преобразования данных от битовых сигналов до документа.

**Особенности виртуальных соединений**

Разные уровни протоколов сервера и клиента не взаимодействуют друг с другом напрямую, но они взаимодействуют через физиче-

ский уровень. Постепенно переходя с верхнего уровня на нижний, данные непрерывно преобразуются, «обрастают» дополнительными данными, которые анализируются протоколами соответствующих уровней на сопредельной стороне. Это и создает эффект виртуального взаимодействия уровней между собой. Однако несмотря на виртуальность, это все-таки соединения, через которые тоже проходят данные. Это очень важный момент с точки зрения компьютерной безопасности. Одновременно с теми запросами на поставку данных, которые клиент направляет серверу, передается масса служебной информации, которая может быть как желательной, так и нежелательной. Например, обязательно передаются данные о текущем адресе клиента, о дате и времени запроса, о версии его операционной системы, о его правах доступа к запрашиваемым данным и прочее. Передается и немало косвенной информации, например о том, по какому адресу он посылал предыдущий запрос. Известны случаи, когда даже передавались идентификационные коды процессоров компьютеров. На использовании виртуальных соединений основаны такие позитивные свойства электронных систем связи, как возможность работать по одному физическому каналу сразу с несколькими серверами. Но на них же основаны и такие негативные средства, как «троянские программы». Троянская программа — разновидность «компьютерного вируса», создающая во время сеансов связи виртуальные соединения для передачи данных о компьютере, на котором установлена. Среди этих данных может быть парольная информация, информация о содержании жесткого диска и т. п. В отличие от обычных компьютерных вирусов троянские программы не производят разрушительных действий на компьютере и потому лучше маскируются.

**Сетевые службы**

На виртуальных соединениях основаны все службы современного Интернета. Так, например, пересылка сообщения от сервера к клиенту может проходить через десятки различных компьютеров. Это совсем не означает, что на каждом компьютере сообщение должно пройти через все уровни, ему достаточно «подняться» до сетевого уровня (определяющего адресацию) при приеме и вновь

«опуститься» до физического уровня при передаче. В данном случае служба передачи сообщений основывается на виртуальном соединении сетевого уровня и соответствующих ему протоколах (рисунок 5.11).

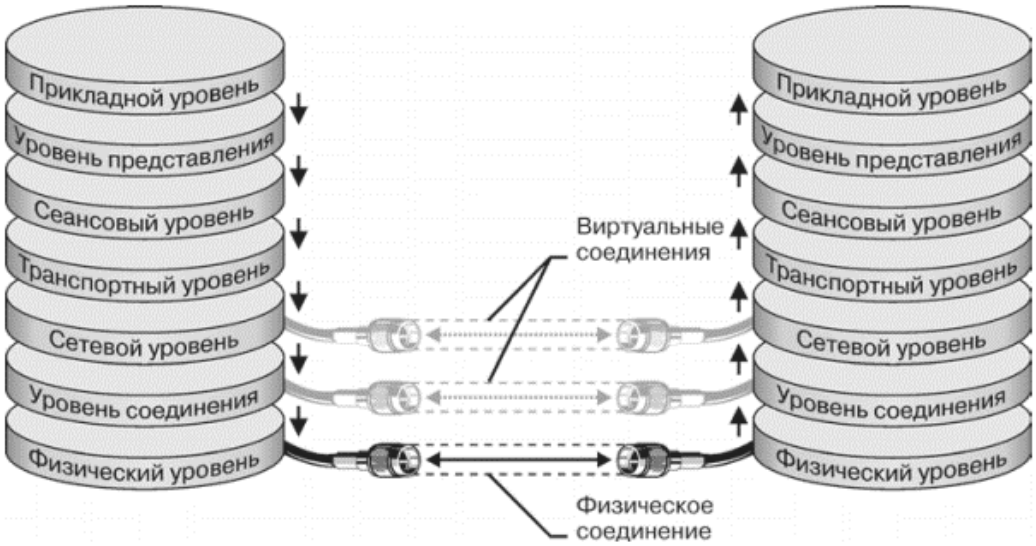



Рисунок 5.11 — Простейшая модель службы передачи сообщений

### Интернет. Основные понятия

В дословном переводе на русский язык Интернет — это межсеть, то есть в узком смысле слова Интернет — это объединение сетей. Однако в 90-е годы XX века у этого слова появился и более широкий смысл: Всемирная компьютерная сеть. Интернет можно рассматривать в физическом смысле как несколько миллионов компьютеров, связанных друг с другом всевозможными линиями связи, однако такой «физический» взгляд на Интернет слишком узок. Лучше рассматривать Интернет как некое информационное пространство. Интернет — это не совокупность прямых соединений между компьютерами. Так, например, если два компьютера, находящиеся на разных континентах, обмениваются данными в Интернете, это совсем не значит, что между ними действует одно прямое или виртуальное соединение. Данные, которые они посылают друг другу, разбиваются на пакеты, и даже в одном сеансе связи разные пакеты одного сообщения могут пройти разными маршрутами. Какими бы маршрутами ни двигались пакеты данных, они все равно достигнут пункта назначения и будут собраны вместе в цельный документ. При этом данные, отправленные позже, могут приходить раньше, но это

не помешает правильно собрать документ, поскольку каждый пакет имеет свою маркировку. Таким образом, Интернет представляет собой как бы «пространство», внутри которого осуществляется непрерывная циркуляция данных. В этом смысле его можно сравнить с теле- и радиоэфиром, хотя есть очевидная разница по крайней мере в том, что в эфире никакая информация храниться не может, а в Интернете она перемещается между компьютерами, составляющими узлы сети, и какое-то время хранится на их жестких дисках.

### Основы функционирования Интернета

В техническом понимании *TCP/IP* — это не один сетевой протокол, а два протокола, лежащих на разных уровнях (это так называемый стек протоколов). Протокол *TCP* — протокол транспортного уровня. Он управляет тем, как происходит передача информации. Протокол *IP* — адресный. Он принадлежит сетевому уровню и определяет, куда происходит передача.

**Протокол *TCP*.** Согласно протоколу *TCP* отправляемые данные «нарезаются» на небольшие пакеты, после чего каждый пакет маркируется таким образом, чтобы в нем были данные, необходимые для правильной сборки документа на компьютере получателя. Для понимания сути протокола *TCP* можно представить игру в шахматы по переписке, когда двое участников разыгрывают одновременно десяток партий. Каждый ход записывается на отдельной открытке с указанием номера партии и номера хода. В этом случае между двумя партнерами через один и тот же почтовый канал работает как бы десяток соединений (по одному на партию). Два компьютера, связанные между собой одним физическим соединением, могут точно так же поддерживать одновременно несколько *TCP*-соединений. Например, два промежуточных сетевых сервера могут одновременно по одной линии связи передавать друг другу в обе стороны множество *TCP*-пакетов от многочисленных клиентов. Когда мы работаем в Интернете, то по одной-единственной телефонной линии можем одновременно принимать документы из Америки, Австралии и Европы. Пакеты каждого из документов поступают порознь, с разделением во времени, и по мере поступления собираются в разные документы.

**Протокол *IP***. Теперь рассмотрим адресный протокол — *IP (Internet Protocol)*. Его суть состоит в том, что у каждого участника Всемирной сети должен быть свой уникальный адрес, *IP*-адрес). Без этого нельзя говорить о точной доставке *TCP*-пакетов на нужное рабочее место. Этот адрес выражается очень просто — четырьмя байтами, например 195.38.46.11.

Структура *IP*-адреса организована так, что каждый компьютер, через который проходит какой-либо *TCP*-пакет, может по этим четырем числам определить, кому из ближайших «соседей» надо переслать пакет, чтобы он оказался «ближе» к получателю. В результате конечного числа перебросок *TCP*-пакет достигает адресата. Выше мы не случайно взяли в кавычки слово «ближе». В данном случае оценивается не географическая «близость». В расчет принимаются условия связи и пропускная способность линии. Два компьютера, находящиеся на разных континентах, но связанные высокопроизводительной линией космической связи, считаются более «близкими» друг к другу, чем два компьютера из соседних поселков, связанные простым телефонным проводом. Решением вопросов, что считать «ближе», а что «дальше», занимаются специальные средства — маршрутизаторы. Роль маршрутизатора в сети может выполнять как специализированный компьютер, так и специальная программа, работающая на узловом сервере сети. Поскольку один байт содержит до 256 различных значений, то теоретически с помощью четырех байтов можно выразить более четырех миллиардов уникальных *IP*-адресов ($256^4$ за вычетом некоторого количества адресов, используемых в качестве служебных). На практике же из-за особенностей адресации к некоторым типам локальных сетей количество возможных адресов составляет порядка двух миллиардов, но и это по современным меркам достаточно большая величина.

## Службы Интернета

Когда говорят о работе в Интернете или об использовании Интернета, то на самом деле речь идет не об Интернете в целом, а только об одной или нескольких из его многочисленных служб. В зависимости от конкретных целей и задач клиенты сети используют те службы, которые им необходимы. В простейшем понимании

служба — это пара программ, взаимодействующих между собой согласно определенным правилам, называемым протоколами. Одна из программ этой пары называется сервером, а вторая — клиентом. Соответственно, когда говорят о работе служб Интернета, речь идет о взаимодействии серверного оборудования и программного обеспечения с клиентским оборудованием и программным обеспечением. Разные службы имеют разные протоколы. Они называются прикладными протоколами. Их соблюдение обеспечивается и поддерживается работой специальных программ. Таким образом, чтобы воспользоваться какой-то из служб Интернета, необходимо установить на компьютере программу, способную работать по протоколу данной службы. Такие программы называют клиентскими или просто клиентами. Так, например, для передачи файлов в Интернете используется специальный прикладной протокол *FTP (File Transfer Protocol)*. Следовательно, чтобы получить из Интернета файл, необходимо:

– иметь на компьютере программу, являющуюся клиентом *FTP* (*FTP*-клиент);

– установить связь с сервером, предоставляющим услуги *FTP* (*FTP*-сервером).

**Терминальный режим**

Исторически одной из ранних является служба удаленного управления компьютером *Telnet.* Подключившись к удаленному компьютеру по протоколу этой службы, можно управлять его работой. Такое управление еще называют консольным, или терминальным. В прошлом эту службу широко использовали для проведения сложных математических расчетов на удаленных вычислительных центрах. Так, например, если для очень сложных вычислений на персональном компьютере требовались недели непрерывной работы, а на удаленной супер-ЭВМ всего несколько минут, то персональный компьютер применяли для удаленного ввода данных в ЭВМ и для приема полученных результатов. В наши дни в связи с быстрым увеличением мощности персональных компьютеров необходимость в подобной услуге сократилась, но тем не менее службы *Telnet* в Интернете продолжают существовать. Часто протоколы *Telnet* применяют для дистанционного управления техническими объектами, например телескопами, видеокамерами, промышленными роботами.

Каждый сервер, предоставляющий *Telnet*-услуги, обычно предлагает свое клиентское приложение. Его надо получить по сети, установить на своем компьютере, подключиться к серверу и работать с удаленным оборудованием.

**Электронная почта (*E-Mail*)**

Эта служба также является одной из наиболее ранних. Ее обеспечением в Интернете занимаются специальные почтовые серверы. Обратите внимание на то, что когда мы говорим о каком-либо сервере, не имеется в виду, что это специальный выделенный компьютер. Здесь и далее под сервером может пониматься программное обеспечение. Таким образом, один узловой компьютер Интернета может выполнять функции нескольких серверов и обеспечивать работу различных служб, оставаясь при этом универсальным компьютером, на котором можно выполнять и другие задачи, характерные для средств вычислительной техники. Почтовые серверы получают сообщения от клиентов и пересылают их по цепочке к почтовым серверам адресатов, где эти сообщения накапливаются. При установлении соединения между адресатом и его почтовым сервером происходит автоматическая передача поступивших сообщений на компьютер адресата. Почтовая служба основана на двух прикладных протоколах: *SMTP* и *POP3*. По первому происходит отправка корреспонденции с компьютера на сервер, а по второму — прием поступивших сообщений. Существует большое разнообразие клиентских почтовых программ. К ним относится, например, программа *Microsoft Outlook Express*, входящая в состав операционной системы *Windows* как стандартная.

**Списки рассылки (*Mail List*)**

Обычная электронная почта предполагает наличие двух партнеров по переписке. Если же партнеров нет, то достаточно большой поток почтовой информации в свой адрес можно обеспечить, подписавшись на списки рассылки. Это специальные тематические серверы, собирающие информацию по определенным темам и переправляющие ее подписчикам в виде сообщений электронной почты. Темами списков рассылки может быть что угодно, например вопросы, связанные с изучением иностранных языков, научно-технические

обзоры, презентация новых программных и аппаратных средств вычислительной техники (рисунок 5.12). Большинство телекомпаний создают списки рассылки на своих узлах, через которые рассылают клиентам аннотированные обзоры телепрограмм. Списки рассылки позволяют эффективно решать вопросы регулярной доставки данных.


Рисунок 5.12 — Список рассылки, посвященный вопросам разработки и производству оптических модуляторов

### Служба телеконференций (*Usenet*)

Служба телеконференций похожа на циркулярную рассылку электронной почты, в ходе которой одно сообщение отправляется не одному корреспонденту, а большой группе (такие группы называются телеконференциями или группами новостей).

Обычное сообщение электронной почты пересылается по узкой цепочке серверов от отправителя к получателю. При этом не предполагается его хранение на промежуточных серверах. Сообщения, направленные на сервер группы новостей, отправляются с него на все серверы, с которыми он связан, если на них данного сообщения еще нет. Затем процесс повторяется. Характер распространения каждого

отдельного сообщения напоминает лесной пожар. На каждом из серверов поступившее сообщение хранится ограниченное время (обычно неделю), и все желающие могут в течение этого времени с ним ознакомиться. Распространяясь во все стороны, менее чем за сутки сообщения охватывают весь земной шар. Далее распространение затухает, поскольку на сервер, который уже имеет данное сообщение, повторная передача производиться не может. Ежедневно в мире создается порядка миллиона сообщений для групп новостей. Выбрать в этом массиве действительно полезную информацию практически невозможно. Поэтому вся система телеконференций разбита на тематические группы. Сегодня в мире насчитывают около 100 000 тематических групп новостей. Они охватывают большинство тем, интересующих массы. Особой популярностью пользуются группы, посвященные вычислительной технике. Основной прием использования групп новостей состоит в том, чтобы задать вопрос, обращаясь ко всему миру, и получить ответ или совет от тех, кто с этим вопросом уже разобрался. При этом важно следить за тем, чтобы содержание вопроса соответствовало теме данной телеконференции. Многие квалифицированные специалисты мира (конструкторы, инженеры, ученые, врачи, педагоги, юристы, писатели, журналисты, программисты и прочие) регулярно просматривают сообщения телеконференций, проходящие в группах, касающихся их сферы деятельности. Такой просмотр называется мониторингом информации. Регулярный мониторинг позволяет специалистам точно знать, что нового происходит в мире по их специальности, какие проблемы беспокоят большие массы людей и на что надо обратить особое внимание в своей работе. В современных промышленных и проектно-конструкторских организациях считается хорошим тоном, если специалисты высшего эшелона периодически (один, два раза в месяц) отвечают через систему телеконференций на типовые вопросы пользователей своей продукции. Так, например, в телеконференциях, посвященных легковым автомобилям, нередко можно найти сообщения от главных конструкторов крупнейших промышленных концернов. При отправке сообщений в телеконференции принято указывать свой адрес электронной почты для обратной связи. В тех случаях, когда есть угроза переполнения электронного «почтового ящика»

корреспонденцией, не относящейся к непосредственной производственной деятельности, вместо основного адреса, используемого для деловой переписки, указывают дополнительный адрес. Как правило, такой адрес арендуют на сервере одной из бесплатных анонимных почтовых служб, например *www.hotmail.com*. Огромный объем сообщений в группах новостей значительно затрудняет их целенаправленный мониторинг, поэтому в некоторых группах производится предварительный «отсев» бесполезной информации (в частности, рекламной), не относящейся к теме конференции. Такие конференции называют модерируемыми. В качестве модератора может выступать не только человек, но и программа, фильтрующая сообщения по определенным ключевым словам. В последнем случае говорят об автоматической модерации. Для работы со службой телеконференций существуют специальные клиентские программы. Так, например, приложение *Microsoft Outlook Express*, указанное выше как почтовый клиент, позволяет работать также и со службой телеконференций. Для начала работы надо настроить программу на взаимодействие с сервером групп новостей, оформить «подписку» на определенные группы и периодически, как и электронную почту, получать все сообщения, проходящие по теме этой группы. В данном случае слово «подписка» не предполагает со стороны клиента никаких обязательств или платежей — это просто указание серверу о том, что сообщения по указанным темам надо доставлять, а по прочим — нет. Отменить подписку или изменить ее состав можно в любой удобный момент.

### Служба *World Wide Web* (*WWW*)

Безусловно, это самая популярная служба современного Интернета. Ее нередко отождествляют с Интернетом, хотя на самом деле это лишь одна из его многочисленных служб. *World Wide Web* — это единое информационное пространство, состоящее из сотен миллионов взаимосвязанных электронных документов, хранящихся на *Web*-серверах. Отдельные документы, составляющие пространство *Web*, называют *Web*-страницами. Количество существующих *Web*-страниц уже измеряется миллиардами, причем энергичный рост объема *World Wide Web* продолжается. Группы тематически объединенных

*Web*-страниц называют *Web*-узлами (альтернативный термин — *Web*-сайт или просто сайт). Один физический *Web*-сервер может содержать достаточно много *Web*-узлов, каждому из которых, как правило, отводится отдельный каталог на жестком диске сервера. От обычных текстовых документов *Web*-страницы отличаются тем, что они оформлены без привязки к конкретному носителю. Например, оформление документа, напечатанного на бумаге, привязано к параметрам печатного листа, который имеет определенную ширину, высоту и размеры полей. Электронные *Web*-документы предназначены для просмотра на экране компьютера, причем заранее неизвестно, на каком. Не известны ни размеры экрана, ни параметры цветового и графического разрешения, не известна даже операционная система, с которой работает компьютер клиента. Поэтому *Web*-документы не могут иметь «жесткого» форматирования. Оформление выполняется непосредственно во время их воспроизведения на компьютере клиента и происходит оно в соответствии с настройками программы, выполняющей просмотр. Программы для просмотра *Web*-страниц называют браузерами. В период «неустойчивости» терминологии применялись также термины браузер или обозреватель, которые еще можно встретить в литературе. Во всех случаях речь идет о некотором средстве просмотра *Web*-документов. Браузер выполняет отображение документа на экране, руководствуясь командами, которые автор документа внедрил в его текст (если автор применяет автоматические средства подготовки *Web*-документов, необходимые команды внедряются автоматически). Такие команды называются тегами. От обычного текста они отличаются тем, что заключены в угловые скобки. Большинство тегов используются парами: открывающий тег и закрывающий. Закрывающий тег начинается с символа «/».

«*CENTRE*» этот текст должен выравниваться по центру экрана.

«*LEFT*» этот текст выравнивается по левой границе экрана.

«*RIGHT*» этот текст выравнивается по правой границе экрана.

Сложные теги имеют кроме ключевого слова дополнительные атрибуты и параметры, детализирующие способ их применения. Правила записи тегов содержатся в спецификации особого языка разметки, близкого к языкам программирования. Он называется язы-

ком разметки гипертекста — *HTML* (*HyperText Markup Language*). Таким образом, *Web*-документ представляет собой обычный текстовый документ, размеченный тегами *HTML*. Такие документы также называют *HTML*-документами или документами в формате *HTML*. При отображении *Web*-документа на экране с помощью браузера теги не показываются, и мы видим только текст, составляющий документ. Однако оформление этого текста (выравнивание, цвет, размер и начертание шрифта и прочее) выполняется в соответствии с тем, какие теги имплантированы в текст документа.

Существуют специальные теги для внедрения графических и мультимедийных объектов (звук, музыка, видеоклипы). Встретив такой тег, браузер делает запрос к серверу на доставку файла, связанного с тегом, и воспроизводит его в соответствии с заданными атрибутами и параметрами тега. В последние годы в *Web*-документах находят широкое применение так называемые активные компоненты. Это тоже объекты, но они содержат не только текстовые, графические и мультимедийные данные, но и программный код, то есть могут не просто отображаться на компьютере клиента, но и выполнять на нем работу по заложенной в них программе. Для того чтобы активные компоненты не могли выполнить на чужом компьютере разрушительные операции (что характерно для «компьютерных вирусов»), они исполняются только под контролем со стороны браузера. Браузер не должен допустить исполнения команд, несущих потенциальную угрозу: например, он пресекает попытки осуществить операции с жестким диском. Возможность внедрения в текст графических и других объектов, реализуемая с помощью тегов *HTML*, является одной из самых эффектных с точки зрения оформления *Web*-страниц, но не самой важной с точки зрения самой идеи *World Wide Web*. Наиболее важной чертой *Web*-страниц, реализуемой с помощью тегов *HTML*, являются гипертекстовые ссылки. С любым фрагментом текста или, например, с рисунком с помощью тегов можно связать иной *Web*-документов, то есть установить гиперссылку. В этом случае при щелчке левой кнопкой мыши на тексте или рисунке, являющемся гиперссылкой, отправляется запрос на доставку нового документа. Этот документ в свою очередь тоже может иметь гиперссылки на другие документы. Тем самым совокупность огромного

числа гипертекстовых электронных документов, хранящихся на серверах *WWW*, образует своеобразное гиперпространство документов, между которыми возможно перемещение. Произвольное перемещение между документами в *Web*-пространстве называют *Web*-серфингом (выполняется с целью ознакомительного просмотра). Целенаправленное перемещение между *Web*-документами называют *Web*-навигацией (выполняется с целью поиска нужной информации). Гипертекстовая связь между сотнями миллионов документов, хранящихся на физических серверах Интернета, является основой существования логического пространства *World Wide Web*. Однако такая связь не могла бы существовать, если бы каждый документ в этом пространстве не обладал своим уникальным адресом. Выше мы говорили, что каждый файл одного локального компьютера обладает уникальным полным именем, в которое входит собственное имя файла (включая расширение имени) и путь доступа к файлу, начиная от имени устройства, на котором он хранится. Теперь мы можем расширить представление об уникальном имени файла и развить его до Всемирной сети. Адрес любого файла во всемирном масштабе определяется унифицированным указателем ресурса — *URL*. Адрес *URL* состоит из трех частей:

– указание службы, которая осуществляет доступ к данному ресурсу (обычно обозначается именем прикладного протокола, соответствующего данной службе). Так, например, для службы *WWW* прикладным является протокол *HTTP (HyperText Transfer Protocol* — протокол передачи гипертекста). После имени протокола ставится двоеточие (:) и два знака «/» (косая черта): *http://...*.

– указание доменного имени компьютера (сервера), на котором хранится данный ресурс: *http://www.abcde.com...*

– указания полного пути доступа к файлу на данном компьютере. В качестве разделителя используется символ «/» (косая черта): *http://www.abcde.com/Files/Mew/abcdefg,zlp*.

При записи *URL*-адреса важно точно соблюдать регистр символов. В отличие от правил работы в *MS-DOS* и *Windows* в Интернете строчные и прописные символы в именах файлов и каталогов считаются разными. Именно в форме *URL* и связывают адрес ресурса с гипертекстовыми ссылками на *Web*-страницах. При щелчке на ги-

перссылке браузер посылает запрос для поиска и доставки ресурса, указанного в ссылке. Если по каким-то причинам он не найден, выдается сообщение о том, что ресурс недоступен (возможно, что сервер временно отключен или изменился адрес ресурса).

**Служба имен доменов (*DNS*)**

Когда ранее говорили о протоколах Интернета, то сказали, что адрес любого компьютера или любой локальной сети в Интернете может быть выражен четырьмя байтами, например так:

195.28.132.97

А только что заявили, что каждый компьютер имеет уникальное доменное имя, например такое:

*www.abcdef.com*

Это просто две разные формы записи адреса одного и того же сетевого компьютера. Человеку неудобно работать с числовым представлением *IP*-адреса, зато доменное имя запоминается легко, особенно если учесть, что, как правило, это имя имеет содержание. Например, *Web-сервер* компании *Microsoft* имеет имя *www.microsoft.com*, а *Web*-сервер компании «Космос ТВ» имеет имя *www.kosmostv.ru*. Нетрудно «реконструировать» и имена для других компаний.

С другой стороны, автоматическая работа серверов сети организована с использованием четырехзначного числового адреса. Благодаря ему промежуточные серверы могут осуществлять передачу запросов и ответов в нужном направлении, не зная, где конкретно находятся отправитель и получатель. Поэтому необходим перевод доменных имен в связанные с ними *IP*-адреса. Этим и занимаются серверы службы имен доменов *DNS*. Наш запрос на получение одной из страниц сервера *www.abcde.com* сначала обрабатывается сервером *DNS* и далее он направляется по *IP*-адресу, а не по доменному имени.

**Служба передачи файлов (*FTP*)**

Прием и передача файлов составляют значительный процент от прочих Интернет-услуг. Необходимость в передаче файлов возникает, например, при приеме файлов программ, при пересылке крупных документов (например, книг), а также при передаче архивных фай-

лов, в которых запакованы большие объемы информации. Служба *FTP* имеет свои серверы в мировой сети, на которых хранятся архивы данных. Со стороны клиента для работы с серверами *FTP* может быть установлено специальное программное обеспечение, хотя в большинстве случаев браузеры *WWW* обладают встроенными возможностями для работы и по протоколу *FTP*.

Протокол *FTP* работает одновременно с двумя *TCP*-соединениями между сервером и клиентом. По одному соединению идет передача данных, а второе соединение используется как управляющее. Протокол *FTP* предоставляет серверу средства для идентификации обратившегося клиента. Этим часто пользуются коммерческие серверы и серверы ограниченного доступа, поставляющие информацию только зарегистрированным клиентам, они выдают запрос на ввод имени пользователя и связанного с ним пароля. Однако существуют и десятки тысяч *FTP*-серверов с анонимным доступом для всех желающих. В этом случае в качестве имени пользователя надо ввести слово: anonymous, а в качестве пароля задать адрес электронной почты. В большинстве случаев программы-клиенты *FTP* делают это автоматически.

***IRC***

Служба *IRC (Internet Relay Chat)* предназначена для прямого общения нескольких человек в режиме реального времени. Иногда службу *IRC* называют чат-конференциями или просто чатом. В отличие от системы телеконференций, в которой общение между участниками обсуждения темы открыто всему миру, в системе *IRC* общение происходит только в пределах одного канала, в работе которого принимают участие обычно лишь несколько человек. Каждый пользователь может создать собственный канал и пригласить в него участников «беседы» или присоединиться к одному из открытых в данный момент каналов. Существует несколько популярных клиентских программ для работы с серверами и сетями, поддерживающими сервис *IRC*.

## 5.4 Компьютерные вирусы. Понятие о компьютерной безопасности

В вычислительной технике понятие безопасности является весьма широким. Оно подразумевает и надежность работы компьютера, и сохранность ценных данных, и защиту информации от внесения в нее изменений неуполномоченными лицами, и сохранение тайны переписки при электронной связи. Разумеется, во всех цивилизованных странах на страже безопасности граждан стоят законы, но в сфере вычислительной техники правоприменительная практика пока развита недостаточно, а законотворческий процесс не успевает за развитием технологий, поэтому надежность работы компьютерных систем во многом опирается на меры самозащиты.

### Компьютерные вирусы

Компьютерный вирус — это программный код, встроенный в другую программу, или в документ, или в определенные области носителя данных и предназначенный для выполнения несанкционированных действий на несущем компьютере. Основными типами компьютерных вирусов являются:

– программные вирусы;

– загрузочные вирусы;

– макровирусы.

К компьютерным вирусам примыкают и так называемые троянские кони (троянские программы, троянцы).

### Программные вирусы

Программные вирусы — это блоки программного кода, целенаправленно внедренные внутрь других прикладных программ. При запуске программы, несущей вирус, происходит запуск имплантированного в нее вирусного кода. Работа этого кода вызывает скрытые от пользователя изменения в файловой системе жестких дисков и/или в содержании других программ. Так, например, вирусный код может воспроизводить себя в теле других программ — этот процесс называется размножением. По прошествии определенного времени, создав достаточное количество копий, программный вирус может

перейти к разрушительным действиям — нарушению работы программ и операционной системы, удалению информации, хранящейся на жестком диске. Этот процесс называется вирусной атакой. Самые разрушительные вирусы могут инициировать форматирование жестких дисков. Поскольку форматирование диска — достаточно продолжительный процесс, который не должен пройти незамеченным со стороны пользователя, во многих случаях программные вирусы ограничиваются уничтожением данных только в системных секторах жесткого диска, что эквивалентно потере таблиц файловой структуры. В этом случае данные на жестком диске остаются нетронутыми, но воспользоваться ими без применения специальных средств нельзя, поскольку неизвестно, какие сектора диска каким файлам принадлежит. Теоретически восстановить данные в этом случае можно, но трудоемкость этих работ исключительно высока. Считается, что никакой вирус не в состоянии вывести из строя аппаратное обеспечение компьютера. Однако бывают случаи, когда аппаратное и программное обеспечение настолько взаимосвязаны, что программные повреждения приходится устранять заменой аппаратных средств. Так, например, в большинстве современных материнских плат базовая система ввода-вывода (*BIOS*) хранится в перезаписываемых постоянных запоминающих устройствах (так называемая флэш-память). Возможность перезаписи информации в микросхеме флэш-памяти используют некоторые программные вирусы для уничтожения данных *BIOS*. В этом случае для восстановления работоспособности компьютера требуется либо замена микросхемы, хранящей *BIOS*, либо ее перепрограммирование на специальных устройствах, называемых программаторами.

Программные вирусы поступают на компьютер при запуске непроверенных программ, полученных на внешнем носителе (гибкий диск, компакт-диск и т. п.) или принятых из Интернета. Особое внимание следует обратить на слова при запуске. При обычном копировании зараженных файлов заражение компьютера произойти не может. В связи с этим все данные, принятые из Интернета, должны проходить обязательную проверку на безопасность, а если получены незатребованные данные из незнакомого источника, их следует уничтожать, не рассматривая. Обычный прием распространения

«троянских» программ — приложение к электронному письму с «рекомендацией» извлечь и запустить якобы полезную программу.

### Загрузочные вирусы

От программных вирусов загрузочные вирусы отличаются методом распространения. Они поражают не программные файлы, а определенные системные области магнитных носителей (гибких и жестких дисков). Кроме того, на включенном компьютере они могут временно располагаться в оперативной памяти. Обычно заражение происходит при попытке загрузки компьютера с магнитного носителя, системная область которого содержит загрузочный вирус. Так, например, при попытке загрузить компьютер с гибкого диска происходит сначала проникновение вируса в оперативную память, а затем в загрузочный сектор жестких дисков. Далее этот компьютер сам становится источником распространения загрузочного вируса.

### Макровирусы

Эта особая разновидность вирусов поражает документы, выполненные в некоторых прикладных программах, имеющих средства для исполнения так называемых макрокоманд. В частности, к таким документам относятся документы текстового процессора *Microsoft Word* (они имеют расширение *.DOC*). Заражение происходит при открытии файла документа в окне программы, если в ней не отключена возможность исполнения макрокоманд. Как и для других типов вирусов, результат атаки может быть как относительно безобидным, так и разрушительным.

### Методы защиты от компьютерных вирусов

Существуют три рубежа защиты от компьютерных вирусов:

– предотвращение поступления вирусов;

– предотвращение вирусной атаки, если вирус все-таки поступил на компьютер;

– предотвращение разрушительных последствий, если атака все-таки произошла.

Существуют три метода реализации защиты:

– программные методы защиты;

– аппаратные методы защиты;

– организационные методы защиты.

В вопросе защиты ценных данных часто используют бытовой подход: «болезнь лучше предотвратить, чем лечить». К сожалению, именно он и вызывает наиболее разрушительные последствия. Создав бастионы на пути проникновения вирусов в компьютер, нельзя положиться на их прочность и остаться неготовым к действиям после разрушительной атаки. К тому же вирусная атака далеко не единственная и даже не самая распространенная причина утраты важных данных. Существуют программные сбои, которые могут вывести из строя операционную систему, а также аппаратные сбои, способные сделать жесткий диск неработоспособным. Всегда существует вероятность утраты компьютера вместе с ценными данными в результате кражи, пожара или иного стихийного бедствия. Поэтому создавать систему безопасности следует в первую очередь «с конца», с предотвращения разрушительных последствий любого воздействия, будь то вирусная атака, кража в помещении или физический выход жесткого диска из строя. Надежная и безопасная работа с данными достигается только тогда, когда любое неожиданное событие, в том числе и полное физическое уничтожение компьютера, не приведет к катастрофическим последствиям.

### Средства антивирусной защиты

Основным средством защиты информации является резервное копирование наиболее ценных данных. В случае утраты информации по любой из вышеперечисленных причин жесткие диски переформатируют и подготавливают к новой эксплуатации. На «чистый» отформатированный диск устанавливают операционную систему с дистрибутивного компакт-диска, затем под ее управлением устанавливают все необходимое программное обеспечение, которое тоже берут с дистрибутивных носителей. Восстановление компьютера завершается восстановлением данных, которые берут с резервных носителей. При резервировании данных следует также иметь в виду и то, что надо отдельно сохранять все регистрационные и парольные данные для доступа к сетевым службам Интернета. Их не следует хранить на компьютере. Обычное место хранения — служебный дневник в сейфе руководителя подразделения. Создавая план меро-

приятий по резервному копированию информации, необходимо учитывать, что резервные копии должны храниться отдельно от компьютера. То есть, например, резервирование информации на отдельном жестком диске того же компьютера только создает иллюзию безопасности. Относительно новым и достаточно надежным приемом хранения ценных, но не конфиденциальных данных является их хранение в *Web*-папках на удаленных серверах в Интернете. Есть службы, бесплатно предоставляющие пространство (до нескольких Мбайт) для хранения данных пользователя.

Резервные копии конфиденциальных данных сохраняют на внешних носителях, которые держат в сейфах, желательно в отдельных помещениях. При разработке организационного плана резервного копирования учитывают необходимость создания не менее двух резервных копий, сохраняемых в разных местах. Между копиями осуществляют ротацию. Например, в течение недели ежедневно копируют данные на носители резервного комплекта «А», а через неделю их заменяют комплектом «Б» и т. д. Вспомогательными средствами защиты информации являются антивирусные программы и средства аппаратной защиты. Так, например, простое отключение перемычки на материнской плате не позволит осуществить стирание перепрограммируемой микросхемы ПЗУ (флэш-*BIOS*) независимо от того, кто будет пытаться это сделать: компьютерный вирус, злоумышленник или неаккуратный пользователь.

Существует достаточно много программных средств антивирусной защиты. Они предоставляют следующие возможности.

- Создание образа жесткого диска на внешних носителях (например, на гибких дисках). В случае выхода из строя данных в системных областях жесткого диска сохраненный «образ диска» может позволить восстановить, если не все данные, то по крайней мере их большую часть. Это же средство может защитить от утраты данных при аппаратных сбоях и при неаккуратном форматировании жесткого диска.
- Регулярное сканирование жестких дисков в поисках компьютерных вирусов. Сканирование обычно выполняется автоматически при каждом включении компьютера и при размещении внешнего диска в считывающем устройстве. При сканировании следует иметь

в виду, что антивирусная программа ищет вирус путем сравнения кода программ с кодами известных ей вирусов, хранящимися в базе данных. Если база данных устарела, а вирус является новым, сканирующая программа его не обнаружит. Для надежной работы следует регулярно обновлять антивирусную программу. Желательная периодичность обновления — один раз в две недели; допустимая — один раз в три месяца.

- Контроль изменения размера и других атрибутов файлов. Поскольку некоторые компьютерные вирусы на этапе размножения изменяют параметры зараженных файлов, контролирующая программа может обнаружить их деятельность и предупредить пользователя.

- Контроль обращений к жесткому диску. Поскольку наиболее опасные операции, связанные с работой компьютерных вирусов, так или иначе обращены на модификацию данных, записанных на жестком диске, антивирусные программы могут контролировать обращения к нему и предупреждать пользователя о подозрительной активности.

### Защита информации в Интернете

При работе в Интернете следует иметь в виду, что насколько ресурсы Всемирной сети открыты каждому клиенту, настолько же и ресурсы его компьютерной системы могут быть при определенных условиях открыты всем, кто обладает необходимыми средствами. Для частного пользователя этот факт не играет особой роли, но знать о нем необходимо, чтобы не допускать действий, нарушающих законодательства тех стран, на территории которых расположены серверы Интернета. К таким действиям относятся вольные или невольные попытки нарушить работоспособность компьютерных систем, попытки взлома защищенных систем, использование и распространение программ (в частности, компьютерных вирусов), нарушающих работоспособность компьютерных систем. Работая во Всемирной сети, следует помнить о том, что абсолютно все действия фиксируются и протоколируются специальными программными средствами и информация как о законных, так и о незаконных действиях обязательно где-то накапливается. Таким образом, к обмену информацией

в Интернете следует подходить как к обычной переписке с использованием почтовых открыток. Информация свободно циркулирует в обе стороны, но в общем случае она доступна всем участникам информационного процесса. Это касается всех служб Интернета, открытых для массового использования. Однако даже в обычной почтовой связи наряду с открытками существуют и почтовые конверты. Использование почтовых конвертов при переписке не означает, что партнерам есть, что скрывать. Их применение соответствует давно сложившейся исторической традиции и устоявшимся морально-этическим нормам общения. Потребность в аналогичных «конвертах» для защиты информации существует и в Интернете. Сегодня Интернет является не только средством общения и универсальной справочной системой — в нем циркулируют договорные и финансовые обязательства, необходимость защиты которых как от просмотра, так и от фальсификации очевидна. Начиная с 1999 года, Интернет становится мощным средством обеспечения розничного торгового оборота, а это требует защиты данных кредитных карт и других электронных платежных средств. Принципы защиты информации в Интернете опираются на определение информации, сформулированное нами в первом разделе этого пособия. Информация — это продукт взаимодействия данных и адекватных им методов. Если в ходе коммуникационного процесса данные передаются через открытые системы (а Интернет относится именно к таковым), то исключить доступ к ним посторонних лиц невозможно даже теоретически. Соответственно, системы защиты сосредоточены на втором компоненте информации — на методах. Их принцип действия основан на том, чтобы исключить или, по крайней мере, затруднить возможность подбора адекватного метода для преобразования данных в информацию. Одним из приемов такой защиты является шифрование данных.

## Рекомендуемая литература

1. Информатика. Базовый курс / под ред. С. В. Симонович. – 2-е изд. – СПб. : Питер, 2005. – 640 с.

2. Гловацкая А. П. Методы и алгоритмы вычислительной математики : учеб. пособие для вузов / А. П. Гловацкая. – М. : Радио и связь, 1999. – 408 с.

3. Мудров А. Е. Численные методы для ПЭВМ на языках Бейсик, Фортран и Паскаль / А. Е. Мудров. – Томск : МП «Раско», 1991. – 272 с.

4. Бланшет Ж. Qt 4: программирование GUI на C++ / Ж. Бланшет, М. Саммерфилд. – М. : Кудиц-Пресс, 2008. – 736 с.

5. Прат С. Язык программирования C++. Лекции и упражнения / С. Прат. – Вильямс, 2012. – 635 с.

Учебное издание

**Семкин** Артем Олегович
**Перин** Антон Сергеевич

# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ. ОБЩИЕ ВОПРОСЫ ИНФОРМАТИКИ, АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

Учебное пособие для студентов
технических направлений подготовки и специальностей

Подписано в печать 23.11.2020.
Формат 60×84/16. Усл. печ. л. 9,53.
Тираж 100. Заказ № 277.

Томский государственный университет
систем управления и радиоэлектроники

634050, г. Томск, пр. Ленина, 40.
Тел. (3822) 533018.