СПРАВОЧНИК

СМС

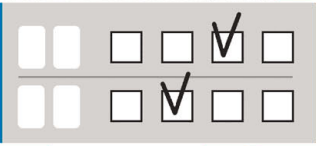
СУПЕРМОБИЛЬНЫЙ

СПРАВОЧНИК

ИНФОРМАТИКА

КРАТКАЯ ТЕОРЕТИЧЕСКАЯ ИНФОРМАЦИЯ

ИНФО



МАТЕРИАЛЫ ДЛЯ ПОДГОТОВКИ К ЕГЭ

ЕГЭ

QR-КОД К КАЖДОЙ ИЗУЧАЕМОЙ ТЕМЕ

QR-КОД

Э

12+

СУПЕРМОБИЛЬНЫЙ

СПРАВОЧНИК

С.Ю. ПАНОВА

ИНФОРМАТИКА

- КРАТКАЯ ТЕОРЕТИЧЕСКАЯ ИНФОРМАЦИЯ
- МАТЕРИАЛЫ ДЛЯ ПОДГОТОВКИ К ЕГЭ
- QR-КОД К КАЖДОЙ ИЗУЧАЕМОЙ ТЕМЕ

МОСКВА

УДК 373.167:004
ББК 32.81я721
П 16

Панова С. Ю.

П 16 Информатика / С. Ю. Панова. – М. : Эксмо, 2013. – 160 с. – (Супермобильный справочник).

ISBN 978-5-699-59586-0

Справочник охватывает весь школьный курс информатики. Материал систематизирован и представлен в сжатом и наглядном виде. С помощью QR-кода предоставляется быстрый доступ к информационным ресурсам общего пользования (Wikipedia) по каждой конкретной теме для самостоятельного углубленного изучения. Справочник поможет эффективно подготовиться к ЕГЭ, а также сэкономить время.

УДК 373.167:004
ББК 32.81я721

ISBN 978-5-699-59586-0

СОДЕРЖАНИЕ

3. Средства ИКТ

ПРЕДИСЛОВИЕ

Справочник представляет собой краткое изложение школьного курса информатики для учащихся старших классов и абитуриентов и ориентирован на подготовку к единому государственному экзамену. В книгу включены материалы по трем разделам школьной программы: «Информационные процессы и системы», «Информационная деятельность человека», «Средства ИКТ».

Справочник прост и удобен в использовании:

- материалы школьного курса систематизированы и изложены в конспективной, удобной для повторения и запоминания форме;
- в справочнике объединены теоретические материалы, соответствующие требованиям и формату ЕГЭ;
- используемые в справочнике QR-коды дают возможность получить максимально быстрый доступ к информационным ресурсам Интернета.

В каждом QR-коде зашифрована ссылка по конкретной теме на информационный ресурс, которую легко можно считать обычным мобильным телефоном, установив специальную программу типа Upcode или ScanLife.

Издание подготовлено в соответствии с современными требованиями школьной программы и может быть полезно при выполнении домашних заданий, подготовке к самостоятельным и контрольным работам, Единому государственному экзамену.

1 ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ И СИСТЕМЫ

1.1. Информация и ее кодирование

1.1.1. Виды информационных процессов

Термин «**информация**» происходит от латинского слова «*informatio*», что означает «разъяснение, изложение, набор сведений». В настоящее время нет четкого определения информации, поскольку это понятие в различных отраслях человеческой деятельности трактуют по-разному.

Информация — это:

- сведения об объектах и явлениях окружающей среды, которые уменьшают степень неопределенности, неполноты знаний о них;
- обозначение содержания, полученного от внешнего мира в процессе приспособления к нему;
- любые сведения, данные, сообщения, передаваемые посредством сигналов;
- мера упорядоченности, структурированности любой материальной системы.

Информационные процессы:

- *сбор информации* — поиск и отбор необходимых сообщений из разных источников;
- *обработка информации* — получение новых сообщений из тех, что имеются;
- *передача информации* — перемещение сообщений от источника к приемнику по каналу передачи;
- *хранение информации* — фиксирование сообщений на материальном носителе;

- *защита информации* — создание условий от случайной потери, повреждения, изменения или несанкционированного доступа к информации.

1.1.2. Процесс передачи информации, источник и приемник информации. Сигнал, кодирование и декодирование. Искажение информации

Информация передается в виде сообщений от некоторого *источника* (живое существо или техническое устройство) информации к ее *приемнику* посредством *канала связи* между ними. Источник посылает передаваемое сообщение, которое кодируется в передаваемый *сигнал*.

Сигнал — это материально-энергетическая форма представления информации; процесс передачи информации, один или несколько параметров которого, изменяясь, отображают сообщение. Сигналы могут быть *аналоговыми* (непрерывными) или *дискретными* (импульсными). Передача информации по каналам связи часто сопровождается воздействием помех, вызывающих *искажение* и *потерю* информации.

Для нейтрализации помех при передаче информации зачастую используют помехоустойчивый избыточный код, который позволяет восстановить исходную информацию даже в случае некоторого ее искажения.

Кодирование — процесс представления информации в определенной стандартной форме. **Декодирование** — обратный процесс восстановления информации из закодированного представления. Кодирование является отобра-

жением произвольного множества A во множество конечных последовательностей (слов) некоторого алфавита B. Декодирование является обратным отображением. Примером кодирования служит представление натуральных чисел в r-й системе исчисления, при котором каждому числу N ставится в соответствие слово $(b_1 b_2 \ldots b_l)$ в алфавите $Br = \{0, 1, \ldots, r-1\}$, причем $b_1 \neq 0$ и $b_1 r_{l-1} + \ldots + b_{l-1} r + b_l = N$.

1.1.3. Дискретное (цифровое) представление текстовой, графической, звуковой информации и видеоинформации. Единицы измерения количества информации

Информация может быть представлена в аналоговой или дискретной форме. При аналоговом представлении физическая величина принимает бесконечное множество значений, причем ее значения изменяются непрерывно. При дискретном представлении физическая величина принимает конечное множество значений, причем ее величина изменяется скачкообразно.

Самым простым способом дискретного представления графических данных является пиксельное кодирование: изображение задается массивом целых чисел (например, для каждой точки цветного изображения могут храниться интенсивности ее красной, зеленой и синей составляющих).

Для дискретного представления информации используется двоичный код, алфавит которого состоит из двух цифр — 0 и 1. Каждая цифра

машинного двоичного кода несет количество информации, равное одному биту.

1 байт = 8 бит
1 килобайт (Кб) = 1024 байта = 2^{10} байт
1 мегабайт (Мб) = 1024 килобайта = 2^{20} байт
1 гигабайт (Гб) = 1024 мегабайта = 2^{30} байт
1 терабайт (Тб) = 1024 гигабайта = 2^{40} байт

Бит — наименьшая единица измерения информации; это количество информации, необходимое для однозначного определения одного из двух равновероятных событий.

Количество информации. Алфавитный подход

При алфавитном подходе информационное сообщение рассматривается как последовательность знаков определенной знаковой системы. По **формуле Хартли** можно рассчитать, какое количество информации I несет каждый из N знаков алфавита:

$$I = \log_2 N$$

Пример 1. В русском алфавите 32 буквы (буква *ё* обычно не используется), т. е. количество возможных информационных событий равно 32. Тогда информационный объем одного символа равен:

$$I = \log_2 32 = 5 \text{ (бит)}$$

Формула Хартли задает связь между количеством возможных событий N и количеством информации I:

$$N = 2^I.$$

Если N не является целой степенью 2, то число $\log_2 N$ не является целым числом и необходимо выполнить округление I в большую сторону.

При решении задач в таком случае I можно найти как $\log_2 N'$, где N' — ближайшая к N степень двойки, такая, что $N' > N$.

Пример 2. В английском алфавите 26 букв. Информационный объем одного символа можно найти так:

$$N = 26;\ N' = 32 = 2^5;\ \ I = \log_2 2^5 = 5 \text{ (бит)}$$

Если количество символов алфавита равно N, а количество символов в записи сообщения — M, то информационный объем данного сообщения вычисляется по формуле:

$$I = M \cdot \log_2 N.$$

Количество информации. Вероятностный подход

Вероятностный подход предполагает, что возможные события имеют различные вероятности реализации. Количество информации I в сообщении о том, что произошло одно из N событий, определяется по **формуле Шеннона**:

$$I = -(p_1 \log_2 p_1 + p_2 \log_2 p_2 + \ldots + pN \log_2 pN),$$

где I — количество информации;
N — количество возможных событий;
p_i — вероятность i-го события.

Пример. Пусть при бросании несимметричной четырехгранной пирамидки вероятности отдельных событий равны:

$$P_1 = \frac{1}{2};\ \ P_2 = \frac{1}{4};\ \ P_3 = \frac{1}{8};\ \ P_4 = \frac{1}{8}.$$

Тогда количество информации, которое будет получено после реализации одного из событий, можно вычислить по формуле Шеннона:

$$I = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{4}\log_2\frac{1}{4} + \frac{1}{8}\log_2\frac{1}{8} + \frac{1}{8}\log_2\frac{1}{8}\right) = \frac{14}{8}\text{ (бит)} = 1{,}75\text{ (бита)}$$

1.1.4. Скорость передачи информации и пропускная способность канала передачи

Скорость передачи информации (*скорость информационного потока*) — количество информации, передаваемое за единицу времени (бит/с, Кбит/с, Мбит/с, байт/с, Кбайт/с, Мбайт/с).

Пропускная способность канала — максимальная скорость передачи информации по каналу связи.

1.2. Системы, компоненты, состояние и взаимодействие компонентов. Информационное взаимодействие в системе, управление, обратная связь

Система — совокупность взаимосвязанных элементов, образующих единое целое и функционирующих совместно для достижения единой цели.

Элемент — простейшая неделимая часть системы.

Информационной системой (ИС) называется программно-аппаратный комплекс, надежно хранящий информацию в памяти компьютера, выполняющий специфические для конкретной предметной области преобразования информации и вычисления, предоставляющий пользователю удобный и легко осваиваемый интерфейс.

Среда — совокупность ресурсов, предоставляемых в распоряжение пользователя системы. Среда должна обеспечивать надежность хранения и эффективность доступа, соответствующие области применения ИС.

В зависимости от конкретной области применения (страхование, транспорт, образование и т. д.) ИС значительно различаются по своим функциям, архитектуре, реализации. Два свойства характерны для всех информационных систем:

- любая ИС предназначена для сбора, хранения и обработки информации;
- все ИС ориентируются на конечного пользователя. Поэтому ИС обязана обладать простым, удобным, легко осваиваемым интерфейсом. Пользователю должны быть предоставлены все необходимые для его работы функции и в то же время не доступны лишние действия. Обычно такой интерфейс является графическим, т. е. содержащим меню, кнопки, подсказки и т. п.

Для функционирования ИС необходимы следующие основные компоненты:

- база данных (БД) и ее схема;
- набор программ, включая СУБД (систему управления базой данных);
- пользователи;
- технические средства.

Технические средства информационных систем могут включать:

- средства вычислительной техники (серверное оборудование, рабочие станции, принтеры и т. д.);
- локальные сети;
- копировально-множительную аппаратуру;
- средства связи (учрежденческие АТС, каналы связи и канальное оборудование, телефоны,

факсимильные аппараты, мобильные средства связи).

Информационное взаимодействие — процесс взаимодействия двух и более субъектов, целью и основным содержанием которого является изменение имеющейся информации хотя бы у одного из них.

Система управления — это совокупность управляющего объекта, объекта управления и каналов прямой и обратной связи между ними. Связь обеспечивает возникновение и сохранение структуры и целостных свойств системы.

Обратные связи, в основном, выполняют осведомляющие функции, отражая изменение состояния системы в результате управляющего воздействия на нее. Процессы управления, адаптации, саморегулирования, самоорганизации, развития невозможны без использования обратных связей.

Связь характеризуется направлением, силой и характером (или видом)

1.3. Моделирование

1.3.1. Описание (модель) реального объекта и процесса, соответствие описания объекту и целям описания. Схемы, таблицы, графики, формулы как описания

Объект — это некоторая часть окружающего мира, рассматриваемая как единое целое. Конкретизировать объект можно с помощью параметров.

Параметры — это признаки, которые характеризуют свойства объекта. Они могут быть *количественными* (рост, вес, возраст, размер и пр.) и *качественными* (форма, материал, цвет, запах, вкус и пр.).

Процесс — смена состояний объекта в течение времени и, как результат, изменение параметров объекта.

Модель — искусственно созданный объект (в виде схем, чертежей, таблиц, логико-математических знаковых формул, компьютерной программы и пр.), который отображает и воспроизводит в более простом, уменьшенном виде структуру, свойства, взаимосвязи и отношения между элементами исследуемого объекта.

Оригинал — исследуемый объект по отношению к модели.

Моделирование — исследование каких-либо объектов на моделях. Объектом моделирования может быть объект, явление или процесс.

Информационная модель — целенаправленно отобранная информация об объекте, представленная в некоторой форме.

Формы представления информационных моделей:

- сигнальная;
- устная (словесная);
- символьная, или знаковая (числа, текст, символы);

- табличная;
- в виде схем, карт;
- графическая.

Пример. Рассмотрим использование табличных моделей для решения задач. Таблица стоимости перевозок между станциями A, B, C, D, E построена следующим образом: числа в ячейках означают стоимость проезда между соответствующими соседними станциями. Если на пересечении строки и столбца пусто, то станции не являются соседними. Стоимость проезда по маршруту складывается из стоимостей проезда между соответствующими соседними станциями. Выбрать таблицу, для которой выполняется условие: «Минимальная стоимость проезда из A в B не больше 6».

Таблица 1

	A	*B*	*C*	*D*	*E*
A			3	1	
B			4		2
C	3	4			2
D	1				
E		2	2		

Таблица 2

	A	*B*	*C*	*D*	*E*
A			3	1	1
B			4		
C	3	4			2
D	1				
E	1		2		

Таблица 3

	A	*B*	*C*	*D*	*E*
A			3	1	
B			4		1
C	3	4			2
D	1				
E		1	2		

Решение. Прежде всего отметим, что данные в таблицах симметричны относительно главной диагонали, т. е. проезд из A в B стоит столько же, сколько из B в A.

Рассмотрим первую таблицу. Выберем все возможные варианты проезда из A в B и рассчитаем их стоимость (в скобках). Стоимость обоих возможных вариантов маршрута равна 7:

1) $AC(3) + CB(4)$; 2) $AC\ (3) + CE(2) + EB(2)$.

Аналогично поступим для второй таблицы. Снова получим стоимость обоих вариантов маршрута, равную 7:

1) $AC(3) + CB(4)$; 2) $AE(1) + EC(2) + CB(4)$.

Выпишем все варианты для третьей таблицы. Стоимость последнего варианта маршрута равна 6:
1) $AC(3) + CB(4)$; 2) $AC(3) + CE(2) + EB(1)$.
Ответ: таблица номер 3 содержит маршрут из A в B, стоимость которого не превышает 6.

Компьютерная модель — это модель, реализуемая с помощью программных средств.

Этапы создания компьютерной модели

1.3.2. Математические модели

Математическая модель — это знаковая модель, сформулированная на языке математики и логики.

В настоящее время расчеты для большинства математических моделей проводят на компьютерах, используя специальные прикладные программные комплексы, которые позволяют:

- в несколько раз сократить время проведения исследований;
- уменьшить количество участников эксперимента;
- повысить точность и достоверность эксперимента, а следовательно, увеличить контроль;
- за счет средств графической визуализации, например анимации, получить реальную «картинку»;
- повысить качество и информативность эксперимента за счет увеличения числа контролируемых параметров и более точной обработки данных. На экране компьютера возможно, например, формирование целой системы приборов, которые будут отслеживать изменение параметров объекта.

1.3.3. Использование сред имитационного моделирования (виртуальных лабораторий) для проведения компьютерного эксперимента в учебной деятельности

Моделирование занимает центральное место в исследовании объекта. Компьютеры предоставляют широкие возможности для постановки компьютерных экспериментов. Компьютерное моделирование позволяет воссоздать явления, которые в реальных условиях воспроизвести невозможно.

Примеры информационных компьютерных моделей для различных отраслей знаний

Физика	Моделирование: ▶ движения на плоскости и в пространстве; ▶ различного вида колебаний, процесса расщепления атомного ядра; ▶ работы двигателя, турбины и т. п.; ▶ магнитных, электрических явлений и т. д.

Химия	Моделирование: ▶ строения молекул; ▶ процесса взаимодействия веществ; ▶ отдельных этапов химического производства; ▶ процессов нагревания или остывания различных тел и т. п.
Биология	Моделирование: ▶ развития биологического объекта в зависимости от условий (например, климатических); ▶ побочных действий лекарственных препаратов; ▶ процесса распространения эпидемий
Экономика	Моделирование: ▶ работы предприятия, банка, отрасли экономики или экономики в целом; ▶ процесса миграции трудовых ресурсов, кризисных явлений в экономике и т. д.

1.4. Системы счисления

1.4.1. Позиционные системы счисления

Система счисления (СС) — это система записи чисел с помощью определенного набора цифр.

Позиционная СС — система, при которой одна и та же цифра имеет различное значение, определяемое местом в числе.

Непозиционная СС — система, у которой значение цифры в числе остается неизменным при вариации ее положения в числе.

Основание — количество различных цифр, используемых в позиционной системе счисления.

Разряд — позиция цифры в числе; возрастает справа налево, от младших разрядов к старшим.

Развернутая форма числа — это запись, которая представляет собой сумму произведений цифр числа на значение позиций.

Развернутая форма записи чисел произвольной системы счисления имеет вид:

$$X = \sum_{i=n-1}^{-m} a_i q^i,$$

где X — число; a — цифры численной записи, соответствующие разрядам; i — индекс; m — количество разрядов дробной части числа; n — количество разрядов целой части числа; q — основание системы счисления.

Пример. Записать развернутую форму десятичного числа 327,46.

Решение. Выпишем количество разрядов числа и основание системы: $n = 3$, $m = 2$, $q = 10$.

$$X = \sum_{i=2}^{-2} a_i q^i = a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1} + a_{-2} \cdot 10^{-2} =$$
$$= 3 \cdot 10^2 + 2 \cdot 10^1 + 7 \cdot 10^0 + 4 \cdot 10^{-1} + 6 \cdot 10^{-2}$$

Основание системы счисления указывают в виде нижнего индекса или буквенного обозначения после числа: B — двоичная система, O — восьмеричная, H — шестнадцатеричная.

Если основание используемой системы счисления больше 10, то для цифр вводят буквенные обозначения (или условные обозначения со скобкой).

В шестнадцатеричной системе счисления 16 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, что соответствует следующим числам десятеричной системы счисления: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15. Примеры чисел: $17D,ECH_{16}$; $F12AH_{16}$.

Пример. Записать развернутую форму двенадцатеричного числа $7A,5B_{12}$.

Решение. В двенадцатеричной системе счисления 10 = A, а 11 = B. Тогда число можно записать так:

$$7A,5B_{12} = B \cdot 12^{-2} + 5 \cdot 12^{-1} + A \cdot 12^0 + 7 \cdot 12^1.$$

Перевод чисел из десятичной системы счисления в заданную

Для преобразования целого числа десятичной системы счисления в число любой другой системы счисления последовательно выполняют деление числа нацело на основание системы счисления, пока не получат нуль. Числа, которые возникают как остаток от деления на основание системы, представляют собой последовательную запись разрядов числа в выбранной системе счисления от младшего разряда к старшему.

Пример. Перевести десятичное число 475 в двоичную систему счисления.

Решение:

475	2	
1	237	2
1	118	2
0	59	2
1	29	2
1	14	2
0	7	2
1	3	2
1	1	2
1	0	

Ответ: 11011011_2.

Для преобразования десятичных дробей в число любой системы счисления последовательно выполняют умножение на основание системы счисления, пока дробная часть произведения не будет равна нулю. Полученные целые части являются разрядами числа в новой системе, и их необходимо представлять цифрами этой новой системы счисления. Целые части в дальнейшем отбрасываются.

Не каждое число может быть точно выражено в новой системе счисления, поэтому иногда вычисляют только требуемое количество разрядов дробной части

Пример. Перевести десятичную дробь $0{,}375_{10}$ в двоичную систему счисления.

Ответ: $0{,}011_2$.

Решение.

$$\begin{array}{rl} \times\ 0{,}375 & \longrightarrow 0 \\ \hline 2 & \\ \times\ 0{,}750 & \longrightarrow 0 \\ \hline 2 & \\ 0{,}150 & \longrightarrow 1 \\ \hline \times\ 0{,}50 & \\ \hline 2 & \\ 1{,}00 & \longrightarrow 1 \end{array}$$

Перевод чисел двоичной системы счисления

1. Перевод числа из восьмеричной системы счисления в двоичный код. Необходимо каждую цифру исходного числа представить триадой двоичных символов ($8 = 2^I$; $I = 3$). Лишние нули в старших разрядах отбрасываются.

Примеры:

$$1234{,}777_8 = 001\ 010\ 011\ 100{,}111\ 111\ 111_2 = 1010011100{,}111111111_2;$$
$$1234567_8 = 001\ 010\ 011\ 100\ 101\ 110\ 111_2 = 1010011100101110111_2.$$

2. Перевод двоичного числа в восьмеричную систему счисления. Следует каждую триаду двоичных цифр заменить восьмеричной цифрой. При этом, если необходимо, число выравнивается путем дописывания нулей перед целой частью или после дробной.

Примеры:

$$1100111_2 = 001\ 100\ 111_2 = 147_8;$$
$$11{,}1001_2 = 011{,}100\ 100_2 = 3{,}44_8;$$
$$110{,}0111_2 = 110{,}011\ 100_2 = 6{,}34_8.$$

3. Перевод двоичного числа в шестнадцатеричное. Необходимо разбить двоичные цифры на группы по четыре и преобразовать каждую группу в шестнадцатеричную цифру ($16 = 2^I$; $I = 4$).

Примеры:

$$1100111_2 = 0110\ 0111_2 = 67_{16};$$
$$11{,}1001_2 = 0011{,}1001_2 = 3{,}9_{16};$$
$$110{,}0111001_2 = 0110{,}0111\ 0010_2 = 6{,}72_{16}.$$

4. Перевод шестнадцатеричного числа в двоичный код. Каждую цифру исходного числа следует представить четверкой двоичных цифр.

Примеры:

$$1234{,}AB77_{16} =$$
$$= 0001\ 0010\ 0011\ 0100{,}1010\ 1011\ 0111\ 0111_2 =$$
$$= 1001000110100{,}1010101101110111_2;$$
$$CE4567_{16} = 1100\ 1110\ 0100\ 0101\ 0110\ 0111_2.$$

Для перевода числа из одной произвольной системы счисления в другую следует выполнить промежуточное преобразование в десятичное число. Для перехода от восьмеричной системы счисления к шестнадцатеричной и обратно в качестве вспомогательного используют двоичный код числа.

1.4.2. Арифметические операции в двоичной системе счисления

Сложение	Вычитание	Умножение
$0 + 0 = 0$	$0 - 0 = 0$	$0 \cdot 0 = 0$
$0 + 1 = 1$	$1 - 0 = 1$	$0 \cdot 1 = 0$
$1 + 0 = 1$	$1 - 1 = 0$	$1 \cdot 0 = 0$
$1 + 1 = 10$	$10 - 1 = 1$	$1 \cdot 1 = 1$

Примеры:

Сложение:

$$\begin{array}{r} 111 \\ +\ 101 \\ \hline 1100 \end{array} \qquad \begin{array}{r} 10101 \\ +\ 1111 \\ \hline 100100 \end{array}$$

Вычитание:

$$\begin{array}{r} 10001 \\ -\ 101 \\ \hline 1100 \end{array} \qquad \begin{array}{r} 11011 \\ -\ 1101 \\ \hline 1110 \end{array}$$

Умножение:

$$\begin{array}{r} 110 \\ \times\ 11 \\ \hline 110 \\ +\ 1100 \\ \hline 10010 \end{array} \qquad \begin{array}{r} 111 \\ \times\ 101 \\ \hline 111 \\ +\ 0000 \\ \hline 11100 \\ \hline 100011 \end{array}$$

Аналогично выполняются арифметические действия в восьмеричной, шестнадцатеричной и других системах счисления. При этом величина переноса в следующий разряд при сложении и заем из старшего разряда при вычитании определяется величиной основания системы счисления.

При выполнении любых арифметических операций над числами, представленными в разных системах счисления, следует предварительно перевести их в одну и ту же систему.

Правило выполнения операции сложения одинаково для всех систем счисления: если сумма складываемых цифр больше или равна основанию системы счисления, то единица переносится в следующий слева разряд. При вычитании, если необходимо, делают заем

Дополнительный код отрицательных чисел

Числовые данные обрабатываются в компьютере в двоичной системе счисления. Существуют разные форматы для хранения чисел в памяти компьютера в двоичном коде. Например, целые числа могут занимать 1, 2 или 4 байта памяти. Если для целых неотрицательных чисел отведен 1 байт (8 бит, или двоичных разрядов), то минимально возможное число — 0 (соответствует восьми нулям, хранящимся в восьми битах), а максимальное — 255 (соответствует восьми единицам).

$$1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + \\ + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 255_{10}.$$

Для n-разрядного представления диапазон будет составлять от 0 до $2n - 1$.

Для хранения целых чисел со знаком в двоичной системе используются два формата: фор-

мат значения числа со знаком и формат дополнительного кода.

В *формате значения числа со знаком* старший (крайний слева) разряд отводится под знак числа: если число положительное, то в знаковый разряд записывается 0, если число отрицательное — 1. Такое представление чисел в компьютере называется *прямым кодом*.

Формат дополнительного кода позволяет заменить арифметическую операцию вычитания операцией сложения, что существенно упрощает работу процессора и увеличивает его быстродействие. В таком случае целые отрицательные числа записываются в виде дополнения модуля отрицательного числа до нуля. Дополнительный код целого отрицательного числа A, хранящегося в n ячейках, равен $2^n - |A|$. Для положительных чисел дополнительный код совпадает с прямым кодом.

Алгоритм получения дополнительного кода отрицательного числа:

1. Записать прямой код модуля числа в n двоичных разрядах.
2. Получить из прямого кода **обратный код** числа (инвертировать) путем замены нулей единицами, а единиц — нулями, кроме цифр знакового разряда. Используется как промежуточное звено.
3. Прибавить единицу к полученному обратному коду.

Пример. Получить дополнительный код числа -2014_{10} для 16-разрядного представления.

Решение.

1. Двоичный код числа 2014_{10} со знаковым разрядом	Прямой код	1000011111011110

2. Инвертирование (исключая знаковый разряд)	Обратный код	1111100000100001
3. Прибавление единицы	Дополнительный код	+ 1111100000100001 0000000000000001 1111100000100010

Арифметические операции с двоичными числами в дополнительном коде

При использовании формата дополнительного кода для алгебраического сложения двоичных чисел положительные слагаемые представляют в прямом коде, а отрицательные — в дополнительном. Затем суммируют эти коды, включая знаковые разряды, которые при этом рассматриваются как старшие разряды. При переносе из знакового разряда единицу переноса отбрасывают. В результате получают алгебраическую сумму в прямом коде, если эта сумма положительная, или в дополнительном, если сумма отрицательная.

Пример. Найти разность $13_{10} - 12_{10}$ для восьмибитного представления.

Решение. Представим заданные числа в двоичной системе счисления:

$13_{10} = 1101_2$ и $12_{10} = 1100_2$.

Запишем прямой, обратный и дополнительный коды для числа -12_{10} и прямой код для числа 13_{10} в восьми битах:

	13_{10}	-12_{10}
Прямой код	00001101	10001100
Обратный код		11110011
Дополнительный код		11110100

Вычитание заменим сложением (для удобства контроля над знаковым разрядом условно отделим его знаком «_»).

$$\begin{array}{r} 0\_0001101 \\ +\ 1\_1110100 \\ \hline 10\_0000001 \end{array}$$

Так как произошел перенос из знакового разряда, первую единицу отбрасываем и в результате получаем 00000001.

1.5. Логика и алгоритмы

1.5.1. Высказывания, логические операции, кванторы, истинность высказывания

Алгебра логики — раздел математической логики, в котором методы алгебры используются в логических преобразованиях.

Логическое высказывание — это любое повествовательное предложение, в отношении которого можно однозначно утверждать, что его содержание истинно или ложно.

В алгебре логики различают *простые высказывания*, обозначаемые латинскими буквами, и *сложные*, составленные из нескольких простых с помощью логических связок, таких как «не», «и», «или», «тогда и только тогда», «если ... то». Истинность или ложность получаемых таким образом сложных высказываний определяется значением простых высказываний.

Логических значений всего два: **истина** (*TRUE*) и **ложь** (*FALSE*). Им соответствует цифровое представление — 1 и 0.

Логические операции

Операция, применяемая к одной величине, называется *унарной*.

Инверсия (логическое отрицание) — логическая операция, в результате которой из данного высказывания получается новое высказывание — *отрицание исходного*. Обозначается символически чертой сверху ($\overline{A}$) или условными обозначениями $\neg A$, *not A* (читается «не *A*», «*A* ложно», «неверно, что *A*», «отрицание *A*»).

Высказывание $\neg A$ ложно, когда *A* истинно, и истинно, когда *A* ложно.

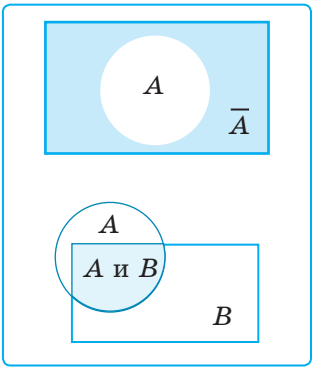
Таблица истинности операции инверсии

A	¬A
истина	ложь
ложь	истина

или

A	¬A
1	0
0	1

Геометрически отрицание можно представить следующим образом: если *A* — некоторое множество точек, то $\overline{A}$ — это дополнение множества *A*, т. е. все точки, которые не принадлежат множеству *A*.



Конъюнкция (логическое умножение) — операция, соединяющая два или более высказывания при помощи связки «и», которая символически обозначается с помощью знака $\wedge$ и читается «*A* и *B*». Для обозначения конъюнкции применяются также следующие знаки: $A \cdot B$; *A & B*, *A and B*, а иногда между высказываниями не ставится никакого знака: *AB*.

Высказывание $A \wedge B$ истинно только тогда, когда оба высказывания *A* и *B* истинны.

Таблица истинности операции конъюнкции

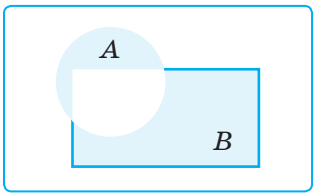
A	B	A ∧ B
истина	ложь	ложь
ложь	истина	ложь
ложь	ложь	ложь
истина	истина	истина

или

A	B	A ∧ B
1	0	0
0	1	0
0	0	0
1	1	1

Геометрически конъюнкцию можно представить следующим образом: если A, B — некоторые множества точек, то $A \wedge B$ — пересечение множеств A и B.

Дизъюнкция (логическое сложение) — операция, соединяющая два или более высказывания при помощи связки «или», которая символически обозначается с помощью знака $\vee$ и читается «A или B». Для обозначения дизъюнкции применяются также следующие знаки: $A + B$; A *or* B; $A \mid B$.



Высказывание $A \vee B$ ложно только тогда, когда оба высказывания A и B ложны.

Таблица истинности операции дизъюнкции

A	B	A ∨ B
истина	ложь	истина
ложь	истина	истина
ложь	ложь	ложь
истина	истина	истина

или

A	B	A ∨ B
1	0	1
0	1	1
0	0	0
1	1	1

Геометрически логическое сложение можно представить следующим образом: если A, B — это некоторые множества точек, то $A \vee B$ — это объединение множеств A и B.

Дизъюнкция строго-разделительная (сложение по модулю два) — логическая операция,

соединяющая два высказывания при помощи связки «или», употребленной в исключающем смысле (называется также *исключающее* «или»). Операция символически обозначается с помощью знака $\oplus$ и читается «либо A, либо B».

Высказывание $A \oplus B$ истинно только тогда, когда высказывания A и B имеют различные значения.

Таблица истинности операции строгой дизъюнкции

A	B	A ⊕ B
истина	ложь	истина
ложь	истина	истина
ложь	ложь	ложь
истина	истина	ложь

или

A	B	A ⊕ B
1	0	1
0	1	1
0	0	0
1	1	0

Импликация — логическая операция, соединяющая два высказывания при помощи связки «если... то» в сложное высказывание, которое символически обозначается с помощью знака $\rightarrow$ и читается «если A, то B», «A влечет B», «из A следует B», «A имплицирует B». Для обозначения импликации применяется также знак $\supset$.

Для операции импликации справедливо утверждение, что из лжи может следовать все что угодно, а из истины — только истина.

Таблица истинности операции импликации

A	B	A → B
истина	ложь	ложь
ложь	истина	истина
ложь	ложь	истина
истина	истина	истина

или

A	B	A → B
1	0	0
0	1	1
0	0	1
1	1	1

Эквивалентность (двойная импликация, равнозначность) — логическая операция, позволяющая из двух высказываний A и B получить новое высказывание $A \equiv B$, которое читается «A эквивалентно B». Эта операция может быть выражена связками «тогда и только тогда», «необходимо и достаточно», «равносильно». Для обозначения эквивалентности применяются также знаки ~, $\Leftrightarrow$.

Таблица истинности операции эквивалентности

A	B	A ~ B
истина	ложь	ложь
ложь	истина	ложь
ложь	ложь	истина
истина	истина	истина

или

A	B	A ~ B
1	0	0
0	1	0
0	0	1
1	1	1

Связь между логическими операциями

Сложение по модулю два	$A \oplus B$	$(\overline{A} \wedge B) \wedge (A \wedge \overline{B})$
Импликация	$A \to B$	$\overline{A} \vee B$
Эквивалентность	$A \sim B$	$(\overline{A} \wedge \overline{B}) \vee (A \wedge B)$

Значения сложных высказываний

Для сложных логических выражений, содержащих несколько логических операций, определен порядок выполнения действий (*приоритет*).

Приоритет выполнения логических операций следующий: отрицание «не» имеет самый высокий приоритет. Затем выполняется конъюнкция «и», после конъюнкции — дизъюнкция «или».

Приоритет выполнения действий

1. Вычисление существующих функциональных зависимостей.
2. Выполнение алгебраических операций (вначале возведение в степень, затем умножение и деление, после чего вычитание и сложение).
3. Выполнение операций сравнения (в порядке записи).
4. Выполнение логических операций (сначала операции отрицания, затем операции логического умножения, потом операции логического сложения, последними выполняются операции импликации и эквивалентности).

В логическом выражении могут использоваться скобки, которые изменяют порядок выполнения операций.

Пример 1. Указать множество целых значений X, для которых истинно выражение:

$$\neg ((X > 2) \to (X > 5)).$$

Решение. Операция отрицания применена ко всему выражению $((X > 2) \to (X > 5))$, следовательно, когда выражение $\neg((X > 2) \to (X > 5))$ истинно, выражение $((X > 2) \to (X > 5))$ ложно. Определим, для каких значений X ложно последнее выражение. Операция импликации принимает значение «ложь» только в одном случае: когда из истины следует ложь. А это выполняется только для $X = 3$; $X = 4$; $X = 5$.

Пример 2. Для каких из приведенных слов ложно следующее высказывание?

¬ (первая буква гласная ∧ третья буква гласная) ⇔ строка из 4 символов

1) алло; 2) кукуруза; 3) ошибка; 4) силач.

Решение. Определим значения высказывания последовательно для всех предложенных слов:

1) для слова *алло*: $\neg(1 \wedge 0) \Leftrightarrow 1$, $1 \Leftrightarrow 1$ — высказывание истинно;
2) для слова *кукуруза*: $\neg (0 \wedge 0) \Leftrightarrow 0$, $1 \Leftrightarrow 0$ — высказывание ложно;
3) для слова *ошибка*: $\neg (1 \wedge 1) \Leftrightarrow 0$, $0 \Leftrightarrow 0$ — высказывание истинно;
4) для слова *силач*: $\neg (0 \wedge 0) \Leftrightarrow 1$, $1 \Leftrightarrow 0$ — высказывание ложно.

Пример 3. Найти значение выражения:

$$1 \le a \vee A \vee \sin\left(\frac{\pi}{a} - \frac{\pi}{b}\right) < 1 \wedge \neg B \wedge \neg(b^a + a^b > a + b \vee A \wedge B)$$

для $a = 2$, $b = 3$, $A =$ *истина*, $B =$ *ложь*.
Решение. Подставим значения в выражение и вычислим его результат в соответствии с порядком операций.
1) $b^a + a^b > a + b$. Результат подстановки значений: $3^2 + 2^3 > 2 + 3$, т. е. $17 > 2 + 3$. Значение — истина.
2) $A \wedge B$. Результат подстановки: истина $\wedge$ ложь. Значение — ложь.
Следовательно, значение выражения ($b^a + a^b > a + b \vee \vee A \wedge B$) равно истина $\vee$ ложь = истина.
3) $1 \le a$. Результат подстановки: $1 \le 2$. Значение — истина.
4) $\sin\left(\frac{\pi}{a} - \frac{\pi}{b}\right) < 1$. Результат подстановки: $\sin\left(\frac{\pi}{2} - \frac{\pi}{3}\right) < 1$.
Значение — истина.
После этих вычислений получим:
истина $\vee A \vee$ истина $\wedge \neg B \wedge \neg$ истина.
Теперь надо выполнить сначала операции отрицания, затем — логического умножения и сложения.
5) $\neg B$. Результат подстановки: $\neg$ ложь. Значение — истина.
$\neg$ истина. Значение — ложь.
6) истина $\wedge$ истина $\wedge$ ложь. Значение — ложь.
7) истина $\vee A \vee$ ложь. Значение — истина.
Таким образом, результат логического выражения при заданных значениях — истина.

Кванторы

Квантор — общее название для логических операций, ограничивающих область истинности какого-либо утверждения. Чаще всего используют:

- *квантор всеобщности* (обозначается $\forall$, читается «для всех...», «для каждого...» или «каждый...», «любой...», «для любого...»).
- *квантор существования* (обозначается $\exists$, читается «существует...», «найдется...»).

В математической логике приписывание квантора к формуле называется *связыванием*, или *квантификацией*

Тождественные преобразования логических выражений

В алгебре логики выполняются основные законы, позволяющие производить тождественные преобразования логических выражений.

Закон	Для дизъюнкции $\vee$	Для конъюнкции $\wedge$
Переместительный	$a \vee в = в \vee a$	$a \wedge в = в \wedge a$
Сочетательный	$a \vee (в \vee c) = (a \vee в) \vee c$	$a \wedge (в \wedge c) = (a \wedge в) \wedge c$
Распределительный	$a \wedge (в \vee c) = (a \wedge в) \vee (a \wedge в)$	$a \vee в \wedge c = (a \vee в) \wedge (a \vee c)$
Правила де Моргана	$\overline{a \vee в} = \bar{a} \wedge \bar{в}$	$\overline{a \wedge в} = \bar{a} \vee \bar{в}$
Идемпотенции	$a \vee a = a$	$a \wedge a = a$
Поглощения	$a \vee a \wedge в = a$	$a \wedge (a \vee в) = a$
Склеивания	$(a \wedge в) \vee (\bar{a} \wedge в) = в$	$(a \vee в) \wedge (\bar{a} \vee в) = в$
Операция переменной с ее инверсией	$a \vee \bar{a} = 1$	$a \wedge \bar{a} = 0$

Закон	Для дизъюнкции $\vee$	Для конъюнкции $\wedge$
Операция с константами	$a \vee 0 = a$ $a \vee 1 = 1$	$a \vee a \wedge в = a$ $a \wedge 0 = 0$
Двойного отрицания	$\overline{\overline{a}} = a$	

Пример 1. Найти логическое выражение, равносильное выражению $A \wedge \neg (\neg B \vee C)$.

Решение. Применяем правило де Моргана для B и C: $A \wedge \neg (\neg B \vee C) = B \wedge \neg C$.

Получаем выражение, равносильное исходному:

$$A \wedge \neg (\neg B \vee C) = A \wedge B \wedge \neg C.$$

Ответ: $A \wedge B \wedge \neg C$.

Пример 2. Указать значение логических переменных A, B, C, для которых значение логического выражения $(A \vee B) \rightarrow (B \vee \neg C \vee B)$ ложно.

Решение. Операция импликации ложна только в случае, когда из истинной посылки следует ложь. Следовательно, для заданного выражения посылка $A \vee B$ должна принимать значение истина, а следствие (т. е. выражение $B \vee \neg C \vee B$) — ложь.

1) $A \vee B$. Результат дизъюнкции — истина, если хотя бы один из операндов — истина.

2) $B \vee \neg C \vee B$. Выражение ложно, если все слагаемые имеют значение ложь, т. е. значения B — ложь; $\neg C$ — ложь, а следовательно, C имеет значение истина.

3) Если рассмотреть посылку и учесть, что B — ложь, то получим, что значение A — истина.

Ответ: значения A — истина, B — ложь, C — истина.

Построение таблиц истинности логических выражений

Для логической формулы всегда можно записать **таблицу истинности**, т. е. представить заданную логическую функцию в табличном виде.

Пример. Рассмотрим построение таблицы истинности для формулы $\overline{X1} \wedge X2 \vee \overline{X1 \vee X2} \vee X1$.

Решение.

$X1$	1	1	0	0
$X2$	1	0	1	0
$\overline{X1}$	0	0	1	1
$\overline{X1} \wedge X2$	0	0	1	0
$X1 \vee X2$	1	1	1	0
$\overline{X1 \vee X2}$	0	0	0	1
$\overline{X1} \wedge X2 \vee \overline{X1 \vee X2}$	0	0	1	1
$\overline{X1} \wedge X2 \vee \overline{X1 \vee X2} \vee X1$	1	1	1	1

Если функция принимает значение 1 при всех наборах значений переменных, то она является **тождественно-истинной**. Если при всех наборах входных значений функция принимает значение 0, она является **тождественно-ложной**. Если набор выходных значений содержит как значения 0, так и 1, функция называется **выполнимой**.

Пример 2. Составить таблицу истинности для охранного устройства, которое использует три датчика и срабатывает при замыкании только двух из них.

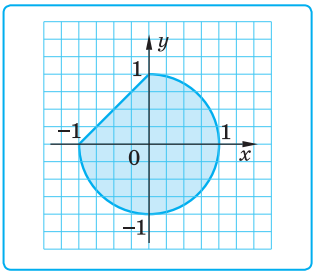
Решение. Очевидно, что результатом решения будет таблица, в которой искомая функция $Y(X1, X2, X3)$ будет иметь значение *истина*, если какие-либо две переменные имеют значение *истина*.

$X1$	1	1	1	1	0	0	0	0
$X2$	1	1	0	0	1	1	0	0
$X3$	1	0	1	0	1	0	1	0
$Y(X1, X2, X3)$	0	1	1	0	1	0	0	0

Описание геометрических областей с помощью логических выражений

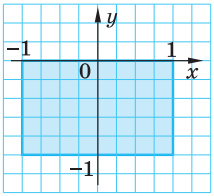
Логические выражения могут быть использованы для описания геометрических объектов. В этом случае задача формулируется так: «Записать для заданной геометрической области такое логическое выражение, которое принимает значение истина для значений x, y тогда и только тогда, когда любая точка с координатами $(x; y)$ принадлежит геометрической области».

Пример 1. Задано изображение геометрической области. Записать логическое выражение, описывающее множество точек, принадлежащих ей.



Решение. Заданную геометрическую область можно представить в виде набора следующих областей. Первая область ($D1$) — полуплоскость $x/(-1) + y/1 \le 1$; вторая ($D2$) — круг с центром в начале координат $x^2 + y^2 \le 1$. Их пересечение ($D1 \cap D2$) представляет собой искомую область.

Ответ: логическое выражение $x/(-1) + y/1 \le 1 \wedge x^2 + y^2 \le 1$.

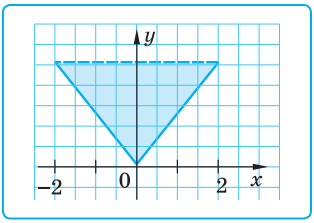


Пример 2. Задано изображение геометрической области. Записать логическое выражение, описывающее множество точек, принадлежащих ей.

Решение. Эту область можно записать так: $|x| \le 1 \wedge y \le 0 \wedge y \ge -1$.

Пример 3. Нарисовать и заштриховать область, для точек которой выполняется логическое условие $y \ge x \wedge y + x \ge 0 \wedge y < 2$.

Решение. Искомая область представляет собой пересечение трех полуплоскостей. Строим на плоскости (x, y) прямые $y = x$; $y = -x$; $y = 2$. Это границы области, причем последняя граница $y = 2$ не принадлежит области, поэтому ее наносим пунктирной линией. Для выполнения неравенства $y \ge x$ нужно, чтобы точки находились слева от прямой $y = x$, а неравенство $y = -x$ выполняется для точек, которые находятся справа от прямой $y = -x$. Условие $y < 2$ выполняется для точек, лежащих ниже прямой $y = 2$.



В результате получим область, изображенную на рисунке.

1.5.2. Цепочки (конечные последовательности), деревья, списки, графы, матрицы (массивы), псевдослучайные последовательности

Граф — это множество вершин и множество ребер, элементами которого являются пары вершин.

Дерево — это связный ациклический граф. Связность означает наличие путей между любой парой вершин, ацикличность — отсутствие циклов и то, что каждую пару вершин соединяет только один путь.

Виды графов:

- *ориентированный граф* — граф с ориентированными ребрами (имеющими направление; отображаются линиями со стрелками);
- *взвешенный граф* — граф, ребра которого имеют вес (число, поставленное в соответствие ребру);
- *полный граф* — граф со всеми возможными ребрами;
- *подграф* — граф, содержащий подмножество вершин данного графа и подмножество инцидентных им (соединяющих) ребер;
- *связный граф* — граф, в котором между любыми двумя вершинами есть путь;
- *псевдограф* — граф, в котором допустимы кратные ребра и петли.

Определения теории графов

Маршрут (или цепь) — последовательность вершин, соединенных ребрами (в ориентированном графе — однонаправленными).

Путь — маршрут без повторяющихся вершин.

Цикл — замкнутый путь.

Компонента связности — связный подграф.

Мост — ребро, при удалении которого граф распадается на несколько компонент связности.

Висячая вершина (или лист) — вершина, из которой выходит одно ребро.

Гамильтонов путь — путь, проходящий через все вершины графа по одному разу.

Эйлеров маршрут — маршрут, проходящий по всем ребрам графа по одному разу.

Способы представления графов в компьютере — «оцифровка» графов

1. *Матрица смежности.* Это таблица $n \times n$, где n — количество вершин графа. Если из вершины

i есть ребро к вершине j, то в клетке с координатами (i, j) указывают 1, а если нет — то 0. Для взвешенного графа вместо 1 указывают вес. Если граф неориентированный, то матрица получается симметричной. Таким способом нельзя задать только взвешенные псевдографы, но они почти никогда не применяются. Такой способ требует $n \times n$ ячеек памяти.

2. *Список ребер.* Простое перечисление всех ребер: для каждого ребра — начальная и конечная вершины (для взвешенного графа — еще и вес). Такой способ задания требует достаточно мало памяти: $2 \times m$ ячеек, где m — количество ребер. Обычно этот способ используется для задания входных данных, которые затем конвертируются программой в другую форму представления.
3. *Список смежности* — набор строк по числу вершин графа. Каждая вершина начинает одну из строк списка, а правее в строке перечислены вершины, связанные с ней ребрами. Таким образом, всего будет столько же значений, сколько ребер графа, не считая первых ячеек каждой строки. Необходимое количество памяти: $m + n$ ячеек, где m — количество ребер графа, n – количество его вершин.

Пример. Сравнить объем памяти, необходимый для представления графа в виде матрицы смежности и списка смежности. Рассмотреть критичный случай — полный граф.

Решение. Подсчитаем, сколько ребер у графа. Из каждой из n вершин исходит $n - 1$ ребро (ко всем оставшимся вершинам). Учтем, что у каждого ребра два конца и при сложении количества ребер для всех вершин мы посчитаем его дважды. Итого

получаем для полного графа $m = n \times (n - 1)/2$. Таким образом, в критичном случае представление графа в виде списка смежности занимает порядка $n \times n$ ячеек памяти — такое же количество, что и матрица смежности.

Псевдослучайная последовательность — последовательность чисел, которые вычисляются по некоторым правилам, но соответствуют случайным числам в задаче. Чаще всего для генерации таких последовательностей используют специальные функции. Например, $(t^5 + 4 \cdot t^3 + 11 \cdot t) \bmod 100$, где t — количество секунд, прошедших за сутки. Можно также использовать в формулах расчета такие величины, как температура чипа или количество параллельных процессов.

1.5.3. Выигрышные стратегии

Выигрышная стратегия — это определение алгоритма действий в игре первого игрока, при котором он выиграет независимо от действий второго.

Конечную игру с полной информацией можно представить в виде ориентированного графа, вершинами которого являются все допустимые позиции игры, а ребра указывают возможные ходы. Этот граф обязательно будет ациклическим (не будет содержать циклов), в противном случае окончание игры не гарантировано.

Для определения результата в ряде игр не требуется просчитывать большое число позиций. Для них удается описать выигрышную стратегию с помощью простой функции, аргументом которой является позиция, а значением — вы-

игрышный ход для данной позиции или сообщение о том, что позиция является проигрышной.

Пример 1. За круглым столом двое играют одинаковыми монетами. За ход можно выложить одну монету на стол (не накладывая на уже лежащие монеты). Проиграет тот, для чьих монет на столе не останется места. Кто выиграет при правильной игре?
Решение. Выиграет второй игрок: он должен ставить монету симметрично ходу первого игрока относительно центра стола. Тогда если первый игрок смог положить монету, то второй тоже сможет положить на симметричную позицию.

Пример 2. На столе лежат N камней. Двое играющих по очереди могут взять от одного до четырех камней. Кто не может сделать ход (камней не осталось) — проигрывает. Определить выигрышную стратегию.
Решение. Если N делится на 5 без остатка, то второй игрок может гарантировать себе выигрыш, дополняя ход противника до 5 (если первый взял один камень, то второй берет четыре и т. д.). Если N на 5 не делится, то выигрывает первый игрок: он должен сначала взять число камней, равное остатку от деления N на 5, а потом дополнять ход противника до 5. Для первого игрока все позиции с числом камней, кратным 5, являются проигрышными, остальные — выигрышными.

1.5.4. Сложность вычисления; проблема перебора

Зависимость количества операций алгоритма от его параметра n может быть указана с помощью математического обозначения $O(n)$ (читается «о большое от n»). Функция f является $O(g)$, если f не превышает функцию g, умноженную на константу. Например, $2x^2 - 3x + 5$ является $O(x^2)$.

Таким образом, алгоритм с асимптотикой $O(n^2)$ работает порядка $n^2 \cdot C$ времени, где C — некоторая константа.

Пример 1. Определить, сколько времени потребуется для сортировки 10^4 чисел по алгоритму, сложность которого $O(n^2)$.

Решение. Принято считать, что современный компьютер за секунду совершает 10^8 элементарных операций. (Это можно определить следующим образом. Если мощность процессора 2,5 ГГц, значит, он выполняет $2{,}5 \cdot 10^9$ тактов в секунду. Одна элементарная операция занимает порядка 25 тактов, следовательно, за 1 с будет выполнено 10^8 элементарных операций.) Значит, если алгоритм применить для сортировки 10^4 чисел, потребуется 10^8 операций, которые будут выполнены за 1 с.

Пример 2. Имеется отсортированный массив из n чисел, необходимо найти в нем заданное число. Эта задача очень актуальна: она применима не только к числам, но и к любым хранимым объектам, доступ к которым происходит по числовому ключу.

Решение. Один из вариантов решения — перебрать все числа по очереди и проверить, есть ли среди них искомое. Такое решение работает за $O(n)$. Теперь рассмотрим метод так называемого бинарного поиска: возьмем среднее число массива и сравним его с искомым. Если искомое число больше, то надо искать его в большей половине массива, а если нет — то в меньшей. Таким образом, мы каждым действием сужаем область поиска в два раза, а значит, всего будет сделано $O(\log_2 n)$ операций.

1.5.5. Кодирование с исправлением ошибок

Самоконтролирующиеся коды — это коды, которые позволяют обнаружить наличие ошибок.

Блочные коды — это коды, в которых информация передается по частям, блокам одинаковой длины.

Самокорректирующиеся коды — это коды, позволяющие обнаружить непосредственное место ошибки.

Код Хэмминга

Код Хэмминга — самоконтролирующийся и самокорректирующийся код. В соответствии с ним к исходному сообщению добавляют контрольные (проверочные) биты, каждый из которых проверяет наличие ошибки в определенной группе битов сообщения.

Пронумеруем биты в результирующем сообщении. Контрольные биты размещают в нем на позициях, чьи номера являются степенями 2; остальные позиции занимают биты исходного сообщения. Рассмотрим номера всех позиций в двоичной системе счисления. Группы проверок организуют так: в первой группе — все позиции с номерами, в двоичной записи которых в младшем (первом) бите стоит 1; во второй группе — все позиции с номерами, в записи которых во втором по старшинству бите стоит 1; в третьей — если у номера в третьем по старшинству бите 1 и т. д. Для каждой группы заполним соответствующий ей проверочный бит, дополняющий сумму единиц в группе битов до четного числа.

Посмотрим, сколько же проверяющих разрядов понадобится. Пусть результирующее сообщение длиной n бит состоит из m информационных и k проверочных бит. Тогда k должно быть равно длине двоичной записи максимального номера последовательности:

$$k = \log_2 (n + 1).$$

Количество m информационных бит определяется так:

$$k = \log_2 (m + k + 1) \Leftrightarrow m + k + 1 = 2^k \Leftrightarrow m = 2^k - k - 1.$$

Таблица соответствия информационных и контрольных битов

k	2	3	4	5	6
m	1	2 – 4	5 – 11	12 – 26	27 – 57

Допустим, при передаче сообщения произошла ошибка в бите с номером x. Достаточно узнать, в каких разрядах двоичного разложения x стоят единицы, чтобы получить весь номер x. Вычислим для принятого сообщения (без проверочных разрядов) контрольные биты и выясним, какие из них не сходятся с полученными. Допустим, не сходится контрольный бит, отвечающий за группу битов, номера которых содержат единицу, например, в 4-м разряде. Значит, в числе x в 4-м разряде единица. Таким образом можно восстановить весь номер x.

Пример 1. Требуется отправить сообщение 0100010000111101. Сформировать для него код Хэмминга.

Решение. Нумеруем позиции от 1 до 16. Для 16 информационных потребуется 5 проверочных битов. Они будут вставлены в сообщение на позиции с номерами, кратными 2: xx0x100x0100001x11101. Чтобы определить их значение, формируем 5 групп битов, расположенных правее проверочных:

1) биты на позициях 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 (двоичная запись этих номеров заканчивается единицей);

2) биты 3, 6, 7, 10, 11, 14, 15, 18, 19 (в предпоследнем разряде двоичной записи номеров единица);

3) биты 5, 6, 7, 12, 13, 14, 15, 20, 21 (единицы в 3-м разряде);
4) биты 9–15 (единицы в 4-м разряде);
5) биты 17–21 (единицы в 5-м разряде).
Суммируем биты первой группы — сумма нечетная, значит, нужно добавить до четности 1 (первый контрольный бит — 1). Сумма второй группы четная, значит, второй контрольный бит — 0. Для третьей группы — 1, для четвертой и пятой — 0. Итак, закодированное сообщение имеет вид: **10011000010000101 1101**.

Пример 2. Проверить, имеется ли ошибка в полученном сообщении 10011000011000101 1101, закодированном кодом Хэмминга, и в случае наличия исправить ее.

Решение. Удаляем из полученного сообщения проверочные биты (на 1-й, 2-й, 4-й, 8-й и 16-й позициях: **10011000011000101 1101**, т. е. биты 1, 0, 1, 0, 0). Для оставшегося сообщения в соответствии с алгоритмом примера 1 вычисляем проверочные биты: 0, 1, 1, 1, 0. Сравниваем их с удаленными проверочными битами — не совпали биты 1-й, 2-й и 8-й позиций. Следовательно, сообщение пришло с ошибкой, ошибочный бит имеет номер $1 + 2 + 8 = 11$. Изменяем значение 11-го бита — получаем отправленное сообщение: 0100010000111101.

1.5.6. Сортировка

Сортировка данных — это упорядочение данных по некоторому признаку.

Сортировка слиянием — один из методов сортировки данных. Для этой сортировки массив данных надо разбить на 2 части (например, пополам), отсортировать таким же способом каждую часть, а потом соединить их.

Сортировка основана на *рекурсии*, то есть существует часть метода (в программе реализованная в виде функции), которая использует сама себя. Задача разбивается на подзадачи до тех пор, пока в какой-то момент в выделенной части массива не останется один элемент. Эти действия удобно представить в виде дерева рекурсии (в данном случае бинарным): вершины — вызовы функции, ребра — взаимоотношения между вызовами. Тогда на дереве можно увидеть глубину рекурсии — расстояние от корня дерева до листьев. В данном случае глубина равна $\log_2 n$, так как массив все время делится пополам (где n — количество элементов исходного массива).

Слияние (соединение) двух отсортированных фрагментов происходит следующим образом. На каждом шаге берут меньший из двух первых элементов подмассивов и переносят его в результирующий массив. Эту операцию повторяют до тех пор, пока не закончатся элементы в одном из фрагментов. Тогда все оставшиеся элементы второго фрагмента дописывают в конец результирующего массива.

Таким образом, общая сложность сортировки $O(n \cdot \log_2 n)$.

1.6. Элементы теории алгоритмов

1.6.1. Формализация понятия алгоритма

Алгоритм — определенная последовательность действий для решения поставленной задачи.

Исполнитель — субъект, способный исполнять некоторый набор команд.

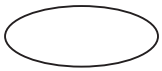
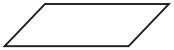
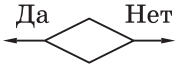
Язык — алфавит и некоторый набор правил, которым должен следовать алгоритм, написанный на этом языке.

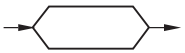
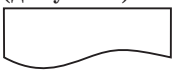
Способы представления алгоритмов:

- словесная запись (на естественном языке);
- блок-схема (графическое изображение);
- псевдокод;
- формальные алгоритмические языки (языки программирования).

Блок-схемы

Функциональные блоки языка блок-схем

Название символа. Графическое изображение	**Описание**
Пуск / останов 	Блоки начала и конца алгоритма
Ввод / вывод данных 	Блоки ввода / вывода в общем виде
Процесс (операторные блоки) 	Выполнение вычислительного действия или последовательности действий, которые изменяют значение, форму представления или размещение данных
Условие (условный блок) 	Выбор направления выполнения алгоритма. Если условие, записанное внутри ромба, выполняется, то управление передается по стрелке «да», в противном случае — по стрелке «нет»

Название символа. Графическое изображение	Описание
Начало цикла с параметром 	Организация циклических конструкций с известным количеством *итераций* (повторений) и известным шагом изменения параметра цикла. Внутри блока для параметра цикла указываются через запятую его начальное значение, конечное значение и шаг изменения
Предопределенный процесс 	Обращение к вспомогательным алгоритмам, самостоятельным модулям, библиотечным подпрограммам
Печать (документ) 	Вывод результатов на печать

Псевдокод

Псевдокод занимает промежуточное положение между естественным языком и языками программирования.

Учебный алгоритмический язык — пример псевдокода. Алфавит учебного алгоритмического языка является открытым. Кроме алфавита, в алгоритмической нотации определяются *служебные слова*, которые являются неделимыми. Служебные слова обычно выделяются жирным шрифтом или подчеркиванием.

Служебные слова учебного алгоритмического языка

алг (заголовок алгоритма)	**нц** (начало цикла)	**знач**
нач (начало алгоритма)	**кц** (конец цикла)	**и**
кон (конец алгоритма)	**дано**	**или**
арг (аргумент)	**надо**	**не**
рез (результат)	**если**	**да**
цел (целый)	**то**	**нет**
сим (символьный)	**иначе**	**при**
лит (литерный)	**всё**	**выбор**
лог (логический)	**пока**	**утв**
вещ (вещественный)	**для**	**ввод**
таб (таблица)	**от**	**вывод**
длин (длина)	**до**	

Вид записи алгоритма на псевдокоде

алг название алгоритма (аргументы
и результаты)
дано условия применимости алгоритма
надо цель выполнения алгоритма
нач описание промежуточных величин,
последовательность команд (тело
алгоритма)
кон

Команды учебного языка:

- *оператор присваивания.* Обозначается «:=» и служит для вычисления выражения, стоящего справа, и присваивания его значения переменной, указанной слева. Например, если переменная a имела значение 5, то после выполнения оператора присваивания $a := a + 1$, значение переменной a изменится на 6.
- *операторы ввода/вывода:*
  ввод (список имен переменных)
  вывод (список вывода)

Список вывода может содержать комментарии, которые заключаются в кавычки.

- *операторы ветвления*
  если… то… иначе… всё
  выбор… при… иначе… всё
- *операторы цикла*
  нц для… от… до… кц
  нц пока… кц
  нц… до… кц

Пример 1. Запишем на псевдокоде алгоритм сложения квадратов целых чисел от 1 до n включительно.

Решение.

алг Сумма квадратов **(арг цел** n**, рез цел** S**)**
дано | $n > 0$
надо | $S = 1 \cdot 1 + 2 \cdot 2 + 3 \cdot 3 + \ldots + n \cdot n$
нач
  цел i
  ввод n
    $S := 0$
  нц для i **от** 1 **до** n
    $S := S + i * i$
  кц
  вывод "$S =$ ", S
кон

При записи алгоритма в словесной форме, в виде блок-схемы или на псевдокоде допускается произвольное изображение команд.

Пример 2. Исполнитель *Устроитель* может выполнить только две команды, одна из которых уменьшает число на 1, вторая — увеличивает его втрое. Командам присвоены номера: *1* — вычти 1; *3* — умножь на 3.

Написать набор команд (не более пяти) получения из числа 3 числа 16. В ответе указать только номера команд.

Решение.

Номер команды	*1*	*3*	*3*	*1*	*1*
Результат выполнения	3 – 1 = = 2	2 · 3 = = 6	6 · 3 = = 18	18 – 1 = = 17	17 – 1 = = 16

Ответ: 13311.

Пример 3. Черепашка является исполнителем для создания графических объектов на рабочем поле. При движении Черепашка оставляет след в виде линии. Черепашка может исполнять следующие команды:

Название команды	**Параметры**	**Действия исполнителя**
вп	*число шагов*	Продвигается в направлении головы на указанное число шагов
нд	*число шагов*	Продвигается в направлении, противоположном направлению головы, на указанное число шагов
пр	*число градусов*	Поворачивается направо относительно направления, заданного головой Черепашки
лв	*число градусов*	Поворачивается налево относительно направления, заданного головой Черепашки
повтори	*количество повторений, список повторяемых команд*	Повторяет действия, указанные в списке повторяемых команд, заданное число раз. Список (тело цикла) должен быть заключен в квадратные скобки

Записать для исполнителя Черепашка алгоритмы:
а) построения квадрата со стороной 100;

б) построения правильного шестиугольника со стороной 50;
в) построения изображения цифры 4, если голова Черепашки смотрит на север.
Ответы: а) **Повтори 4 [вп 100 пр 90]**; б) **Повтори 6 [вп 50 пр 360/6]**; в) **вп 100; повтори 2 [лв 135 вп 50]**.

Основные понятия языков программирования

Константа — величина, которая в ходе выполнения программы не изменяет своего значения.

Переменная — это объект программы, имеющий имя и изменяемое значение. Для хранения переменной выделяются ячейки памяти компьютера.

Величина — объект-переменная, с которой связывается определенное множество значений. Такому объекту присваивается имя — *идентификатор*.

Значение — это непосредственно то, чему равна переменная в конкретный момент времени (может быть число, символ, текст и т. д.). Значение переменной в программе можно задать двумя способами: присваиванием и с помощью процедуры ввода.

Тип переменной определяет диапазон всех значений, которые может принимать данная переменная, и допустимые для нее операции.

Стандартные типы — это числовые, литерные и логические типы.

Литерный тип — это символы и строки, он дает возможность работать с текстом. *Литерные величины* — это произвольные последовательности символов — букв, цифр, знаков препинания, пробела и других специальных знаков (возможными символами могут быть символы таблицы ASCII). В учебном алгоритмическом языке литерные величины обозначаются **лит**. Во многих языках программирования различают символьный **(char)** и строковый **(string)** типы.

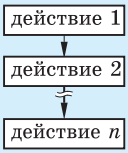
Логический тип определяет логические переменные, которые могут принимать только два значения — истина (True) или ложь (False).

Основные алгоритмические конструкции

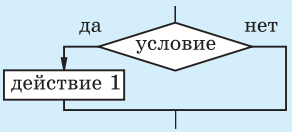
Логическая структура любого алгоритма может быть представлена комбинацией трех базовых структур:

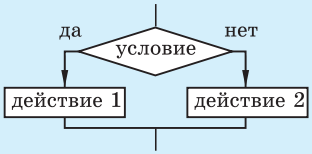
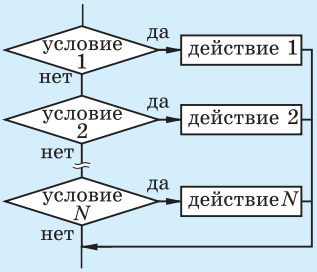
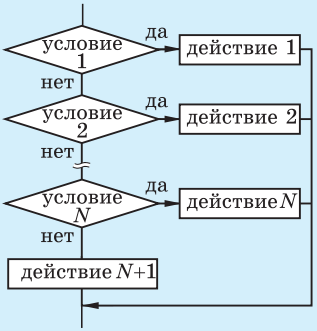
1) *следование* — образуется из последовательности действий, следующих одно за другим;
2) *ветвление* (*развилка*) — реализует выбор одного из альтернативных путей алгоритма в зависимости от результатов проверки некоторого условия;
3) *цикл* — обеспечивает многократное выполнение некоторой совокупности действий, которая называется *телом цикла*.

Линейная структура алгоритма (следование)

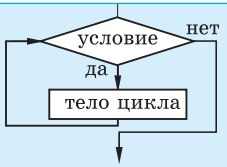
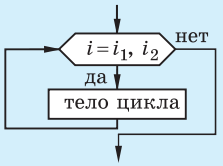
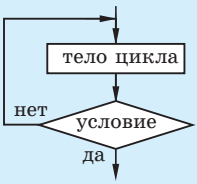
Учебный алгоритмический язык	Язык блок-схем
действие 1 *действие 2* ... *действие N*	

Структуры ветвления

Учебный алгоритмический язык	Язык блок-схем
1) если — то	
если *условие* **то** *действие 1* **всё**	

Учебный алгоритмический язык	Язык блок-схем
2) если — то — иначе	
если *условие* **то** *действие 1* **иначе** *действие 2* **всё**	
3) выбор	
выбор **при** *условие 1*: *действие 1* **при** *условие 2*: *действие 2* ... **при** *условие N*: *действие N* **всё**	
4) выбор — иначе	
выбор **при** *условие 1*: *действие 1* **при** *условие 2*: *действие 2* ... **при** *условие N*: *действие N* **иначе** *действие N* + 1 **всё**	

Структуры цикла

Учебный алгоритмический язык	Язык блок-схем
1) *цикл с предусловием* (*цикл* **пока**)	
нц **пока** условие тело цикла (последовательность действий) **кц**	
2) *цикл с параметром* (*цикл* **для**)	
нц **для** i **от** i_1 **до** i_2 тело цикла (последовательность действий) **кц**	
3) *цикл с постусловием* (*цикл* **до**)	
нц тело цикла (последовательность действий) **до** условие **кц**	

1.6.2. Вычислимость. Эквивалентность алгоритмических моделей

Машина Тьюринга

Машина Тьюринга представляет собой бесконечную ленту, разделенную на ячейки, и управляющее устройство, которое может

находиться в одном из множества (конечного) состояний. Устройство может перемещаться влево и вправо по ленте, читать и записывать в ее ячейки символы некоторого (конечного) алфавита.

Алгоритм для машины Тьюринга записывается в виде правил перехода для управляющего устройства машины. Каждое правило сопоставляет состоянию устройства и значению в текущей ячейке ленты некоторое действие.

Машина Тьюринга задается конечным алфавитом, множеством состояний и правилами перехода. Правила записываются в виде

$$q_1\, a_1 \rightarrow q_2\, a_2\, D,$$

где q_1 и a_1 — текущее состояние и наблюдаемый символ соответственно; q_2 и a_2 — состояние, в которое перейдет управляющее устройство, и символ, который оно запишет в ячейку; D — движение управляющего устройства, которое может принимать значения L (налево), R (направо), N (остаться на месте).

Пример. Пусть на ленте написано число в двоичной системе счисления, «головка» находится в конце числа. Записать алгоритм прибавления единицы к числу.

Решение. Начальное состояние машины — q_1, завершающее — q_0. Правила будут такими:

$q_1\ 1 \rightarrow q_1\ 1\ L$
$q_1\ 0 \rightarrow q_2\ 1\ R$
$q_2\ 1 \rightarrow q_2\ 0\ R$
$q_2\ 0 \rightarrow q_0\ 0\ N$

Правила удобно записывать в виде таблицы, в которой по горизонтали отмечены состояния, по вертикали — алфавит, а в ячейках — необ-

ходимые действия. Так, правило записывается в виде таблицы:

	q_0	q_1	q_2
0		q_2 1 R	q_0 0 N
1		q_1 1 L	q_2 0 R

Машина Поста

Машина Поста эквивалентна машине Тьюринга по функциональности, но проще устроена. Машина состоит из бесконечной ленты, разделенной на ячейки, и управляющей каретки. Алфавит машины Поста состоит только из 1 и 0, и у машины нет состояний.

Алгоритм для каретки записывается в виде последовательности строк и имеет всего 6 команд:

L — перейти к ячейке слева;

R — перейти к ячейке справа;

1 — поставить в ячейке символ 1;

0 — поставить в ячейке символ 0 (т. е. удалить символ 1);

S — завершить работу машины;

? p_1, p_2 — если в текущей ячейке стоит 0, то перейти к выполнению строки p_1, иначе к строке p_2.

Пример. Записать алгоритм прибавления 1 к числу в двоичной системе счисления на машине Поста.

Решение.

1. ? 4, 2
2. L
3. ? 4, 2
4. 1
5. R
6. ? 10, 7
7. 0
8. R
9. ? 10, 7
10. S

Нормальные алгоритмы Маркова

Алгоритм Маркова составляется из конечного упорядоченного множества подстановок — правил преобразования слова заменой его фрагментов.

Правила преобразования (подстановки) выполняются поочередно над исходной (непустой) строкой символов, пока не будет выполнено специальное завершающее правило. Каждый раз выполнение начинается с самой первой возможной подстановки.

Заключительные формулы подстановки помечаются символом «*», который не входит в алфавит (в противном случае используют другой символ). Если заключительное правило не задано, алгоритм завершается, когда нельзя будет применить ни одно из правил (ни одна подстановка больше не подходит).

Пример. Выполнить преобразование слова *abacaba* с помощью нормального алгоритма Маркова (НАМ):

1. $aba \to bab$ 2. $acab \to *b$
3. $cb \to aac$ 4. $ca \to cb$

Решение. Находим самое верхнее правило, которое можно применить к данному слову, и применяем его. Это подстановка (1) — получаем *babcaba*. В полученном слове *babcaba* снова возможна подстановка (1) → *babcbab* → (3) *babaacab* → (1) *bbabacab* → (1) *bbbabcab* → (4) *bbbabcbb* → (3) *bbbabaacb* → (1) *bbbbabacb* → (1) *bbbbabcb* → (3) *bbbbabaac* → (1) *bbbbbabac* → (1) *bbbbbbabc*. Завершающее правило (2) не было применено ни разу, и алгоритм завершил работу из-за невозможности выполнить ни одну заданную подстановку.

Ответ: *bbbbbbabc*.

1.6.3. Построение алгоритмов и практические вычисления

Циклы

Пример 1. Составить блок-схему алгоритма вычисления суммы знакопеременного ряда $S = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$ с заданной точностью ε. Представить алгоритм в виде псевдокода.

Решение. Используем обозначения:

S — частичная сумма ряда (стартовое значение равно 0);

ε — точность вычисления;

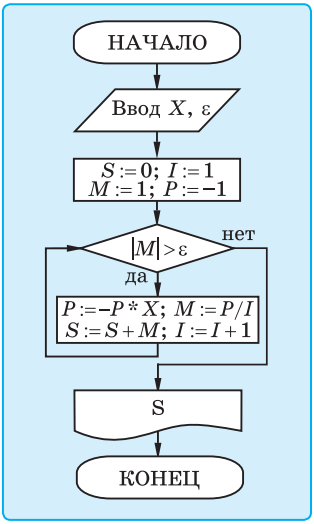
i — номер очередного слагаемого;

m — значение очередного слагаемого;

p — числитель очередного слагаемого.

Требуемая точность будет достигнута, когда очередное слагаемое станет по абсолютной величине меньше ε.

Составим блок-схему алгоритма (см. рисунок).



На псевдокоде алгоритм можно записать следующим образом:

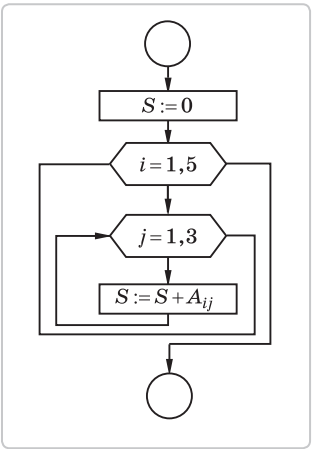
алг Сумма (**арг вещ** x, ε, **рез вещ** S)
дано | $0 < x < 1$
надо | $S = x - x^2/2 + x^3/3 - x^4/4 + \ldots$
нач цел i, **вещ** m, p

```
ввод x, ε
S := 0;  i := 1
m := 1;  p := −1
нц пока abs(m) > ε
    p := −p * x
    m := p/i
    S := S + m
    i := i + 1
кц
вывод S
кон
```

Пример 2. Вычислить сумму элементов для заданной матрицы $A(5,3)$ (таблицы из 5 строк и 3 столбцов).

Решение. Алгоритм решения задачи на псевдокоде:

алг Сумма **(арг вещ таб** $A(1{:}5,1{:}3)$**, рез вещ** S**)**



дано | **таб** A
надо | $S = A_{1,1} + A_{1,2} + A_{1,3} + \ldots + A_{5,1} + A_{5,2} + A_{5,3}$

```
нач цел i, j, вещ S
    S := 0
    нц для i от 1 до 5
      нц для j от 1 до 3
        S := S + A[i, j]
      кц
    кц
      вывод S
кон
```

Основная часть блок-схемы расчета суммы элементов матрицы отображена на рисунке.

Массивы

Массив — упорядоченный набор данных, имеющий имя и состоящий из фиксированного числа однотипных элементов. Каждый элемент массива снабжен индексом.

Индекс — порядковый номер элемента, определяющий его положение в массиве. Таким образом, элементы массива идентифицируются с помощью имени массива и с помощью своих индексов.

Количество элементов массива называется *размерностью* массива.

Массив может быть *одномерным* (вектор) и *многомерным* (матрица).

Пример 1. В массиве a каждый элемент равен 0 или 1. Заменить все нули единицами и наоборот.
Решение. Достаточно одного оператора присваивания в теле цикла:
$a[i] := 1 - a[i]$

Пример 2. В массиве каждый элемент равен 0, 1 или 2. Переставить элементы массива так, чтобы сначала располагались все 0, затем все 1 и, наконец, все 2. Дополнительный массив не использовать.
Решение. Можно не переставлять элементы массива, а подсчитать количество 0, 1, 2 и заново заполнить массив требуемым образом.

```
алг Сумма (арг цел n, рез арг вещ таб A[1:n])
    дано | массив A содержит нули, единицы и двойки
    надо | упорядочить массив по возрастанию
    нач цел i, k1, k2
            k1 := 0; k2 := 0
            нц для i от 1 до n
                  если A[i] = 0
                      то k1 := k1 + 1          всё
                  если A[i] = 1
                      k2 := k2 + 1 всё
            кц
            нц для i от 1 до k1
                  A[i] := 0
            кц
            нц для i от k1 + 1 до k1 + k2
                  A[i] = 1
            кц
```

```
        нц для i от k1 + k2 до n
            A[i] = 2
        кц
кон
```

Пример 3. Даны два n-элементных массива x и Y одного типа. Поменять местами все x_i и Y_i, $(i = 1...n)$, не используя промежуточные величины.

Решение. Обмен можно выполнить в цикле для всех i от 1 до n с помощью серии из трех операторов присваивания:

$x[i] := x[i] + y[i]$
$y[i] := x[i] - y[i]$
$x[i] := x[i] - y[i]$.

Пример 4. Найти сумму элементов одномерного массива $A(n)$.

Решение. Для суммирования положительных элементов массива вместо оператора $S := S + A[i]$ необходимо записать:

```
если A[i] > 0
    то S := S + A[i]
    всё
```

На псевдокоде алгоритм расчета суммы выглядит следующим образом:

```
алг Сумма (арг цел n, арг вещ таб A[1:n], рез вещ S)
    дано  | массив A
    надо  | найти сумму элементов массива
    нач
        цел i
        S := 0
        нц для i от 1 до n
            S := S + A[i]
        кц
    кон
```

Поиск в массиве, сортировка массива

Поиск — одна из важных невычислительных задач. Проведение поиска в упорядоченном и неупорядоченном массивах отличается. В неупорядоченном массиве, если нет никакой дополнительной информации об элементе поиска, его выполняют с помощью последовательного просмотра всего массива. Такой поиск называют *линейным*.

В упорядоченном массиве поиск можно значительно ускорить, применяя *метод половинного деления*, или *бинарный поиск*.

Сортировка массива — это перераспределение элементов массива в заданном порядке. Основная цель сортировки — облегчить последующий поиск. Методы сортировки условно разделяют по главной идее алгоритма, а реализуются они целым семейством алгоритмов.

Метод сортировки массива с помощью обмена. При использовании этого метода сортировки сравниваются соседние элементы массива и при необходимости меняются местами до тех пор, пока массив не будет полностью упорядочен. Повторные просмотры массива каждый раз сдвигают наименьший (наибольший) элемент оставшейся части массива к его концу. Метод широко известен под названием «*пузырьковая сортировка*». Большие (меньшие) элементы массива, подобно пузырькам, «всплывают» на соответствующую позицию. Основной фрагмент программы содержит два вложенных цикла, причем внутренний цикл удобнее выбрать с убывающим шагом.

Алгоритм пузырьковой сортировки массива $A(n)$ по возрастанию на псевдокоде:

алг Обменная_сортировка (**арг цел** n, **арг рез вещ таб** $A[1{:}n]$)

```
дано | A — массив размерности n
надо | упорядочить массив по возрастанию
нач
  цел i, j
  вещ Tmp
  нц для i от 2 до n
    нц для j от n до 1
      если A[j] < A[j − 1]
        то Tmp := A[j]; A[j] := A[j − 1];
        A[j − 1] := Tmp
      всё
    кц
  кц
кон
```

Метод сортировки массива с помощью прямого включения (вставки). На каждом шаге этого метода массив разделен на две части: левую, уже отсортированную, и правую, еще не отсортированную. Первый элемент правой части вставляется в левую часть массива так, чтобы левая часть осталась отсортированной. В результате отсортированная часть увеличивается на один элемент, а неотсортированная — на один элемент уменьшается. Таким образом, на каждом шаге этого алгоритма приходится выполнять две операции: поиск позиции для вставки элемента и собственно его вставку с последующим сдвигом на одну позицию вправо от элементов отсортированной части. Поскольку операции сравнения и перемещения чередуются друг с другом, этот способ сортировки часто называют *просеиванием*, или *погружением*.

Алгоритм сортировки вставкой массива $A(n)$ по возрастанию на псевдокоде:

```
алг Сортировка_вставкой (арг цел n, арг рез вещ таб A[1:n])
        дано | A — массив размерности n
        надо | упорядочить массив по возрастанию
        нач
            цел i, j
            вещ Tmp
            нц для i от 2 до n
                Tmp := A[i]; j := i − 1
                нц пока j ≥ 1 и A[j] > Tmp
                    A[j + 1] := A[j]
                    j := j − 1
                кц
                A[j + 1] := Tmp
            кц
        кон
```

Метод сортировки массива с помощью прямого выбора. При сортировке этим методом сначала выбирается наименьший (наибольший) элемент массива и меняется местами с первым. Затем выбирается наименьший (наибольший) среди оставшихся $(n - 1)$ элементов и меняется местами со вторым и т. д. до тех пор, пока не останется один наибольший (наименьший) элемент. Сортировка осуществляется с помощью двух вложенных циклов.

Алгоритм сортировки выбором массива $A(n)$ по возрастанию на псевдокоде:

```
алг Сортировка_выбором (арг цел n, арг рез вещ таб A[1:n])
        дано | A — массив размерности n
        надо | упорядочить массив по возрастанию
        нач
            цел i, k
            вещ Min
```

```
нц для i от 1 до n − 1
    Min := A[i];  k := i
    нц для j от i + 1 до n
        если Min > A[j]
            то Min := A[j];  k := j
            всё
    кц
    A[k] := A[i];  A[i] := Min
кц
кон
```

Вспомогательные алгоритмы: процедуры и функции

Подпрограмма — это именованный, логически законченный алгоритм, который оформляется специальным образом и может многократно использоваться при решении более общей задачи. Различают два вида подпрограмм — *процедуры* и *функции*.

Функция — это алгоритмически вычисляемое выражение, которое присваивается идентификатору функции.

Процедуры в общем случае не возвращают результат через свой идентификатор.

Подпрограммы бывают *стандартными* и *пользовательскими*. *Стандартные*, или *встроенные*, подпрограммы входят в состав языка программирования. Эти подпрограммы определенным образом организованы в специальные библиотеки. С помощью встроенных процедур и функций выполняются операции ввода/вывода, работа с файлами, обработка символьной информации, вычисление различных математических функций и т. п.

Подпрограммы, написанные пользователем, называются *пользовательскими*.

1.7. Языки программирования

Программа — это алгоритм, записанный на языке программирования.

Язык программирования — формальный язык, на котором записывается алгоритм для исполнения компьютером. При записи алгоритма на языке программирования должны строго выполняться все правила языка.

Для записи программ используется конечный набор символов, составляющих *алфавит языка программирования*. Программа записывается в виде последовательности символов из этого алфавита. Как и в естественных языках, правильность построения программы из символов алфавита можно проверить, используя правила синтаксиса.

Синтаксис языка программирования — это набор правил, которые определяют способы построения правильных программ из символов алфавита. Зная синтаксис языка, можно построить алгоритм, определяющий, является ли данный текст правильной программой или нет. Этот алгоритм позволяет компьютеру проверять синтаксическую правильность вводимых в него программ.

Семантика языка программирования — это набор правил, по которым исполнитель выполняет программы на этом языке. Пользуясь семантикой языка, можно однозначно определить результат выполнения программы с заданными входными данными.

Структура программы

1. Программа на языке **Pascal** всегда состоит из двух частей: раздела описания и выполняемых команд.

```
program имя программы;
  uses Список используемых библиотек;
  label Список меток;
  const Определение констант;
  type Описание типов;
  var Описание переменных;
  Определение используемых
  процедур;
  Определение используемых
  функций;

  begin
       ...
       Основной  блок программы
       ...
  end
```

описание данных

описание действий (обязательная часть)

2. На языке программирования **QBASIC** программы не имеют заголовка. Текст программы может начинаться непосредственно с какого-либо оператора. Но чаще в первые строки программы выносят название программы или ее назначение. В начале таких строк ставится служебное слово *REM*. Строки, начинающиеся со слова *REM*, не исполняются.

Программа на QBASIC состоит из строк операторов ввода и вывода, операторов ветвления, циклов и др. Иногда строки программы нумеруют числами так, чтобы исполнение программы осуществлялось в порядке возрастания номеров. Шаг нумерации выбирают кратным, например, 5, 10, 100. Это позволяет легко вставлять новые

строки в программу, не нарушая общей нумерации. Номера строк могут использоваться как *метки* в случаях, когда необходимо нарушить построчную последовательность выполнения команд и организовать переход к оператору определенной строки.

Текст программы может заканчиваться служебным словом *END* или строкой *N END*, где *N* — номер строки (метка).

Операторы записываются последовательно друг за другом, допускается запись нескольких операторов в одной строке через разделитель «:» (двоеточие). При передаче управления на строку, содержащую несколько операторов, метка относится к самому левому оператору данной строки, на остальные операторы передать управление невозможно.

1.7.1. Типы данных

Pascal

Простые типы данных

Простые типы данных языка Pascal включают скалярные и ограниченные типы.

Скалярные типы:

- целые;
- вещественные;
- символьный;
- логический

Ограниченные типы:

- перечисляемый;
- ограниченный (диапазон)

Целые типы данных

Тип	Объем памяти	Диапазон значений
byte	1 байт	0 ... 255
word	2 байта	0 ... 65535
shortint	1 байт	−128 ... 127
integer	2 байта	−32768 ... 32767
longint	4 байта	-2^{31} ... $2^{31} - 1$

Арифметические операции и функции для данных целого типа:

- +, −, * — сложение, вычитание, умножение;
- ABS(N) — абсолютная величина N;
- SQR(N) — квадрат числа N;
- SQRT(N) — корень квадратный из N (должно выполняться $N > 0$);
- ODD(N) — проверка N на четность.

Вещественные типы данных

Тип	Диапазон значений
real	$2{,}9 \cdot 10^{-39}$ ... $1{,}7 \cdot 10^{+38}$
single	$1{,}5 \cdot 10^{-45}$ ... $3{,}4 \cdot 10^{+38}$
double	$5{,}0 \cdot 10^{-324}$ ... $1{,}7 \cdot 10^{+308}$
extended	$3{,}4 \cdot 10^{-4932}$ ... $1{,}1 \cdot 10^{+4932}$
comp	$-9{,}2 \cdot 10^{+18}$ ... $9{,}2 \cdot 10^{+18}$

Функции для данных вещественного типа:

- ABS(X) — абсолютное значение X;
- ARCTAN(X) — арктангенс X;
- COS(X) — косинус X;
- SIN(X) — синус X;
- LN(X) — натуральный логарифм X (должно выполняться $X > 0$);

- SQR(X) — квадрат X;
- SQRT(X) — корень квадратный из X (должно выполняться $X > 0$);
- EXP(X) — e^X;
- FRAC(X) — дробная часть X;
- INT(X) — целая часть X;
- PI() — значение числа π;
- TRUNC(X) — получение целой части вещественного числа X;
- ROUND(X) — округление вещественного числа X.

Символьный тип данных

Значением символьного типа **char** (занимает 1 байт памяти) может быть один из 255 упорядоченных элементов множества всех символов, представленных кодами таблицы **ASCII** (*American Standard Code for Information Interchange* — стандартный американский код для обмена информацией).

Символы записываются в одинарных апострофах (например '*a*', '@', '*1*'), которые при выводе на экран не отображаются.

Допускается использование записи символа через его внутренний код, который указывается после символа #. *Внутренний код* — это порядковый номер символа в таблице кодов. Например, символ '*a*' соответствует записи #97; символ '*z*' соответствует записи #90.

Функции для данных символьного типа (в скобках указаны аргумент функции и через двоеточие тип аргумента):

- CHR(x: **byte**): **char** — возвращает символ с заданным ASCII-кодом;
- ORD(c: **char**): **byte** — возвращает ASCII-код указанного символа;

- PRED(c: **char**): **char** — выдает символ, предшествующий символу *c*;
- SUCC(c: **char**): **char** — выдает символ, последующий за символом *c*;
- не определены значения SUCC(#255) и PRED(#0).

Логический тип данных

Значением логического типа **boolean** является множество из двух упорядоченных элементов FALSE и TRUE. Переменным логического типа могут присваиваться только эти два значения. Об их значениях известно:

TRUE > FALSE, SUCC(FALSE) = TRUE,
ORD(FALSE) = 0, ORD(TRUE) = 1,
PRED(TRUE) = FALSE.

Перечисляемые типы данных

Перечисляемые типы данных задаются непосредственно перечислением всех значений, которые может принимать переменная этого типа. Отдельные значения указываются через запятую, а весь список заключается в круглые скобки:

```
type
    color = (red, yellow, green, black);
    animals = (cat, dog, horse);
    days = (Monday, Sunday, Tuesday, Friday);
```

Количество всех возможных значений перечисляемого типа конечно, значит, такие типы, как целые, символьный и логический, тоже можно считать перечисляемыми.

Для переменных перечисляемого типа можно использовать функции SUCC(x) и PRED(x). Например: DEC(5) = 4; PRED('b') = 'a'. Не определены значения SUCC(последний элемент перечисления) и PRED(1-й элемент перечисления).

Ограниченные типы данных (диапазоны)

Содержат значения только из ограниченного поддиапазона некоторого базового типа (любого целочисленного, символьного или введенного программистом перечисляемого типа). Для задания ограниченного типа надо указать границы (сначала меньшую, потом большую) через разделитель (две точки):

```
type
    digits = 1...8;
    symbols = 'b'...'m';
```

Структурированные типы данных

Структурированные типы данных:

- массив;
- строка;
- запись;
- множество;
- файл.

Массив — упорядоченная структура однотипных данных, хранящая их последовательно. Массив обязательно имеет размеры, определяющие, сколько элементов хранится в структуре. Каждый элемент массива снабжен индексом. Тип указывается с помощью конструкции

type

имя типа = **array** [*тип индексов*] **of** *тип элементов*.

Тип индексов массива — любой порядковый тип, кроме **longint**. Можно использовать ограниченные (диапазон) и перечисляемые типы.

Тип элементов массива — любой тип, кроме файлов и объектов.

Строки являются одним из часто используемых типов языка Pascal. Строковый тип обоб-

щает понятие символьных массивов, позволяя динамически изменять их длину.

Для определения строкового типа используется служебное слово **string**, за которым в квадратных скобках указывается максимальная длина строки:

type
имя типа = **string**[*длина строки*];

Максимально возможная длина строки — 255 символов. Указание максимальной длины может опускаться в объявлениях.

Запись — структурированный тип данных, состоящий из фиксированного числа компонентов одного или нескольких типов.

Описание записи начинается идентификатором **record**, а завершается служебным словом **end**. Между ними указывается список компонентов, называемых *полями*, с указанием идентификаторов полей и типа каждого поля:

type
имя типа = **record**
имя поля 1: *тип поля 1*;
имя поля 2: *тип поля 2*;
...
имя поля N: *тип поля N*;
end;

QBASIC

Типы данных языка QBASIC:
- целые;
- вещественные;
- логические;
- символьные;
- массивы;
- файл.

Для того чтобы показать, что используется переменная определенного типа данных, следует к имени этой переменной добавить знак этого типа, например % для целого типа или $ для символьного (G%, A$ и т. п.).

Целые типы данных

Тип	Диапазон значений
% — INTEGER	−32768 ... 32767
& — LONGINT	−2147483648 ... 2147483647

Арифметические операции и функции для данных целого типа:

- +, −, * — сложение, вычитание, умножение;
- ^ — возведение в степень;
- ABS(N) — абсолютная величина N;
- A\B — целая часть от деления;
- A MOD B — остаток от деления;
- FIX(X) — получение целой части вещественного числа X;
- CINT(X) — округление до целого вещественного числа X;
- CLNG(X) — округление до длинного целого вещественного числа X;
- INT(X) — получение наибольшего целого числа, меньшего или равного X.

Вещественные типы данных

Тип	Диапазон значений
! — REAL	$-2{,}9 \cdot 10^{-38} \ldots 1{,}7 \cdot 10^{38}$
# — DOUBLE	$-2{,}9 \cdot 10^{-38} \ldots 1{,}7 \cdot 10^{38}$

Функции для данных вещественного типа:

- ABS(X) — абсолютное значение X;
- ATN(X) — арктангенс X;
- COS(X) — косинус X;
- SIN(X) — синус X;
- TAN(X) — тангенс X;
- EXP(X) — e^X;
- LOG(X) — натуральный логарифм X;
- RND(X) — получение случайного числа;
- SQR(X) — квадратный корень из X;
- CDBL(X) — представление числа X с двойной точностью;
- CSNG(X) — представление числа X с одинарной точностью;
- CINT(X) — округление для целого числа;
- INT(X) — определение наибольшего целого, не превосходящего X;
- SGN(X) — определение знака величины X.

Логический тип данных

Данные логического типа имеют значения *истина* **(1) либо** *ложь* **(0). Над ними возможны следующие операции:**

- NOT — отрицание;
- OR — объединение, или логическое сложение;
- AND — пересечение, или логическое умножение;
- XOR — исключающее «ИЛИ», или сложение по модулю два;
- EQV — эквивалентность;
- IMP — импликация, или следование.

Символьный тип данных

Значениями переменной символьного типа **STRING** (**$**) являются цепочки символов длиной не более 32 767 символов. Чтобы показать, что переменная символьного типа, к ее имени

приписывают знак $ (например, A$). В тексте программы значения символьных переменных заключают в апострофы.

Операции и функции для данных символьного типа:

- + — операция *конкатенации*: сцепление (соединение) двух строк в одну (например, 'мото' + + 'цикл' = 'мотоцикл');
- ASC(X$) — преобразование символа X$ в его десятичный код (коды всех строчных букв меньше кодов всех прописных);
- CHR$(N) — преобразование кода *N* в символьное представление;
- RIGHT$(X$, N) — выделение *N* символов, начиная с самого правого символа в символьном выражении X$;
- LEFT$(X$, N) — выделение *N* символов, начиная с самого левого символа в символьном выражении *X$*;
- MID$(X$, N, M) — выделение *M* символов, начиная с *N*-го символа в символьном выражении X$ (*M* может быть опущено);
- SWAP X$, Y$ — обмен символьными выражениями X$ и Y$ (используется как оператор);
- STRING$(N, X$) — формирование строки из *N* одинаковых символов;
- SPACE$(N) — формирование строки из *N* пробелов;
- OCT$(N) — перевод десятичных чисел в восьмеричное исчисление;
- HEX$(N) — перевод десятичных чисел в шестнадцатеричное исчисление;
- LEN(N$) — определение длины символьного выражения;
- STR$(X$) — перевод числа в символьную форму с резервированием перед символьным выражением одного пробела для знака;

- VAL(X$) — преобразование строки в числовое представление. Если преобразование невозможно — результат 0;
- INSTR (N, X$, Y$) — поиск подстроки Y$ в строке X$, начиная с *N*-го символа (*N* можно опустить).

Тип данных массив

Массив представляет собой заранее известное количество однотипных компонентов, снабженных индексами. Массив может быть *одномерным* или *многомерным*.

Чтобы задать массив, необходимо использовать зарезервированное слово **DIM**, после которого указать имена описываемых массивов и в круглых скобках список из максимальных значений их индексов (по числу измерений). Описание массива должно быть произведено до первого обращения к его элементам.

Нумерация индексов массива начинается с нуля (т. е. минимальное значение индекса равно нулю). Можно изменить минимальное значение индексов массива с помощью оператора **OPTION BASE**. Он должен быть указан до объявления массивов в виде:

OPTION BASE *n*,

где *n* равно единице или нулю.

Оператор **ERASE** отменяет объявление массивов, сделанных оператором **DIM**:

ERASE *список имен массивов*

С помощью оператора **ERASE** во время выполнения программы может быть очищена память, занимаемая ненужными массивами. Это позволяет динамически изменять характеристики массивов (число измерений и значения максимальных индексов).

Файловый тип

Файловый тип представляет собой последовательность связанных между собой однотипных компонентов — *записей*.

В QBASIC имеется две категории файлов, различающихся допустимыми методами работы:

- последовательные;
- с произвольным доступом.

В QBASIC существует специальный оператор для описания именованной константы:

CONST *имя переменной = константа*

Пример: **CONST** F2 = 60, N% = 15

1.7.2. Основные конструкции языка программирования. Система программирования

Pascal

Тело программы на языке Pascal представляет собой последовательность операторов, разделяемых точкой с запятой.

Процедуры ввода/вывода в языке Pascal

- *ввод данных*:
  read(f, X); или **read**(f, X1, X2, … , Xn);
  readln(f, X) или **readln**(f, X1, X2, … , Xn);
- *вывод данных*:
  write(f, X) или **write**(f, X1, X2, … , Xn);
  writeln(f, X) или **writeln**(f, X1, X2, … , Xn);

Простые операторы языка Pascal

- *Оператор присваивания* «:=». Предписывает выполнить выражение, заданное в его правой части, и присвоить результат переменной,

идентификатор которой расположен в левой части. Переменная и выражение должны быть совместимы по типу.

- *Оператор безусловного перехода.* Выполняет переход к конкретному оператору по его метке. Меткой можно пометить любой оператор в программе, но все используемые в программе метки должны быть описаны в разделе **label**:
  label *имя метки 1*, *имя метки 2*, ... *имя метки N*;
  Синтаксис оператора безусловного перехода:
  goto *имя метки*;
- *Пустой оператор.* Не содержит никаких символов и не выполняет никаких действий. Обычно используется для организации различных переходов в программе.
- *Оператор обращения к процедуре.* Для обращения к процедуре необходимо указать ее имя и, если требуется, список фактических параметров.

Структурированные операторы языка Pascal

Оператор	Форма записи
Составной оператор	**begin ... end**
Условный оператор	**if ... then**
	if ... then ... else
Оператор выбора	**case ... of**
Операторы цикла	**for ... do**
	while ... do
	repeat ... until
Оператор над записями	**with**

QBASIC

Программа на языке QBASIC представляет собой последовательность операторов, размещаемых в отдельных строках, длина которых не превышает 255 байт.

Операторы языка подразделяются на *выполняемые* и *невыполняемые*. Выполняемые операторы исполняют определенные операции или изменяют порядок выполнения операторов программы. К невыполняемым относятся операторы управления, такие как оператор конца программы **END**, объявления массивов **DIM**, комментария **REM**.

Операторы ввода/вывода языка QBASIC:

- **INPUT**. Производит ввод данных, присваивая значения соответствующим переменным во время ввода. Например, при выполнении оператора ввода трех целых чисел

  INPUT A%, B%, C%

на экране появится знак вопроса, и необходимо будет ввести с клавиатуры числа через запятую, а затем нажать клавишу Enter;

- **DATA. Содержит данные для программы.** Этот оператор может располагаться в любом месте программы, но обычно его помещают в ее конце;
- **READ**. Выполняет присваивание значений строки **DATA** переменным в строке **READ**. Типы переменных, использованные в операторе **READ**, должны соответствовать значениям в строке **DATA**, например:

  READ A, B%, S$
  READ S1$
  DATA 23.5, 33, "123", 55
- **PRINT**. Выводит данные на экран дисплея. В списке выражений оператора можно использовать в качестве разделителей запятую или точку

с запятой. Запятая задает так называемый «зонный» вывод данных, где под «зоной» понимается расстояние (колонки) в 14 символов. При использовании в качестве разделителя точки с запятой данные выводятся последовательно, друг за другом. Точка с запятой в конце списка выражений отменяет символы «возврат каретки» и «перевод строки». Например, оператор

```
PRINT "A", "B", "C"
```

выведет на экран 3 символа в виде: A B C

- **Конструкция TAB(*n*)** задает позицию, с которой следует начинать вывод данных (*n* — номер позиции: число или арифметическое выражение). Например, оператор

```
PRINT TAB(12); "Сумма"
```

начнет вывод с 12-й позиции.

- Функция **SPC(*n*)** добавляет *n* пробелов. Так, оператор

```
PRINT "Количество"; SPC(4); "Сумма"; SPC(4); "Цена"
```

выполнит вывод, при котором между соседними словами будет по 4 пробела.

Каждый оператор **PRINT** начинает вывод с новой строки.

Процедуры **INPUT** и **PRINT** могут выполнять операции ввода и вывода только для отдельных элементов массива, но не для массива в целом. Для этого используется оператор цикла **FOR**. Например, ввод массива можно организовать следующим образом:

```
INPUT "Введите размер массива N ="; N%
DIM MAS(N%)
FOR I = 1 TO N%
  PRINT "MAS("; I; ") =";
  INPUT MAS(I)
NEXT I
```

Простые операторы языка QBASIC

- *Оператор присваивания* (= или **LET**). С помощью этого оператора какой-либо переменной присваивается значение:

```
F% = 5
ST$ = 'QWERTY'
LET D = (A + B + C) / C
```

- *Оператор безусловного перехода* **GOTO** *N*. Этот оператор нарушает нормальное выполнение программы и переводит выполнение к строке с указанным номером *N*:. Современный стиль программирования не рекомендует использовать этот оператор, т. к. он затрудняет чтение и отладку программы.

Блочные операторы языка QBASIC

Оператор	Формы записи

Условный оператор **IF**
Выполняет действия в зависимости от значения логического условия *A*
	1) краткая форма:
	IF *A* **THEN**
	оператор 1
	оператор 2
	...
	оператор M
	END IF

	2) полная форма:
	IF *A* **THEN**
	оператор 1
	оператор 2
	...
	оператор M
	ELSE
	оператор X
	...
	оператор Y
	ENDIF

Оператор	Формы записи
Оператор выбора **SELECT CASE** Позволяет выбрать любой вариант из допустимых значений переменной	**SELECT CASE** *параметр* **CASE** *значение параметра 1* *оператор 1* **CASE** *значение параметра 2* *оператор 2* **CASE** *значение параметра X* *оператор X* **CASE ELSE** *оператор L* **END SELECT**
Оператор безусловного цикла **FOR**	**FOR** *I=N1* **TO** *N2* **STEP** *X* *тело цикла* **NEXT** *I* где *I* — переменная цикла (должна быть числом); *N1* — начальное значение переменной цикла; *N2* — конечное значение переменной цикла; *X* — шаг, с которым изменяется переменная цикла
Операторы цикла с постусловием **DO LOOP WHILE, DO LOOP UNTIL** Проверка условия выхода из цикла осуществляется после каждого выполнения тела цикла (таким образом, этот цикл всегда будет выполнен хотя бы один раз)	1) форма 1: **DO** *тело цикла* **LOOP WHILE** *условие*
	2) форма 2: **DO** *тело цикла* **LOOP UNTIL** *условие*

Оператор	Формы записи
Операторы цикла с предусловием **DO WHILE LOOP, DO UNTIL LOOP** Проверка условия цикла проводится до начала очередной итерации	1) форма 1: **DO WHILE** *условие* *тело цикла* **LOOP**
	2) форма 2: **DO UNTIL** *условие* *тело цикла* **LOOP**
	3) форма 3: **WHILE** *условие* *тело цикла* **WEND**

1.7.3. Основные этапы разработки программ. Разбиение задачи на подзадачи

Разработка законченного программного продукта в виде компьютерной программы — длительный и трудоемкий процесс. Чтобы окончательный вариант программы работал правильно и содержал как можно меньше ошибок, программисты придерживаются **полного цикла разработки программы**, состоящего из шести базовых этапов:

1. *Постановка и анализ задачи*. Четкое определение задачи и наборов входных и выходных данных.
2. *Разработка алгоритма*. Определение зависимости между входными и выходными данными, создание процедуры их преобразования.
3. *Разработка пользовательского интерфейса*. Определение того, что пользователь должен

видеть на экране, как будут вводиться данные, где и в каком формате будут представлены выходные данные.

4. *Написание программного кода.* **Преобразование** алгоритма в компьютерную программу на языке высокого уровня.
5. *Тестирование и отладка программы. Тестирование* — прогон программы на наборе тестов, для которых известен результат, с целью проверки правильности ее работы. *Отладка* (debug) — процесс выявления и устранения ошибок в программе.
6. *Составление документации.* Подготовка документов, содержащих описание программы, включая техническое задание, блок-схемы, предположения, список входных и выходных переменных (часто совмещается с программным кодом), руководства пользователя.

При написании программы прежде всего следует четко уяснить задачу, которую должна решать программа. Затем предварительно разработанный алгоритм решения задачи записывают в виде упорядоченной последовательности команд (инструкций), т. е. составляется программа, ориентированная на определенную среду программирования.

При написании программы обязательно следует проверять, насколько она соответствует намеченной цели, т. е. выполняет ли программа для всех наборов данных то, что от нее требуется, не производит ли она каких-

Тестовые данные должны обеспечить проверку всех возможных условий возникновения ошибок

либо лишних действий. Основное внимание следует сосредоточить на предотвращении логических ошибок.

Процесс тестирования можно разделить на три этапа:

I этап

- *Просмотр.* Текст программы просматривается на предмет обнаружения явных ошибок и расхождений с алгоритмом.
- *Проверка.* При проверке программы программист по ее тексту мысленно старается восстановить тот вычислительный процесс, который определяет программа, после чего сверяет его с требуемым алгоритмом.
- *Прокрутка.* Имитация программистом выполнения программы на компьютере.

II этап

- *Отладка* — это процесс поиска и устранения ошибок в программе, производимый по результатам ее прогона на компьютере.

III этап

- *Тестирование* — это испытание, проверка правильности работы программы в целом или ее составных частей.

Как бы тщательно ни была отлажена программа, решающим этапом, устанавливающим ее пригодность для работы, является контроль программы по результатам ее выполнения на *системе тестов*. Программу условно можно считать *правильной*, если ее запуск для выбранной системы тестовых исходных данных *во всех случаях дает правильные результаты*.

2 ИНФОРМАЦИОННАЯ ДЕЯТЕЛЬНОСТЬ ЧЕЛОВЕКА

2.1. Профессиональная информационная деятельность. Информационные ресурсы

Деятельность человека, связанную с процессами получения, преобразования, накопления и передачи информации, называют **информационной деятельностью**.

Индустриальное общество сосредоточено на производстве и потреблении товаров, развитии промышленности и ее технической базы. В *информационном обществе* средством и продуктом производства становятся интеллект и знания, а его материально-технической базой — компьютерная техника, информационные технологии, системы телекоммуникационной связи.

Информатизация — это процесс, при котором создаются условия удовлетворения потребностей любого человека в получении необходимой информации.

Процесс информатизации общества является базовой составляющей пятой информационной революции.

Информационные ресурсы — отдельные документы и массивы документов в библиотеках, архивах, фондах, банках данных и других информационных системах.

По имущественной принадлежности информационные ресурсы могут быть государственной и негосударственной (граждан, организаций и общественных объединений) собственностью. Отношения по поводу права собственности на информационные ресурсы регулируются гражданским законодательством РФ.

Для обработки информации и предоставления пользователю создаются автоматизированные информационные системы работы с информационными ресурсами. Они подразделяются на 5 классов:

1) автоматизированные системы информационных ресурсов;
2) системы управления документами и документооборотом;
3) автоматизированные информационно-аналитические системы;
4) автоматизация информационной системы принятия решений:
   - системы традиционного моделирования ситуаций;
   - экспертные системы (аккумулируют знания и представляют специальные варианты принятия решений);
   - нейронные системы (информационные аналитические системы в биржевом деле, в банках, государственном управлении);
   - автоматизированные информационные системы поддержки государственных решений;
5) автоматизированные системы программируемого принятия решений.

К современным техническим средствам работы с информацией относятся не только компьютеры, но и другие устройства, обеспечивающие ее передачу, обработку и хранение:

- сетевое оборудование — модемы, кабели, сетевые адаптеры;
- аналого-цифровые и цифро-аналоговые преобразователи;
- цифровые фото- и видеокамеры, цифровые диктофоны;
- записывающие устройства (CD-R, CD-RW, DVD-RW и др.);
- полиграфическое оборудование;
- цифровые музыкальные студии;
- медицинское оборудование для УЗИ и томографии;
- сканеры в архивах, библиотеках, магазинах, на экзаменах и избирательных участках;
- ТВ-тюнеры для подачи телевизионного сигнала в компьютер;
- плоттеры и различные принтеры;
- мультимедийные проекторы;
- флеш-память, используемая также в плеерах и фотоаппаратах;
- мобильные телефоны и т. д.

Кроме персональных компьютеров, существуют мощные вычислительные системы для решения сложных научно-технических и оборонных задач, обработки огромных баз данных, работы телекоммуникационных сетей (Интернет):

- многопроцессорные системы параллельной обработки данных (управление сложными технологическими процессами);

- серверы в глобальной компьютерной сети, управляющие работой и хранящие огромный объем информации;
- специальные компьютеры для проектно-конструкторских работ (проектирование самолетов и космических кораблей, мостов и зданий и пр.).

Все перечисленные технические средства и системы предназначены для работы с информационными ресурсами в различных отраслях экономики.

Можно выделить несколько основных направлений, где информационная деятельность связана с компьютерами:

1. *Научные исследования*. Расчеты и вычисления — обязательный элемент тех научных исследований, где требуется на основании эксперимента построить гипотезу о закономерностях, проявляемых в нем. Создаются специальные автоматизированные системы для научных исследований.
2. *Создание новых изделий*. Некоторые этапы создания новых изделий могут быть автоматизированы. Системы автоматизированного проектирования (САПР) используются во всех проектных и конструкторских организациях.
3. *Управление*. Системы автоматического управления (АСУ) могут управлять процессами, для которых разработаны математические модели и методы их решения.
4. *Информационные системы (ИС), базы данных (БД)*. Основу ИС составляет банк данных, в котором хранится большая по объему информация о какой-либо области человеческих знаний. Это может быть, например, информация

об инфраструктуре города (транспорт, карта, телефоны, организации и т. д.). Использование Интернета делает доступными сведения из ИС большому числу пользователей.

5. *Обучение.* Одна из важнейших целей создания системы образовательных порталов — в явном виде и с участием специалистов сформировать профессиональную зону и механизмы поиска качественной образовательной информации.
6. *Компьютеры в издательском деле.* Компьютер используется авторами уже на самых первых этапах создания литературных, публицистических и других произведений. Затем с этими текстами работают редакторы издательств.
7. *Автоматизированное рабочее место (АРМ).* АРМ может иметь также выход в Интернет, что позволяет быстро находить необходимую информацию в сфере своей деятельности, получать и отправлять электронные письма, совершать покупки в Интернете, заказывать электронные билеты и т. д.

2.2. Экономика информационной сферы

Информационная экономика — это наука, исследующая хозяйственную деятельность человека, которая предусматривает широкое применение электронных (информационно-коммуникационных) технологий в процессах общественного производства, распределения и потребления общественных благ.

Особенностью информационной экономики является направленность на массовость и глобальный характер хозяйственного взаимодействия, а также распределения созданных благ среди потребителей в глобальном масштабе.

Главные задачи информационной экономики:

1. *На макроуровне* — выбор направления хозяйственного развития в рамках происходящих глобальных процессов, что определяется способностью хозяйственной энергии общества двигаться к новым внешним ресурсам и качественным трансформациям своей структуры.
2. *На микроуровне* — создание субъектами предпринимательства алгоритмов хозяйствования, направленных на получение хозяйственной энергии внешней среды и трансформацию ее в виде работы и создания рабочих мест.

Система информационной экономики характеризуется наличием динамически меняющихся связей, структурированных по видам экономической деятельности и обусловленных включением в систему то одних, то других элементов. При этом соблюдается условие сохранения преемственности между элементами и типами связей для обеспечения целостности правового поля.

Любая система информационной экономики в зависимости от процессов, происходящих в ней в рамках места, событий и времени, характеризуется по:

- особенностям состава элементов;
- числу элементов;
- структуре, т. е. по типу связей, объединяющих элементы.

Под элементом здесь понимается электронная единица, связанная с субъектом хозяйствования, отображающая его характеристики и обеспечивающая возможность осуществления электронного экономического взаимодействия (т. е. формального построения «алгебры» взаимодействия).

Система информационной экономики взаимосвязана с системой реального хозяйствования, обеспечивая ее структурированным отображением происходящих фактов (событий) и их эмпирическим обобщением в разрезе места, времени, видов деятельности и обстоятельств.

2.3. Информационная этика и право, информационная безопасность

Информационная этика — дисциплина, исследующая моральные проблемы, возникающие в связи с развитием и применением информационных технологий. Информационная этика связана с компьютерной этикой и философией информации.

Информационная культура предполагает знание и соблюдение юридических и этических норм и правил — порядочности, честности, точности, корректности, объективности в оценке и представлении информации.

При работе в сети следует руководствоваться определенными негласными, но общеизвестными нормами этики общения:

- *общение в чатах*. В своих сообщениях следует использовать язык, на котором общается

большинство присутствующих; здороваться при появлении в чате; придерживаться темы разговора, если чат специализируется на определенной тематике; не использовать ненормативную лексику;

- *общение по электронной почте.* В частной переписке стиль письма может быть любым, если он приемлем адресатом; в деловой переписке нужно изъясняться кратко и грамотно; подписывать письма; заполнять поле «Тема»; отвечать на письма вовремя; не пересылать большие файлы без предварительной архивации; не рассылать знакомым и незнакомым людям рекламные сообщения;
- *общение на форумах, видеоконференциях.* Общение должно быть кратким и по существу обсуждаемой проблемы; не допускается самореклама; тематика сообщений должна быть адресована всем собеседникам; запрещены высказывания расистского характера, оскорбления и некорректные замечания.

Правовые нормы в области информатики. Базовым юридическим документом является Закон Российской Федерации «Об информации, информатизации и защите информации» (1995 г.), в котором заложены юридические основы гарантий прав граждан на информацию. Закон обеспечивает защиту собственности в сфере информационных технологий, прав и свобод личности от угроз и ущерба, связанных с искажением, порчей и уничтожением персональной информации. На основе этого Закона приняты дополнительные нормативные законодательные акты: «Об авторском

праве и смежных правах», «О правовой охране программ для ЭВМ и баз данных», «О правовой охране топологии интегральных схем», «Об электронно-цифровой подписи» и т. д.

Авторское право. Основной закон — «О правовой охране программ для ЭВМ и баз данных» (1992 г.). Правовая охрана распространяется на все виды программ для ЭВМ на любом языке и в любой форме, но не распространяется на идеи и принципы, лежащие в ее основе. Для оповещения о своих правах разработчик программы использует знак охраны авторского права при первом выпуске программы в свет: символ ©, наименование правообладателя, год первого выпуска программы. В случае нелегального копирования и использования лицензионного программного обеспечения нарушитель должен выплатить разработчику компенсацию в определяемой судом сумме от 5000-кратного до 50000-кратного размера минимальной месячной заработной платы.

Электронная подпись. Законодательная основа электронного документооборота в России — закон «Об электронно-цифровой подписи» (2002 г.). По этому закону электронная цифровая подпись в электронном документе признается юридически равнозначной подписи в документе на бумажном носителе.

Информационная безопасность — это состояние защищенности информационной среды.

Защита информации представляет собой деятельность по предотвращению утечки защищаемой информации, несанкционированных и непреднамеренных воздействий на защищаемую

информацию, т. е. процесс, направленный на достижение этого состояния.

Информационная среда — это совокупность условий, средств и методов на базе компьютерных систем, предназначенных для создания и использования информационных ресурсов.

Информационные угрозы — совокупность факторов, представляющих опасность для функционирования информационной среды.

2 СРЕДСТВА ИКТ

3.1. Архитектура компьютеров и компьютерных сетей

3.1.1. Программная и аппаратная организация компьютеров и компьютерных систем. Виды программного обеспечения

Архитектура компьютера описывает его организацию и принципы функционирования его структурных элементов. Включает основные устройства компьютера и структуру связей между ними.

Основные функции по объединению всех компонентов компьютера в согласованно работающее устройство осуществляет *материнская (системная) плата*. На ней размещаются процессор, основная память, системная шина, различные контроллеры и другие устройства.

Микропроцессор — это программно-управляемое электронное цифровое устройство, обрабатывающее информацию в цифровом виде и выполняющее арифметические и логические операции. Микропроцессор совместно с микропроцессорным комплектом (*чипсетом*) также управляет работой основных узлов и блоков компьютера.

Основная память компьютера состоит из оперативной памяти (ОП, ОЗУ, оперативного запоминающего устройства) и постоянной памяти (ПП, ПЗУ, постоянного запоминающего устройства).

Оперативная память представляет собой набор микросхем, предназначенных для временного

хранения данных, когда компьютер включен (после его выключения содержимое ОЗУ теряется). В этой памяти хранятся исполняемые программы и данные.

Постоянная память — это микросхема, предназначенная для длительного хранения данных, в том числе когда компьютер выключен. К ПЗУ можно обращаться только при чтении данных и программ, но записывать в нее нельзя (запись уже произведена на заводе-изготовителе). ПЗУ хранит вспомогательные программы, которые используются многократно в процессе решения любых задач (например, проверяют состав и работоспособность компьютера, обеспечивают взаимодействие с устройствами и т. п.). Этот набор программ образует *базовую систему ввода-вывода* (*BIOS*, Basic Input/Output System), он получает управление при включении и сбросе (reset) системной платы.

Системная шина передает данные между функциональными блоками компьютера.

Контроллеры обеспечивают обмен информацией с внешними устройствами.

В состав материнской платы входит также *чипсет* (набор микросхем, обеспечивающих согласованную работу всех аппаратных средств компьютера) и ряд других подсистем, которые обеспечивают удобство и функциональность конкретной материнской платы (электропитания, мониторинга физических и электрических параметров).

Компьютер состоит из *программной части* — это все программы, которые устанавливаются на компьютер *аппаратной части* — это все то, из чего состоит компьютер.

Видеокарта (*видеоадаптер*, *графическая карта*) — устройство, преобразующее изображе-

ние, находящееся в памяти компьютера, в видеосигнал для монитора.

Внешняя память компьютера состоит из накопителей на жестких магнитных дисках (НЖМД), а также дисководов чтения и записи оптических дисков — CD и DVD.

Для работы со звуком компьютер должен быть оснащен *звуковой картой*.

Периферийные устройства — устройства, предназначенные для ввода или вывода информации: принтеры, клавиатуры, мыши, сканеры и т. д. Подсоединение их к компьютеру производится через специальные интерфейсы — *порты ввода/вывода*. По способу передачи информации порты ввода/вывода могут быть *последовательными* (информация передается последовательно) и *параллельными* (несколько битов информации передается одновременно). В настоящее время эти порты вытесняются шиной USB и беспроводными технологиями передачи информации.

Устройства ввода информации	Устройства вывода информации
▶ клавиатура; ▶ манипуляторы (мышь, трекбол); ▶ сенсорная панель; ▶ графические планшеты; ▶ сканеры; ▶ звуковая карта, микрофон; ▶ игровые манипуляторы	▶ монитор; ▶ принтеры; ▶ акустические колонки и наушники; ▶ модем

Программное обеспечение (*ПО*) компьютера — это совокупность программ и сопровождающей их документации, предназначенных для решения на компьютере определенных задач.

Классификация программного обеспечения:

- *системное ПО* — программы, которые предназначены для управления работой компьютера и вычислительной сети, их диагностики и профилактики, выполнения различных вспомогательных технологических процессов;
- *прикладное ПО* — программы, предназначенные для реализации конкретных задач по обработке данных, которые пользователь решает в ходе своей деятельности;
- *системы программирования* — комплекс программ, предназначенных для создания новых программ.

3.1.2. Операционные системы. Понятие о системном администрировании

Операционная система (*ОС*) осуществляет организацию вычислительного процесса и управление ресурсами компьютера.

Назначение ОС:

- контроль работоспособности оборудования компьютерной системы;

- выполнение процедуры начальной загрузки;
- управление работой устройств компьютера;
- управление файловой системой;
- взаимодействие пользователя с компьютером;
- загрузка и выполнение прикладных программ;
- распределение ресурсов компьютера (оперативной памяти, процессорного времени, периферийных устройств и данных) между вычислительными процессами, конкурирующими за эти ресурсы.

Сейчас на IBM-совместимые компьютеры устанавливаются ОС *Windows* и *Linux*, на персональные компьютеры Macintosh — *Mac OS*.

Функции ОС по управлению компьютером

Функции	Компоненты
Управление файловой системой	Программные модули
Выполнение команд пользователя	Командный процессор
Управление работой устройств и согласование информационного обмена с другими устройствами, а также настройка отдельных параметров устройств	Драйверы устройств
Реализация графического интерфейса	Программные модули
Обслуживание дисков, архивация файлов, обеспечение работы в сети и т. д.	Сервисные программы (утилиты)
Оперативное получение необходимой информации о работе самой операционной системы и отдельных ее модулей	Справочная система

Графический интерфейс

Современные ОС для персональных компьютеров обеспечивают взаимодействие с пользователем с помощью **графического интерфейса.** Для работы с таким интерфейсом используется мышь или другое координатное устройство ввода. Важнейший элемент графического интерфейса — *окна*. Существуют следующие *виды окон*: окна папок, окна приложений, окна документов, окна справочной системы. Отдельно выделяют *диалоговые окна,* или *диалоговые панели*, с помощью которых пользователь выполняет настройку тех или иных объектов. Диалоговые панели могут включать несколько вкладок, на которых располагаются *управляющие элементы*:

- командные кнопки;
- текстовые поля;
- списки и раскрывающиеся списки;
- счетчики;
- флажки;
- переключатели;
- ползунки и др.

Контекстное меню объекта — небольшое окно, содержащее перечень операций, разрешенных с этим объектом в данном режиме, а также позволяющее ознакомиться со свойствами объекта и при необходимости изменить их.

Файлы и файловые системы

Файл — это упорядоченная, снабженная именем совокупность данных на внешнем носителе, которую операционная система обрабатывает как единое целое. *Имя файла* состоит из двух частей: собственно имени файла и расширения (типа файла).

Для удобства хранения и поиска файлы могут объединяться в **папки** (*каталоги*). Папки могут быть вложены друг в друга, образуя многоуровневую древовидную структуру.

Файловые менеджеры — программы, которые выполняют следующие операции: копирование, перемещение, удаление, переименование файлов и папок.

Корневой каталог (*папка*) содержит вложенные каталоги (папки) первого уровня, каждый из которых также может содержать вложенные каталоги (папки) второго уровня и т. д. В папках всех уровней могут храниться и файлы.

Чтобы обратиться к какому-либо файлу, следует указать его *полное имя*, которое состоит из пути к файлу и имени файла. *Путь* начинается с имени диска, на котором записан файл, затем ставится двоеточие, обратная косая черта и далее перечисляется последовательность всех имен каталогов, которые необходимо открыть, чтобы получить доступ к файлу.

Маска представляет собой последовательность допустимых в именах файлов символов, а также символы «?» и «*». Символ «?» заменяет один допустимый символ. Символ «*» означает любую последовательность допустимых символов (произвольной длины, в том числе пустую).

Пример. В некотором каталоге хранился файл *letter1.doc*. После того как в этом каталоге создали подкаталог и переместили в него указанный файл, полное имя файла стало *D:\Work\Doc\Letter\letter1.doc*. Каково было полное имя файла до перемещения?

Решение. По полному имени файла видно, что он находится в каталоге *Letter*, следовательно, это

и есть вновь созданный каталог. Каталог *Letter* находится в каталоге с полным именем *D:\Work\Doc*. По условию задачи файл изначально хранился в том каталоге, где был создан подкаталог *Letter*. Следовательно, полное имя файла было *D:\Work\Doc\letter1.doc*.

Порядок хранения файлов на носителе информации определяется используемой файловой системой.

Файловая система — это способ организации, хранения и именования данных во внешней памяти компьютера и предоставления пользователю удобного интерфейса для работы с ними.

Структура файловой системы компьютера во многом определяет структуру операционной системы и возможности пользователя.

Для однозначного определения местоположения любого файла диск (внешняя память) должен иметь четкую физическую и логическую структуру. Эта структура создается в процессе *форматирования диска* — программного процесса разметки диска. Форматирование подразделяется на низкоуровневое (физическое) и верхнего уровня (логическое).

При *физическом* (*низкоуровневом*) *форматировании* диск разбивается на *дорожки* — концентрические окружности, пронумерованные от края к центру. Внешняя дорожка (нулевая) содержит служебную информацию. Все дорожки с одинаковыми номерами образуют *цилиндр*. Запись информации на диск идет по цилиндрам — от края (нулевого цилиндра) к центру.

Минимальным блоком информации, который может быть записан на диск или считан с него,

является *сектор* (*часть дорожки*). В начале каждого сектора имеется служебная область, за которой следуют поле данных и поле контрольного кода. В заголовке указываются номера цилиндра, головки и собственно сектора. Стандартный размер поля данных сектора для жесткого диска — 512 байт, для оптического — 2 048 байт.

При логическом форматировании (*верхнего уровня*) область данных диска разбивается на *кластеры* — группы смежных секторов. Размер кластера (число секторов, от 1 до 128) выбирается кратным степени числа 2. Каждый кластер имеет свой номер.

Кластер задает минимальный размер адресуемого пространства. Файлу на диске выделяется целое число кластеров. Если файл занимает более одного кластера, то все кластеры, занимаемые файлом, организуются в *цепочку кластеров*.

В зависимости от того, под какую файловую систему форматируется диск, создается таблица FAT или MFT.

Таблица FAT — таблица размещения файлов для файловых систем FAT, FAT16, FAT32; в ней компьютер запоминает цепочку кластеров, где размещаются файлы и папки диска. При повреждениях компьютер сам восстанавливает эту таблицу по ее копии.

FAT32 — файловая система, использующая 32-разрядную адресацию кластеров. Она поддерживает файлы размером не более 4 Гбайт. Диск, отформатированный под FAT32, содержит 3 области: загрузочную запись диска, таблицу FAT (две копии), область данных (файлы).

Загрузочная область содержит таблицу, описывающую все параметры диска, и короткую

программу, используемую в процедуре начальной загрузки ОС. Если диск готовится как системный, то это будет программа загрузки ОС; если нет — программа, которая при попытке загрузки ОС с этого диска выведет сообщение, что данный диск не является системным.

Файловая система *NTFS* хранит информацию о файлах в главной файловой *таблице MFT*, строки которой соответствуют файлам, а столбцы — их атрибутам. Максимально возможный размер файла для NTFS составляет практически ~16 Тбайт.

NTFS является стандартной файловой системой для многих последних версий ОС Windows. Для разрабатываемой версии Windows 8 разрабатывается новая файловая система.

NTFS позволяет разделить доступ к информации для разных пользователей; назначить им квоты; использовать криптографирование файлов. Система обладает повышенной устойчивостью к ошибкам за счет ведения журнала, хранящего список изменений.

Процесс разбиения файла на фрагменты при записи на диск называется *фрагментацией*. Файлы, которые располагаются в цепочках из несмежных кластеров, называются *фрагментированными*. При большой степени фрагментации замедляется доступ к файлам. *Дефрагментация* — перераспределение файлов на диске, при котором они располагаются в непрерывных областях. Для нее существуют специальные программы — *утилиты дефрагментации*. Дефрагментация чаще всего требуется для файловых систем FAT.

Существуют также *утилиты проверки диска*, которые проверяют правильность информации в таблицах распределения файлов и осуществляют поиск сбойных блоков диска.

Физические ошибки (*дефекты*) — это нарушения поверхности жесткого диска. Обычно они связаны с естественным износом диска.

Логические ошибки — это нарушения в файловой структуре.

Один из видов нарушений файловой структуры — *потерянные кластеры*: отдельные кластеры или их цепочки, помеченные как занятые, но не принадлежащие ни одному файлу. Возможны также *пересечения цепочек кластеров* — ситуация, когда два файла ссылаются на одни и те же кластеры. Для исправления подобных ошибок программы проверки используют информацию из копий таблиц размещения файлов.

Пример 1. Количество секторов в кластере — 32. Количество кластеров на жестком диске — 2^{28}. Определить емкость диска.

Решение. Учитывая, что размер одного сектора 512 байт, имеем:

$512 \cdot 32 \cdot 2^{28} = 2^9 \cdot 2^5 \cdot 2^{28} = 2^{42} = 2^2 \cdot 2^{40}$

$2^2 \cdot 2^{40}$ байт = 4 Тбайт

Ответ: 4 Тбайт.

Пример 2. Размер одного кластера жесткого диска — 1024 байта. На диск записали файлы размером 2 750 байт и 324 Кбайт. Сколько кластеров займут эти файлы?

Решение. Файл размером 2 750 байт должен занять: 2 750 : 1024 = 2,68 кластера. Поскольку файлы могут занимать только целое число кла-

стеров, следует округлить 2,68 до ближайшего большего целого числа — 3 кластера.
Файл размером 324 Кбайт займет: $324 \cdot 2^{10} : 1024 = 324$ кластера.
Вместе эти два файла займут: $3 + 324 = 327$ кластеров.
Ответ: 327 кластеров.

Системное администрирование

Системный администратор (ИТ-администратор) — сотрудник, должностные обязанности которого подразумевают обеспечение штатной работы парка компьютерной техники, сети и программного обеспечения, а также обеспечение информационной безопасности в организации.

В обязанности системных администраторов входит не только слежение за сетевой безопасностью организации, но и создание оптимальной работоспособности компьютеров и программного обеспечения для пользователей.

3.1.3. Безопасность, гигиена, эргономика, ресурсосбережение, технологические требования при эксплуатации компьютерного рабочего места

Одним из факторов обеспечения надежного функционирования средств информационно-компьютерной техники (ИКТ) является соблюдение норм техники безопасности, гигиены, эргономики и ресурсосбережения.

Гигиенические требования:

- рабочее место должно быть достаточно освещено, источник естественного или искусственного

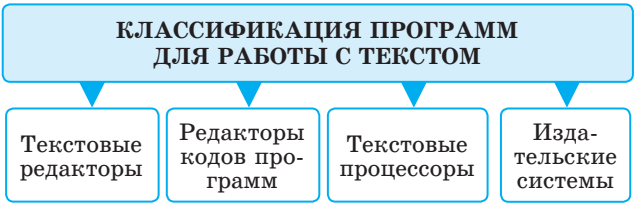
освещения не должен создавать «бликов» на экране монитора;
- необходимо проводить регулярное проветривание и влажную уборку помещения;
- следует делать перерывы через каждый час работы на компьютере;
- нужно использовать специализированную мебель, которая поможет обеспечить удобную посадку за компьютером и рациональное расположение периферийных устройств;
- рабочее место следует содержать в чистоте, не принимать пищу при работе за компьютером, чтобы не засорить и не залить технические устройства.

Требования электробезопасности:
- необходимо обеспечить надежное электропитание и заземление;
- желательно использовать источники бесперебойного питания;
- интерфейсные провода и провода электропитания должны быть расположены так, чтобы максимально снизить вероятность их повреждения неосторожными действиями пользователя;
- для присоединения периферийных устройств нельзя использовать не приспособленные для этого провода и разъемы;
- из розеток и разъемов провода следует вынимать за вилку, а не дергать непосредственно сам провод.

Ресурсосбережение обеспечивается применением энергосберегающих режимов работы оборудования (например, «спящего» режима), а также широким использованием современных компьютерных устройств, разработанных с учетом требований энергосбережения.

3.2. Технология создания и обработки текстовой информации

3.2.1. Понятие о настольных издательских системах. Создание компьютерных публикаций



Текстовые редакторы — программы для создания, редактирования и сохранения в файлах текста без возможности его форматирования.

Редакторы кодов программ — специальные текстовые редакторы, упрощающие и ускоряющие процесс написания программы (с возможностью отображения служебных слов, подсветки синтаксиса, автозавершения ввода, запуска компилятора или интерпретатора, контекстной помощи и т. п.).

Текстовые процессоры — программы для создания, редактирования и сохранения в файлах текста с широкими возможностями его форматирования и вставки сторонних объектов.

Компьютерная верстка — создание образца издания (макета) с помощью персонального компьютера и специального программного обеспечения для последующей печати в типографии или на принтере.

Под термином *компьютерная верстка* понимают не только создание макета страницы для книг и журналов. Этот термин также используется для создания макетов рекламных объявлений, упаковки, дизайна выставочных стендов и т. п.

Настольные издательские системы (*НИС*) — это программы, предназначенные для профессиональной издательской деятельности, позволяющие осуществлять компьютерную верстку широкого спектра основных типов документов.

Примерами НИС являются QuarkXPress, Adobe InDesign, Scribus, Microsoft Publisher, Apple Pages. Они позволяют:

- компоновать (верстать) текст;
- использовать всевозможные шрифты и полиграфические изображения;
- осуществлять редактирование на уровне лучших текстовых процессоров;
- обрабатывать графические изображения;
- обеспечивать вывод документов высокого качества и др.

Редактирование — это изменение содержания документа. К операциям редактирования относятся набор текста, исправление опечаток, копирование, перемещение, удаление частей текста, вставка рисунков, таблиц и других объектов.

Форматирование — это изменение внешнего вида документа и отдельных его частей.

Операции форматирования:

- изменение свойств абзацев и отдельных символов;
- оформление заголовков и подзаголовков;
- преобразование текста в список или таблицу;
- вставка колонтитулов, нумерация страниц и т. д.

Стиль форматирования — это совокупность всех параметров оформления, определяющих формат фрагмента.

3.2.2. Использование готовых и создание собственных шаблонов. Использование систем проверки орфографии и грамматики. Тезаурусы. Использование систем двуязычного перевода и электронных словарей

Шаблоны

Шаблон — это тип документа, при открытии которого создается его копия. В Microsoft Word шаблон может иметь расширение *.docx*, *.dotx* или *.dotm* (файл типа *.dotm* позволяет выполнять макросы в файле).

Как правило, шаблоны — это обычные документы, но содержащие рекомендуемые разделы, обязательный текст или специальную эмблему, а также элементы управления содержимым (например, стандартные раскрывающиеся списки) или стили форматирования. К шаблону или его разделу можно добавить защиту — например, защитить содержимое шаблона от изменений с помощью пароля.

Начать создание шаблона можно с нового документа, сохранив его как шаблон, или же создать шаблон на основе уже имеющегося документа или шаблона.

Создание шаблона из нового документа

1. Кнопка Microsoft Office → Создать → Новый документ → Создать.

2. Введите необходимый текст и другие элементы, задайте форматирование.
3. Кнопка *Microsoft Office* → *Сохранить как*. В диалоговом окне *Сохранение документа* выберите *Надежные шаблоны*. В списке *Тип файла* выберите *Шаблон Word*, задайте имя файла для нового шаблона и нажмите кнопку *Сохранить*.
4. Закройте шаблон.

Создание нового шаблона на основе существующего

1. Кнопка *Microsoft Office* → *Создать* → *Шаблоны* → *Из существующего документа*. Выберите шаблон, похожий на тот, что необходимо создать, и нажмите кнопку *Создать новый*.
2. Внесите необходимые изменения, в том числе в размеры полей и страниц, ориентацию страниц, стили и другие параметры форматирования.
3. Щелкните значок *Кнопка Microsoft Office* → *Сохранить как*. В диалоговом окне *Сохранение документа* → *Надежные шаблоны* в списке *Тип файла* выберите *Шаблон Word*, введите имя файла для нового шаблона и нажмите кнопку *Сохранить*.
4. Закройте шаблон.

Проверка орфографии и грамматики

В большинстве случаев проверка правописания достаточно проста. Нажмите клавишу *F7*, а затем используйте отобразившееся диалоговое окно или область задач для просмотра файла или элемента, над которым ведется работа.

Способы проверки и исправления ошибок в MS Word

1. *Автоматическая проверка* — Word проверяет ошибки в тексте непосредственно при его наборе. После того как слово или предложение набрано, программа подчеркивает слова с грамматическими ошибками зеленым цветом, а орфографическими — красным. С помощью контекстного меню можно выбрать варианты исправления ошибки, добавить слово в словарь пользователя или указать пропуск всех таких же слов.
2. *Проверка правописания вручную* — Word проверяет текст и отображает для каждой ошибки диалоговое окно. Такая проверка удобна, если нужно убедиться в отсутствии ошибок в некоторой части документа. Для выполнения этой проверки используется диалоговое окно *Правописание*.
3. *Автозамена* — служит для исправления ошибок, которые часто возникают при наборе (опечаток). В словарь автозамены заносятся ошибочные и правильные написания слов. При ошибочном наборе такого слова Word автоматически исправит его написание.

Словари и тезаурусы. Машинный перевод

Тезáурус — особая разновидность словарей общей или специальной лексики, в которых указаны смысловые взаимоотношения между терминами (синонимы, антонимы, более узкие и широкие понятия, отношения «часть — целое» и т. п.).

Машинный перевод — выполняемое на компьютере действие по преобразованию текста на одном естественном языке в эквивалентный по

содержанию текст на другом языке, а также результат такого действия.

Программы машинного перевода выполняют перевод на основании знаний о синтаксисе языка (правилах построения предложений, словообразования и т. п.) и использовании двуязычных словарей. В соответствии с этими правилами программа-переводчик сначала анализирует текст на одном языке, а затем конструирует его на другом языке. Такие системы называются *основанными на правилах*. К ним относятся программы PROMT, Pragma, ProLing Office.

Программы *статистического перевода* обладают доступом к многомиллионным базам текстов, переведенных ранее людьми. При переводе текста они отыскивают в базе фрагменты, целиком или частично совпадающие с искомыми, и подставляют их в перевод. Таким образом функционирует система перевода на сайте Google. Все большее распространение получают *гибридные* системы перевода, совмещающие черты систем, основанных на правилах и на статистике.

Существует множество программ — *словарных оболочек*, которые организуют поиск слов в электронных словарях. Такие словари могут быть электронными версиями печатных изданий либо же разработанными специально для использования с конкретной словарной оболочкой. Например, словарная оболочка Abbyy Lingvo позволяет искать переводы слов и выражений с 20 языков в общей сложности в 220 словарях. Существует несколько версий этой программы — все они содержат одну и ту же словарную оболочку, но разные наборы сло-

варей и направлений перевода. Среди словарей не только переводные, но и толковые, тематические, энциклопедические, лингвострановедческие, разговорники, тезаурусы и т. п.

3.2.3. Использование специализированных средств редактирования математических текстов и графического представления математических объектов

Технические тексты часто содержат *математические формулы*. MS Word позволяет вводить в текст формулы просто и удобно с помощью библиотеки математических символов. В группе *Символы* вкладки *Вставка* нужно выбрать команду *Формула / Вставить новую формулу*.

Для упорядочения представления данных часто используются *таблицы*, состоящие из строк и столбцов ячеек, которые могут содержать текст и рисунки. Однако возможности таблиц этим не ограничиваются. Они позволяют отсортировать данные, а также выполнить различные вычисления. Кроме того, с помощью таблиц нетрудно создать привлекательные макеты страниц, расположив нужным образом фрагменты текста и рисунки. Самый быстрый способ создания таблиц в MS Word — с помощью кнопки *Таблица* на вкладке *Вставка*.

Простые схематические рисунки можно выполнить средствами самой программы MS Word. Команда *Фигуры* из группы *Иллюстрации* вкладки *Вставка* позволяет создать одну из стандартных фигур. При этом на ленте открывается

дополнительное меню *Средства рисования*, содержащее вкладку *Формат*.

Команда *Диаграмма* из группы *Иллюстрации* вкладки *Вставка* позволяет разместить в документе *диаграмму*. При этом на ленте открывается дополнительное меню *Работа с диаграммами*, содержащее вкладки *Конструктор* (выбор стиля диаграммы и исходных данных для нее), *Макет* (настройка отдельных элементов диаграммы) и *Формат* (изменение размера диаграммы, задание дополнительных эффектов).

3.2.4. Использование систем распознавания текстов

Системы распознавания (*текстов, речи, изображений и т. д.*) — одна из наиболее перспективных областей применения искусственного интеллекта. Существует решение, максимально приближенное к человеческой способности читать: оно построено на принципах, сформулированных в результате наблюдений за поведением животных и человека.

Оптическое распознавание символов — механический или электронный перевод изображений рукописного, машинописного или печатного текста в последовательность кодов, которые используются для представления в текстовом редакторе. Некоторые системы оптического распознавания текста способны не только извлекать текстовые данные из цифровых изображений, но и восстанавливать исходное форматирование текста, включая изображения, колонки и другие нетекстовые компоненты.

Среди систем оптического распознавания текстов — Abbyy FineReader, CuneiForm, LiveOCR.

3.3. Технология создания и обработки графической и мультимедийной информации

Растровая графика

Растровое графическое изображение состоит из отдельных точек — **пикселей**, образующих строки и столбцы. Основные свойства пикселя — его расположение и цвет. Значения этих свойств кодируются и сохраняются в видеопамяти компьютера.

Качество изображения на экране монитора зависит от пространственного разрешения и глубины цвета.

Разрешение — величина, определяющая количество точек (элементов растрового изображения) на единицу площади (или длины).

Глубина цвета — объем памяти (в битах), используемой для хранения и представления цвета при кодировании одного пикселя растровой графики или видеоизображения.

Для графических изображений могут использоваться различные *палитры* — наборы цветов. Количество цветов N в палитре и количество информации I, необходимое для кодирования цвета каждой точки, связаны соотношением:

$$N = 2^I.$$

Например, для черно-белого изображения палитра состоит из двух цветов. Можно вычислить, какое количество информации необходимо, чтобы закодировать цвет каждой точки:

$$2 = 2^I \Rightarrow I = 1 \text{ бит}.$$

Информационный объем требуемой видеопамяти рассчитывается по формуле:

$$I_{\text{П}} = I \cdot X \cdot Y,$$

где $I_{\text{П}}$ — глубина цвета (в битах на точку), X — количество точек изображения по горизонтали, Y — количество точек изображения по вертикали.

Цветовая модель *(система цветопередачи)* — способ представления различных цветов спектра в виде набора числовых характеристик определенных базовых компонентов.

Цветовая модель **RGB**. Цвета палитры RGB формируются путем сложения базовых цветов (красного R, зеленого G, синего B), имеющих различную интенсивность. Цвет *Color* в палитре определяется с помощью формулы:

$$Color = R + G + B$$

При глубине цвета в 24 бита (трехбайтная кодировка) значение интенсивности каждого базового компонента задается целым десятичным числом от 0 до 255 или двоичным числом от 00000000 до 11111111.

Цветовая модель **CMYK**. Палитра цветов CMYK формируется путем наложения базовых цветов (голубого C, пурпурного M, желтого Y и черного K), доли которых задаются в процентах (целых числах от 0 до 100). Цвет *Color* в палитре определяется с помощью формулы:

$$Color = C + M + Y + K$$

Цветовая модель **HSB**. Палитра цветов HSB формируется путем установки значений трех базовых компонентов — оттенка H, насыщенности S и яркости B. Оттенок H определяет цвет в спектре и задается целым числом от 0 до 360 (0 — красный цвет, 360 — фиолетовый). Эту модель используют художники при создании

компьютерных изображений, моделируя нужный цвет на «виртуальном мольберте» графического редактора.

Пример 1. В процессе преобразования растрового графического изображения количество цветов уменьшилось с 65 536 до 16. Как уменьшился его информационный объем?

Решение.

$2^{I_1} = 65536 = 2^{I_1} = 2^{16} \Rightarrow I_1 = 16;$

$2^{I_2} = 16 = 2^{I_2} = 2^4 \Rightarrow I_2 = 4;$

$\frac{I_1}{I_2} = \frac{16}{4} = 4.$

Ответ: информационный объем уменьшился в 4 раза.

Пример 2. Черно-белое растровое графическое изображение имеет размер 10 × 10 точек. Каков информационный объем изображения?

Решение. В палитре два цвета, следовательно, глубина цвета $I = 1$ бит. Информационный объем $I_{\text{П}}$ найдем по формуле:

$I_{\text{П}} = I \cdot X \cdot Y = 1 \cdot 10 \cdot 10 = 100$ (бит).

Ответ: 100 бит.

Векторная графика

В векторной графике основным элементом изображения является линия. **Линия** — это элементарный объект векторной графики. Чем длиннее растровая линия, тем больше памяти она занимает. Простейшие объекты объединяются в более сложные.

Программы, предназначенные для работы с векторными изображениями, называют *векторными графическими редакторами*. Их используют, когда основным требованием к изображению является высокая точность формы.

3.3.1. Форматы графических и звуковых объектов

Форматы растровых графических файлов

- *BMP* — изображение в этом формате сохраняется посимвольно, без сжатия;
- *JPEG* — использует эффективные алгоритмы сжатия данных, которые значительно уменьшают размеры файлов. Это достигается за счет необратимой потери части данных и ухудшения качества изображения;
- *GIF* — самый «плотный» из графических форматов, не имеющих потери информации, в нем хранятся и передаются малоцветные (до 256 цветов) изображения;
- *TIFF* — формат, поддерживаемый всеми основными графическими редакторами, включает в себя алгоритм сжатия без потерь информации, изображения сохраняются с высоким качеством;
- *PNG* — формат, аналогичный формату GIF, но позволяющий использовать значительно больше цветов в изображении.

Существуют и другие форматы растровых файлов, такие как *PCX, IFF, LBM, IMG, MAC, MSP, PGL*.

Форматы векторных графических файлов

- *WMF* — используется для хранения коллекции графических изображений Microsoft Clip Gallery. Возможные расширения файлов: .wmf, .emf, .wmz, .emz;
- *CGM* — используется как стандартный формат векторных графических данных в сети Интернет;
- *EPS* — формат, поддерживаемый программами для различных операционных систем.

Рекомендуется для создания иллюстраций в настольных издательских системах;

- *CDR* — формат файлов векторного графического редактора CorelDraw!, сохраняет векторную графику, а также текст и растровые изображения. Изображение в файле может состоять из нескольких страниц;
- *AI* — оригинальный формат файлов векторного графического редактора Adobe Illustrator;
- *SVG* — универсальный формат двумерной графики, сохраняет в файле текст, графические изображения и анимацию.

Форматы звуковых файлов

Звуковые форматы делятся на форматы без потери качества (*lossless*) и с потерей качества (*lossy*).

К *lossless*-форматам относятся *WAV*, *APE*, *FLAC*. В них сжатие или не применяется совсем, или применяется обратимое (с помощью механизма архивации).

В *lossy*-форматах — *MP3*, *OGG*, *WMA* и др. — кроме архивации применяется психоакустическое кодирование (например, маскировка более слабого сигнала более сильным), учитывающее физиологические свойства человеческого слуха. При этом ради достижения большего сжатия из сигнала выбрасывается часть информации, и его полное восстановление становится невозможным.

3.3.2. Ввод и обработка графических объектов

Средства и технологии работы с графикой

Аппаратные средства обработки графической информации:

- мониторы и видеокарты, поддерживающие графический режим отображения;

- видеоускорители, позволяющие увеличить скорость выполнения операций по обработке графической информации и, таким образом, разгружающие центральный процессор;
- 3D-акселераторы, способные самостоятельно обрабатывать графические объекты в трехмерном пространстве и в масштабе реального времени;
- манипуляторы «мышь»;
- графические планшеты, предназначенные для ввода изображения прямым рисованием на поверхности планшета;
- сканеры;
- принтеры;
- графопостроители (плоттеры).

Программные средства обработки графической информации:

- графические редакторы для создания, редактирования и просмотра графических изображений;
- средства создания анимации;
- программные средства для работы с трехмерной графикой;
- средства деловой графики.

Обработка изображения

Простые приемы обработки изображений:

- масштабирование;
- обрезка;
- выделение области для обработки;
- изменение яркости /контрастности;
- установка цветового баланса;
- выбор варианта коррекции изображения;
- применение эффектов резкости и размытия.

Инструменты и приемы более тонкой обработки изображений: лупа, «палец», вклейка фрагмента, работа с точками.

3.3.3 Ввод и обработка звуковых объектов

Звук — это продольная волна сжатия/расширения, распространяющаяся в воздушной, водной или другой среде (механические колебания). Звук характеризуется *интенсивностью*, которую человек воспринимает как громкость, и *частотой*, воспринимаемой как тон. Чем больше интенсивность, тем громче звук; чем выше тон, тем выше частота.

Для создания и редактирования цифровых звукозаписей применяются специальные программы — *звуковые редакторы*, например Adobe Audition, Sound Forge. Они позволяют не только записывать, воспроизводить и синтезировать звук, но и проводить его сложную обработку.

Для компьютерной записи звука необходимо подключить источник сигнала к звуковой карте компьютера и настроить ее, т. е. выбрать источник и уровень сигнала. Для этого может быть использована стандартная программа *Регулятор громкости*, входящая в состав ОС Windows. При выборе команды *Параметры/Свойства* откроется окно *Свойства*, в котором нужно выбрать переключатель *Запись*, указать нужный вход (*Microphone* или *Line-in*) и установить необходимый уровень записи. Далее можно использовать любой звуковой редактор.

При использовании звуковых редакторов новый файл создается традиционной командой *File/New*. При этом потребуется установить частоту дискретизации и глубину кодирования звукового сигнала, а также режим записи

(*стерео* или *моно*). Для музыкального сигнала лучше использовать параметры компакт-диска (44 100 Гц, 16 бит, 2 канала); если требования к качеству звука выше, можно поднять частоту дискретизации до 48 000 Гц. Речь с микрофона в нестудийных условиях с приемлемым качеством можно записать с такими параметрами: частота дискретизации 8 000 Гц, 8 бит, 1 канал.

3.4. Обработка числовой информации

3.4.1. Математическая обработка статистических данных

Зависимости между параметрами некоторого объекта, процесса, явления могут быть выражены с помощью математических формул. В некоторых случаях коэффициенты этих формул могут быть получены в результате статистической обработки экспериментальных данных.

Статистика — это наука о сборе, измерении и анализе больших массивов количественных данных. Статистические данные носят приближенный, усредненный характер, их получают путем многократных измерений. Математический аппарат статистики разрабатывает раздел науки под названием «математическая статистика». Статистические данные используются, в частности, для получения упрощенного математического описания сложной или неизвестной зависимости между данными некоторой системы (регрессионные модели).

Статистические данные можно обрабатывать с помощью электронных таблиц. Например, статистические функции электронных таблиц MS Excel позволяют вычислять среднее арифметическое числовых данных (СРЗНАЧ), среднее геометрическое положительных числовых данных (СРГЕОМ), минимальное и максимальное значения набора данных (МАКС, МИН), выполнять различные подсчеты (СЧЁТ, СЧЁТЗ, СЧЁТЕСЛИ, СЧИТАТЬ ПУСТОТЫ и т. д.).

Сферы применения статистического анализа данных:

- экономика (анализ результатов деятельности предприятий и организаций для оценки состояния финансового, сырьевого и других рынков, анализ прибыльности инвестиционной деятельности, составление краткосрочных планов и долгосрочных прогнозов);
- социология и психология (обработка и анализ результатов опросов, тестирования, анкетирования);
- научная деятельность (обработка результатов экспериментов, оценка их достоверности, проверка гипотез и пр.).

3.4.2. Использование динамических (электронных) таблиц для выполнения учебных заданий из различных предметных областей

Прикладные программы, предназначенные для работы с числовыми данными, представленными в прямоугольных таблицах, называются *табличными процессорами*, или *электронными таблицами*.

Эти программы позволяют создавать *динамические (электронные) таблицы*, в которых автоматически происходит пересчет значений формул при изменении исходных данных, используемых в этих формулах. Табличные процессоры предназначены для математических, финансовых, статистических расчетов, построения диаграмм для более наглядного воспроизведения данных и ведения простейших баз данных.

Один из широко распространенных табличных процессоров — MS Excel. Его файл — это электронный документ (*рабочая книга*), который состоит из прямоугольных таблиц (*рабочих листов*). Электронную таблицу (ЭТ) можно редактировать, форматировать, удалять и сохранять во внешней памяти. Рабочий лист состоит из столбцов с именами A, B, C... и строк с номерами 1, 2, 3..., на пересечении которых находятся *ячейки* таблицы.

Ячейка — основная единица хранения данных ЭТ.

Адрес ячейки (ссылка на ячейку) образуется из имени столбца и номера строки: A1, D3, AK10454 и т. п. *Активная ячейка* выделяется на экране жирной рамкой. Данные можно вводить только в активную ячейку, их можно видеть также в строке формул над листом. Активную ячейку можно выделить щелчком мышью.

В ячейку можно поместить данные следующих типов: текст, число, формулу. Текст и числа рассматриваются как константы.

Формулы — одно из важнейших средств табличных процессоров. Формула должна начинаться с одного из знаков: равенство, плюс или минус, может включать в себя числа, адреса ячеек

данного или другого рабочего листа, функции (математические, статистические, финансовые и др.) и знаки математических операций.

Функция — это заранее определенная формула, которая оперирует с одним или несколькими значениями и возвращает одно или несколько значений.

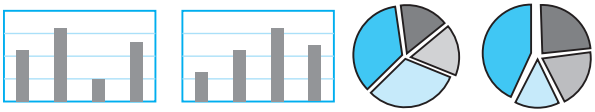
Типы функций:

- математические;
- текстовые;
- логические;
- даты и времени;
- статистические и др.

Пример 1. Дан фрагмент электронной таблицы. После выполнения вычислений была построена диаграмма по значениям диапазона ячеек *B1:B4*. Какая из диаграмм является верной?

	A	B
1	4	= A1 * A2
2	3	= B1 + 5
3		= B2 − B1
4		= B3 + 2 * A1

1) 2) 3) 4)



Решение. Сначала вычислим значения в ячейках *B1:B4* по указанным формулам:

$$B1 = A1 * A2 = 4 * 3 = 12;$$
$$B2 = B1 + 5 = 12 + 5 = 17;$$
$$B3 = B2 - B1 = 17 - 12 = 5;$$
$$B4 = B3 + 2 * A1 = 5 + 2 * 4 = 13.$$

Итак, значения ячеек диапазона *B1:B4* равны 12; 17; 5; 13. Такому набору значений соответствует диаграмма под номером 1.
Ответ: диаграмма 1.

Анализ модели

Естественно-научный эксперимент. Пусть проводится эксперимент: последовательно бросают тяжелый шарик с 1-го, 2-го, ... 10-го этажа, замеряют высоту начального положения шарика и фиксируют время его падения. По результатам эксперимента заполняют таблицу. Дополняют таблицу расчетными значениями времени, вычисленными по известной формуле $t_{\text{расчетное}} = \sqrt{\frac{2H}{g}}$ (зависимость времени падения тела на землю от первоначальной высоты без учета сопротивления воздуха, где H — высота, g — ускорение свободного падения).

H(м)	$t_{\text{из опыта}}$ (с)	$t_{\text{расчетное}}$ (с)	Отклонение
3	0,8	0,78	0,01
6	1,1	1,11	0,005
9	1,3	1,35	0,025
12	1,5	1,56	0,03
16	1,7	1,75	0,025
18	1,9	1,92	0,01
21	2,0	2,07	0,035
24	2,2	2,21	0,005
27	2,3	2,35	0,025
30	2,4	2,47	0,035

Для вычисления $t_{\text{расчетное}}$ в MS Excel была использована функция КОРЕНЬ() извлечения квадратного корня из числа. Формула расчета

в ячейке *C2* выглядит как КОРЕНЬ(2**A2*/9,81) (где *A2* — адрес ячейки, содержащей соответствующее значение высоты).

Визуальное сравнение опытных и расчетных данных (2-го и 3-го столбцов таблицы) позволяет предположить их достаточную близость. Рассчитаем в 4-м столбце таблицы отклонение опытных данных от расчетных значений. Воспользуемся для этого встроенной функцией MS Excel СРОТКЛ(), которая вычисляет среднее абсолютных значений отклонений точек данных от среднего. Функция среднего отклонения является мерой разброса множества данных.

На основании полученных значений можно сделать вывод об адекватности использованной модели зависимости времени падения тела на землю от первоначальной высоты H без учета сопротивления воздуха.

3.4.3. Использование инструментов решения статистических и расчетно-графических задач

Для определения статистической зависимости данных необходимо выполнить два шага:

1. На основании физического смысла имеющихся статистических данных определить вид аналитической зависимости. Это может быть, например, полином второй степени $y = ax^2 + bx + c$, линейная зависимость $y = ax + b$ и т. п. (Во всех формулах x — аргумент, y — значение функции, a, b, c — ее параметры.)
2. По имеющимся статистическим данным найти (с помощью метода наименьших квадра-

тов) значения величин *a*, *b*, *c*, определяющих конкретный вид принятой зависимости.

Полученная аналитическая зависимость называется **регрессионной моделью**.

MS Excel позволяет построить регрессионную модель по статистическим данным и получить значение *коэффициента детерминированности* (*достоверности*) R^2. Он принимает значения от 0 до 1 и показывает, насколько удачной является полученная регрессионная модель. Если $R^2 = 1$, то функция точно проходит через табличные значения, если $R^2 = 0$, то выбранный вид регрессионной модели совсем неудачен. Чем ближе R^2 к 1, тем удачнее модель.

Алгоритм построения регрессионной модели в MS Excel:

1. Выделить обе колонки исходных данных (наборы *X* и *Y*).
2. Вызвать *Мастер диаграмм* и с его помощью построить *Точечную диаграмму*.
3. Для полученной диаграммы выбрать меню *Диаграмма — Добавить линию тренда* (тренд — это график регрессионной модели).
4. В открывшемся диалоговом окне выбрать один из вариантов линий тренда, например линейную. Перейти к вкладке *Параметры* и задать опции: *Показывать уравнение на диаграмме* и *Поместить на диаграмму величину достоверности аппроксимации* (R^2).
5. На экране появятся линия тренда, уравнение с числовыми параметрами и коэффициент R^2. Проанализировать линию тренда (как часто ее точки совпадают с точками исходной диаграммы) и значение R^2.

6. Повторить построение регрессионной модели для других видов аналитических зависимостей.
7. Выбрать из полученных регрессионных моделей ту, которая имеет наибольшее значение R^2.

3.5. Технология поиска и хранения информации

3.5.1. Системы управления базами данных. Организация баз данных

База данных (*БД*) — упорядоченный набор данных об объектах и их взаимосвязях в рамках некоторой предметной области.

Основные типы организации данных в БД и связей между ними:

- *иерархический*. В этой структуре элементы в записи строго упорядочены: один элемент считается главным, остальные — подчиненными;
- *сетевой*. Это более гибкая структура, т. к. в ней дополнительно к вертикальным связям устанавливаются горизонтальные связи;
- *табличный*. Это наиболее распространенная структура. Информация в ней организована в виде таблиц. Каждая строка таблицы содержит информацию об одном отдельном объекте описываемой в БД предметной области, а каждый столбец — определенные характеристики (свойства, атрибуты) этих объектов.

Поля — основные элементы таблицы БД. Они обладают свойствами, от которых зависит, какие типы данных можно вносить в поле и какие операции можно производить над его данными.

Свойства поля	Описание
Размер	Выражается в знаках (или в символах)
Имя	Должно быть уникальным для каждого поля
Подпись	Отображается в заголовке столбца
Формат	Устанавливает формат данных

Основные типы полей:

- *текстовое* — для текста (255 символов);
- *числовое* — для числовых данных;
- *дата/время* — для даты и времени в определенном формате;
- *логическое* — для логических данных, имеющих только два значения (да — 1, нет — 0); имеет длину 1 бит;
- *денежное* — для ввода чисел в денежном формате;
- *поле объекта OLE* — для хранения картинок, музыкальных клипов, видеозаписей и т. д.;
- *счетчик* — для чисел, имеющих свойство автоматического наращивания;
- *поле MEMO* — для длинных текстов (до 65 535 символов (64 Кбайт)).

Модель данных, представляющая собой совокупность таблиц с установленными между ними связями, называется *реляционной*. В реляционной модели каждая таблица описывает один класс объектов.

Типы связей между таблицами БД:

1. «Один к одному» — связанные таблицы имеют одинаковое количество записей, и между этими записями установлено взаимно однозначное соответствие.
2. «Один ко многим» — каждой записи в одной (главной) таблице могут соответствовать несколько записей в другой (подчиненной) таб-

лице, а запись в подчиненной таблице не может иметь более одной соответствующей ей записи в главной таблице.

3. «Многие ко многим» — одной записи в первой таблице могут соответствовать несколько записей во второй таблице и наоборот. В реляционной модели данных две таблицы, находящиеся в отношении «многие ко многим», могут быть связаны только с помощью третьей (связующей) таблицы.

Для создания, наполнения и обработки баз данных разработаны специальные программные средства — *системы управления базами данных (СУБД)*.

Пример 1. Представлен фрагмент таблицы БД клиентов фирмы. Сколько полей и сколько записей в данной таблице?

Код клиента	Фамилия	Адрес	Телефон	E-mail
К1216Р	Карпов	ул. Кирова, 25	2892316	kda@mail.ru
М1347П	Маслов	ул. Королева, 12	7937847	maslov63@rambler.ru

Решение. Поля — это столбцы таблицы, а записи — это строки. Следовательно, в данной таблице 5 полей (*Код клиента, Фамилия, Адрес, Телефон, E-mail*) и 2 записи (о клиентах Карпове и Маслове).

3.5.2. Использование инструментов поисковых систем (формирование запросов)

Основными инструментами обработки данных является *сортировка, фильтр* и *запрос*.

Сортировка — это упорядочивание данных по некоторому признаку. Различают сортировку *по возрастанию* и *по убыванию*. Для числовых значений сортировка означает ранжирование по значению, а для текстовых — упорядочивание по алфавиту.

В СУБД MS Access сортировка таблицы осуществляется только по одному полю. Каждая новая сортировка отменяет результаты предыдущей. Вложенные сортировки выполняются с помощью запросов.

Фильтр — отбор из таблицы БД тех записей, которые удовлетворяют требованиям пользователя. Фильтры позволяют задать сложные условия отбора записей.

Для поиска и отбора записей могут использоваться специальные *подстановочные знаки* (*маска*):

***** — любое количество допустимых символов;

? — любой допустимый символ;

— любая цифра;

[ ] — любой символ из заключенных в скобки (например, маска *b*[*ae*]*ll* найдет *ball, bell*, но не *bill*);

– — любой символ из диапазона (например, *b*[*a*–*c*]*d* найдет *bad*, *bbd*, *bcd*);

! — любой символ, кроме заключенных в скобки (например, *b*[*!ae*]*ll* найдет *bill*, *bull*, но не *ball*, *bell*).

Все виды фильтров можно применять как ко всей таблице БД, так и к уже отобранным по некоторому критерию записям.

Запрос — это мощное средство обработки данных, в том числе соединяющее в себе возможности сортировки и фильтрации. С помощью запросов можно отбирать данные из нескольких таблиц сразу, назначать сортировку и условия отбора для нескольких полей, созда-

вать новые *вычисляемые поля*, в которых данные преобразуются при помощи формул. Запросы являются самостоятельными объектами БД, т. е. сохраняются с некоторым именем и могут использоваться в дальнейшем повторно.

Для создания запросов к БД разработан специальный язык запросов **SQL** *(Structured Query Language)* — структурированный язык запросов.

В СУБД Access реализовано простое средство, которое позволяет создавать запросы без знания SQL, — *бланк запроса*. С его помощью можно сформировать запрос простыми приемами, перетаскивая мышью элементы запроса между окнами и их областями.

С помощью запросов можно производить вычисления. Поле, содержимое которого является результатом расчета на основании содержимого других полей, называется *вычисляемым*. В общем случае оно существует только в результирующей таблице — в исходных (базовых) таблицах такое поле не создается и таблицы не изменяются.

Пример 1. Таблица БД содержит данные об архивировании разных файлов несколькими архиваторами: исходный размер файлов и размер полученных архивов. Ниже приведен фрагмент этой таблицы.

Имя файла	Размер	ZIP	RAR	ARJ
Text1.doc	285	114	106	112
Ref.doc	843	42	34	41

Требуется отобрать файлы, исходный размер которых был больше 2 Мбайт, а при использовании WinRAR он уменьшился более чем в 4 раза. Какой вариант условия следует использовать для формирования запроса:

1) (*Размер* > 2000) И (*Размер*/*RAR* > 4);

2) (*Размер* > 2048) И (*RAR* < 256);
3) (Размер > 2048) ИЛИ (Размер/*RAR* > 4);
4) (*Размер* > 2048) И (*Размер*/*RAR* > 4).

Решение. Исходя из условия задачи, в условии запроса должна использоваться логическая операция «И» (логическое умножение), поэтому вариант 3 не подходит. Кроме того, надо учесть, что 1 Мбайт = 1024 байта. Запрос с условием 1 может отобрать записи о файлах с исходным размером менее 2 Мбайт, следовательно, также не подходит. Правильный вариант — 4.

Ответ: вариант 4.

3.6. Телекоммуникационные технологии

3.6.1. Специальное программное обеспечение средств телекоммуникационных технологий

Компьютерная сеть (КС) — это совокупность компьютеров, соединенных линиями связи и оснащенных коммуникационным оборудованием и программным обеспечением.

Линия связи — это оборудование, с помощью которого осуществляется соединение компьютеров в сеть.

Взаимодействие объектов сети (серверов и клиентов) осуществляется по каналам связи, для которых используются разные физические среды, определяющие средства соединения компьютеров. Широко применяются соединения компьютеров с помощью радиоволн, по оптоволоконным, электрическим, телефонным кабелям и т. д.

Сервер (*хост*) — компьютер, предоставляющий свои ресурсы для совместного использования.

Рабочая станция (*клиент*) — компьютер, пользующийся ресурсами сети.

Основные характеристики каналов связи:

- скорость передачи данных (пропускная способность), измеряется числом бит информации, переданных по сети за одну секунду;
- надежность (способность передавать информацию без искажений и потерь);
- стоимость;
- резервы развития.

Коммуникационное, или *сетевое оборудование*, — это периферийные устройства, которые осуществляют преобразование сигналов, используемых в компьютере, в сигналы, передаваемые по линиям связи, и наоборот.

Объединенные в сеть компьютеры всегда работают под управлением специальных программ управления сетью. Среди них выделяются программы «низкого» и «высокого» уровня. ПО *«низкого» уровня* управляет сетевым оборудованием с целью преобразования сигналов из одного вида в другой. Эти программы ничего «не знают» о структуре передаваемой информации. ПО *«высокого» уровня* распознает и обрабатывает информацию в зависимости от ее характера и способа организации.

К программным средствам функционирования сети относятся такие компоненты, как *сетевые операционные системы* и *сетевые приложения* (браузеры и др.).

Браузер — сетевое программное обеспечение, предназначенное для отображения содержимого веб-страниц.

Сетевая операционная система — это основа функционирования любой сети. Она предназначена для управления принятыми/переданными сообщениями между серверами и рабочими станциями. Кроме того, она позволяет пользователям работать с общими сетевыми дисками или принтерами.

Компьютерные сети делятся на *локальные* и *глобальные*.

Локальная вычислительная сеть (*ЛВС*) объединяет компьютеры, расположенные на небольшом расстоянии друг от друга. Она представляет собой замкнутую систему.

Для подключения компьютера к локальной сети используется специальное устройство, называемое *сетевой картой,* или *сетевым адаптером*. Существуют материнские платы со встроенным сетевым адаптером. Сетевые адаптеры (ArcNet, Ethernet, TokenRing) различаются производительностью (скоростью передачи данных) и соответственно стоимостью. К сетевому адаптеру подключается *сетевой кабель*, например, оптоволоконный или «витая пара». Для подключения к локальной сети портативных компьютеров часто используют беспроводное подключение (радиосвязь или связь на инфракрасных лучах).

Для построения сети используются и другие сетевые устройства: *хабы* (или концентраторы), *коммутаторы, маршрутизаторы* (или роутеры) и др.

Многие организации, заинтересованные в защите информации от несанкционированного доступа, создают собственные *корпоративные сети*. Такие сети могут объединять десятки тысяч компьютеров, размещенных в различных городах и странах.

Глобальная сеть — соединения локальных, региональных и корпоративных сетей и отдельных компьютеров, находящихся на больших расстояниях друг от друга. Большие расстояния требуют наличия дополнительных устройств для обработки больших объемов информации и пересылки ее на большие расстояния. Это *серверы глобальной сети*, представляющие собой очень мощные компьютеры.

Современные глобальные сети все чаще используют системы спутниковой связи, что значительно расширяет их масштабы и возможности.

Интернет — это глобальная компьютерная сеть, объединяющая с помощью разных каналов связи сети различных регионов, организаций, государств на основании общего свода правил (протокола TCP/IP).

Типы адресов в Интернете:

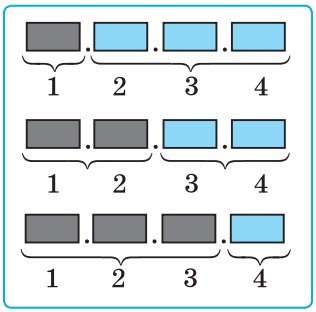
- IP-адреса;
- доменные адреса.

IP-адрес по смыслу аналогичен почтовому индексу. *IP-адрес* — это последовательность из четырех чисел, разделенных точками. Каждое из чисел занимает 1 байт (т. е. может принимать значения 0...255). Так как 1 байт = 8 бит, эти числа часто называют *октетами*. Используется три класса IP-адресов: *A*, *B* и *C*. Класс IP-адреса определяет, сколько октетов отводится под адрес сети и сколько — под адрес компьютера.

Класс A используется для работы с небольшим количеством сетей (до 126), содержащих большое число компьютеров (≈16,8 млн). *Класс B* — для работы со средним количеством сетей (до 16 384), содержащих среднее число компьютеров (до 65 534). *Класс C* — для работы с боль-

шим количеством сетей (до ≈2 млн), содержащих малое число компьютеров (до 254).

Класс определяют по значению первого октета:



- если в первом октете число от 1 до 126 — класс *A*;
- если в первом октете число от 128 до 191 — класс *B*;
- если в первом октете число от 192 до 223 — класс C.

Доменный адрес состоит из имен, разделенных точками. Имена строятся по иерархическому принципу, самый правый — домен верхнего (первого) уровня. Он определяет страну или тип организации, которой принадлежит компьютер. Затем справа налево указываются домены следующих уровней. *Домен второго уровня* определяет организацию, которая владеет или управляет сетью, включающей данный компьютер. Доменный адрес компьютера включает в себя как минимум два уровня доменов. Примеры доменных имен:

gramota.ru
translate.google.com

Домены могут сочетать региональные и организационные уровни, например: edu.ru — домены образовательных организаций в России.

Доменные адреса должны быть преобразованы в IP-адреса. Существуют специальные *DNS-серверы* (Domain Name System — система

доменных имен), которые хранят таблицы соответствия доменных и IP-адресов.

В Интернете для обращения к веб-документам используется система адресации *URL* (Uniform Resource Locator — унифицированный указатель ресурса). Она применяется для указания конкретного файла на конкретном компьютере через имя файла и структуру папок компьютера. *URL* включает в себя: протокол доступа к документу; доменное имя или IP-адрес сервера, на котором находится документ; путь к файлу и собственно имя файла. Протокол доступа к документу определяет способ передачи информации; после указания имени протокола ставят двоеточие и две косые черты. Для доступа к веб-страницам используется протокол передачи гипертекста HTTP (*http*://).

Например, адрес URL

http://home.microsoft.com/intl/ru/www_tour.html

содержит такие элементы:

1) http:// — префикс, указывающий тип протокола. В данном случае протокол **HTTP** означает, что адрес относится к веб-серверу. В качестве префикса (протокола) могут быть также указаны ftp://, file://, news://;
2) home.microsoft.com — доменное имя сервера. После него может следовать число, обозначающее порт, через который будет производиться подключение к хосту;
3) /intl/ru/ — папка *ru* корневого каталога *intl* сервера;
4) www_tour.html — имя файла.

К наиболее давним службам Интернета относятся:

- *E-mail* (*электронная почта*) — служба передачи электронных сообщений;

- *Usenet* (*телеконференции, группы новостей*) — разновидность сетевой газеты или доски объявлений;
- *FTP* — служба хранения и передачи файлов;
- *IRC* — служба для текстового общения в реальном времени;
- *Telnet* — служба удаленного доступа к компьютерам;
- *World Wide Web* (*WWW*, *Всемирная паутина*) — служба поиска и просмотра гипертекстовых документов, включающих в себя графику, звук и видео.

В процессе развития Интернета возникли новые сервисы:

- *потоковое мультимедиа* (служба непрерывной передачи информации от провайдера потокового вещания — онлайн-телевидение, веб-радио, подкастинг, видеолекции, интернет-концерты и т. д.);
- *файлообменные сети* (совместное равноправное использование файлов);
- *голосовая и видеосвязь* (IP-телефония, или VoIP — передача речи по каналам Интернета, аудиоконференции, видеозвонки и пр.);
- *облачные онлайн-хранилища* (сетевой доступ к устройствам хранения данных на многочисленных, распределенных в сети серверах, как SkyDrive, Google Drive, Dropbox, могут быть представлены как локальный диск на компьютере пользователя);
- *облачные вычисления* (использование прикладных программ и др. через Интернет, например в рамках модели SaaS (Software as a Service — ПО как услуга) без необходимости установки ПО на своем ПК);

- *службы мгновенных сообщений* (обмен в реальном времени текстовыми сообщениями, например ICQ, голосовыми и видео, например Skype и пр.);
- *микроблогинг* (публикация коротких заметок, например Twitter);
- *электронные платежные системы*;
- *картографический сервис* (например, Google Maps, Яндекс.Карты);
- *массовые многопользовательские игры* и многие другие.

Существовавшие издавна службы значительно расширили спектр предоставляемых ими сервисов. В рамках Всемирной паутины (WWW) активно развиваются *веб-форумы* (обсуждение участниками вопросов определенной тематики), *блоги* (среда сетевого общения через интернет-журналы), *вики-проекты* (сайты, коллективно создаваемые и изменяемые самими пользователями, такие как Википедия), *интернет-магазины* и *аукционы*, *видео-* и *фотохостинг* (размещение и предоставление доступа к фото- и видеоблогам, фильмам и клипам, как, например, YouTube, Flickr, Panoramio и др.), *социальные сети* (сервис взаимодействия пользователей, объединенных общими интересами, взаимоотношениями и т. п., например Facebook, ВКонтакте).

Веб 2.0 — сфера веб-сервисов и проектов, развиваемых и улучшаемых самими пользователями, в т.ч. социальные сети, вики-справочники, блоги, фото- и видеохостинги и т. д.

Для поиска информации в Интернете существуют различные *поисковые системы*. Большинство из них предназначены для поиска в WWW, существуют также системы поиска

файлов на FTP-серверах, товаров в интернет-магазинах, новостей в группах Usenet.

Современные поисковые системы часто являются *информационными порталами*, которые предоставляют пользователям, кроме возможностей поиска, доступ к другим информационным ресурсам: новостям, информации о погоде, валютных курсах и т. д.

3.6.2. Инструменты создания информационных объектов для Интернета

Информационный объект — представление объекта предметной области в информационной системе, определяющее его структуру, атрибуты, ограничения целостности и, возможно, поведение.

Наиболее часто информация размещается на веб-серверах WWW в виде веб-страниц. *Веб-страница* представляет собой текстовый файл с гиперссылками (гипертекстовый документ), размеченный на языке *HTML*. Несколько веб-страниц, объединенных между собой гиперссылками, общей темой, оформлением и находящихся обычно на одном веб-сервере, называются *веб-сайтом*.

Существует множество управляющих программ — *CMS*, *систем управления содержимым*, — которые предоставляют инструменты для создания (часто на базе множества шаблонов), редактирования, контроля, добавления и удаления информации веб-сайта. Обычно они реализованы в виде специального визуального редактора, который автоматически генерирует HTML-код веб-страницы. Это позволяет пользо-

вателям-непрограммистам легко создавать личные веб-страницы, блоги, интернет-магазины и т. п. К наиболее популярным системам CMS относятся WordPress, Joomla!, Drupal.

Веб-разработка — процесс создания веб-сайтов как целостных информационных систем с присущей им динамичностью, интерактивностью, размещением информации в базах данных. Создание таких веб-приложений требует участия веб-дизайнеров, веб-программистов, системных администраторов. Инструменты создания веб-сайтов непрерывно совершенствуются и включают такие средства, как *CSS* (технология задания единых стилей оформления для множества веб-страниц), *Javascript* (язык сценариев для придания веб-страницам интерактивности), *Ajax* (технология динамического изменения содержания веб-страниц), *ActiveX* (технология использования на веб-страницах управляющих элементов), *VRML* (язык моделирования и включения в веб-страницы трехмерных анимационных изображений), *Java* (язык программирования, не зависящий от используемой компьютерной платформы и ОС), *PHP* (язык программирования для создания динамических веб-сайтов) и многие другие.

Средства языка HTML

Гипертекстовая технология заключается в том, что текст представляется как многомерный с иерархической структурой. *Разметка гипертекста* — использование специальных кодов, легко отделяемых от содержания документа и используемых для реализации гипертекста. Применение этих кодов подчиняется строгим правилам, определяемым спецификацией языка HTML.

Особенность описания документа средствами языка HTML связана с принципиальной невозможностью достижения абсолютной точности при воспроизведении исходного документа. Поэтому язык HTML предназначен не для форматирования документа, а для его функциональной разметки.

Управляющие конструкции языка HTML называются **тегами** (*дескрипторами*) и вставляются непосредственно в текст документа. Тег содержит ключевое слово (и, возможно, некоторые параметры-атрибуты), заключенное в угловые скобки <...> (например, <*DIV*>).

Теги HTML бывают парными и непарными. *Непарные теги* оказывают воздействие на весь документ или определяют разовый эффект в том месте, где они вставлены. При использовании *парных тегов* в документе размещают открывающий и закрывающий теги, которые воздействуют на часть документа, заключенную между ними. *Закрывающий* тег отличается от *открывающего* наличием символа «/» перед ключевым словом (</DIV>). Закрытие парных тегов выполняется так, чтобы соблюдались правила вложения:

<B> <I> *текст* </I> </B>

К открывающему тегу может быть добавлен *атрибут*, который представляет собой дополнительные ключевые слова, разделенные пробелами. Способ применения некоторых атрибутов требует указания *значения атрибута*. Значение атрибута отделяется от ключевого слова символом « = » и заключается в кавычки, например:

<H1 *align* = «*left*»>

Ключевые слова:

- HTML — начало и конец документа HTML;
- HEAD — начало и конец раздела заголовка;

- TITLE — начало и конец общего заголовка документа;
- BODY — начало и конец тела документа.

Структура документа HTML:

```
<HTML>
<HEAD> <TITLE> Заголовок документа
</TITLE> </HEAD>
<BODY>
    Текст документа
</BODY>
</HTML>
```

Теги заголовков и абзацев:

- от <H1></H1> до <H6> </H6> — шесть уровней заголовков;
- <P> </P> — абзац;
-
 — переход на другую строку;
- <HR> — горизонтальная линия.

Тег <FONT> и его атрибуты *face, size и color* задают гарнитуру, размер и цвет шрифта любого фрагмента текста.

Нумерованный список располагается внутри парного тега <OL> </OL>. Каждый элемент списка определяется тегом <LI>.

Маркированный список располагается внутри парного тега <UL> </UL>. Каждый элемент списка определяется тегом <LI>. Вид маркера можно задать с помощью атрибута TYPE тега <UL>: *disc* — диск, *square* — квадрат, *circle* — окружность.

Список определений располагается внутри парного тега <DL> </DL>. Текст оформляется в виде термина, который выделяется непарным тегом <DT>, и определения, которое следует за тегом <DD>.

Вставка рисунков или видеофрагментов выполняется с помощью тега <IMG>, имеющего основные атрибуты:

- *src* — исходный URL-адрес файла рисунка;
- *align* — способ выравнивания рисунка (например, *align* = "*center*" для центрирования);
- *border* — видимая рамка вокруг рисунка;
- *width* — ширина рисунка;
- *height* — высота рисунка;
- *dynsrc* — исходный URL-адрес видеофайла;
- *loop* — число повторений видео (если задано *loop* = "*infinite*", видео будет повторяться непрерывно);
- *start* — условие запуска видео (например, *start* = "*mouseover*" запускает видео при перемещении указателя мыши по области видео);

Гиперссылка состоит из двух частей: указателя ссылки и адресной части. *Указатель ссылки* — это текст или рисунок на веб-странице, щелчок по которому вызывает переход на другую страницу. *Адресная часть* гиперссылки — это URL-адрес документа, на который устанавливается ссылка. URL-адрес может быть абсолютным и относительным. *Абсолютный URL-адрес* — это полный интернет-адрес. *Относительный URL-адрес* указывает на местоположение документа относительно того документа, в котором находится указатель ссылки.

Гипертекстовая ссылка задается парным тегом <A>, который содержит обязательный атрибут *href*. В качестве значения атрибута используется URL-адрес документа, на который указывает ссылка, например:

<A *href* = "*http://www.site.com/index.htm*">

В примере указан абсолютный адрес. Обычно в такой форме задается ссылка на внешний документ. При использовании относительного адреса ссылка рассматривается как внутренняя, например:

<A *href = "text.htm"*>

Достоинство внутренней ссылки в том, что она сохраняет свою работоспособность при изменении адреса веб-узла.

Текст, располагающийся между тегами <A> и </A>, обычно изображается браузерами шрифтом определенного цвета (по умолчанию голубого); некоторые программы подчеркивают его. Посредством атрибутов в теге <BODY> можно изменить цвета текста гиперссылок.

Для публикации (размещения) сайта нужно найти подходящее место на одном из серверов Интернета и получить от провайдера URL-адрес сайта, а также секретные имя пользователя и пароль, которые необходимы администратору сайта для его редактирования.

3.7. Технологии управления, планирования и организации деятельности человека

Информатизация общества — совокупность взаимосвязанных политических, социально-экономических, научных факторов, которые обеспечивают свободный доступ каждому члену общества к любым источникам информации (кроме составляющих государственную и коммерческую тайну).

В середине XX века в науке и технике были получены три группы важнейших результатов:

- доказана аналогичность информационных процессов, происходящих при управлении в системах самой различной природы (технических, биологических, социальных);
- созданы ЭВМ и налажено их промышленное производство;
- разработаны методы решения дискретных оптимизационных задач в процессе управления (особенно в системах организационного управления).

Управляющие системы

Управление — совокупность управляющих воздействий, направленных на то, чтобы действительный ход процесса соответствовал желаемому. Таким образом, управление предполагает, что существует некоторый орган, систематически или по мере необходимости вырабатывающий управляющие воздействия. Такой управляющий орган принято называть *системой управления*. Управление обычно осуществляется через исполнительные органы, которые и изменяют действительный ход процесса. Управление должно быть целенаправленным. Управляющие воздействия должны быть скоординированы между собой, а не носить случайный характер, при котором не исключена возможность воздействий, прямо противоположных друг другу.

Процесс управления — это целенаправленное воздействие управляющей системы на управляемую систему, ориентированное на достижение определенной цели и использующее главным образом информационный поток. Оптимальное управление заключается в выборе наилучших

управляющих воздействий из множества возможных с учетом ограничений и на основе информации о состоянии управляемого объекта и внешней среды.

Администраторы, или руководители — люди, которые выполняют функции по принятию решений в процессах планирования и оперативного управления, реализуемых на низших уровнях управления, а также контроль над реализацией принятых решений. (За рубежом применяют термины *manager* (руководитель, управляющий) и *management* (административное управление) в отличие от *control* (управление в производственных системах).)

В производственных системах человек с помощью технических средств, которыми он манипулирует, непосредственно управляет технологическим или производственным процессом. Человека, осуществляющего такое управление, называют *оператором,* а систему, составным элементом которой является оператор, называют *эргатической* (эргатив — действующее лицо, деятель).

Системный подход к управлению

Системный подход рассматривает эффективность управления как результат, зависящий от многих факторов в совокупности. *Система* — это целостная совокупность, состоящая из взаимосвязанных частей, каждая из которых вносит свой вклад в характеристики целого.

Основные признаки систем:

- целостность и делимость;
- наличие устойчивых связей между элементами системы;
- организация;

- наличие таких качеств (свойств), которые присущи системе в целом, но не свойственны ни одному из ее элементов в отдельности.

Наличие взаимосвязей (взаимодействия) между элементами определяет особое свойство сложных систем — *организованную сложность*. Добавление элементов в систему не только вводит новые связи, но и изменяет характеристики многих или всех прежних взаимосвязей, приводит к исключению некоторых из них или появлению новых.

Важным принципом построения системы управления предприятием является рассмотрение предприятия как системы с многоуровневой (иерархической) структурой.

Иерархические системы управления предприятием

Иерархия — разделение систем на высшие и низшие, порядок подчинения низших по долж-

ности систем высшим. Одну и ту же систему можно разделить на подсистемы по-разному, это зависит от выбранных правил объединения элементов в подсистемы. Наилучшим, очевидно, будет набор правил, который обеспечивает системе в целом наиболее эффективное достижение цели.

Правила разбиения на подсистемы:

1. Каждая подсистема должна реализовывать единственную функцию системы.
2. Выделенная в подсистему функция должна быть легко понимаема независимо от сложности ее реализации.
3. Связь между подсистемами должна вводиться только при наличии связи между соответствующими функциями системы.
4. Связи между подсистемами должны быть простыми (насколько это возможно).

Подсистемы, непосредственно входящие в одну систему более высокого уровня, действуя совместно, должны выполнять все функции той системы, в которую они входят

Нижний уровень управления является источником информации для принятия управленческих решений на более высоком уровне. Если рассматривать поток информации от уровня к уровню, то количество информации уменьшается с повышением уровня, но при этом растет ее смысловое содержание.

Централизация — возможность направлять на реализацию решений крупные ресурсы, что позволяет решать сложные проблемы, требующие больших капиталовложений. Однако в многоуровневых системах это снижает оперативность оценки обстановки, приводит к задержкам принятия решений, к искажениям как

в процессе передачи информации, так и при ее обработке на промежуточных уровнях. Поэтому в сложных централизованных системах всегда присутствуют элементы *децентрализации*.

При рациональном сочетании элементов централизации и децентрализации информационные потоки в системе должны быть организованы таким образом, чтобы информация использовалась в основном на том уровне, где она возникает, т. е. надо стремиться к минимальной передаче данных между уровнями системы. В децентрализованных одноуровневых системах всегда выше уровень оперативности как при сборе информации о состоянии управляемой системы, оценке ситуации, так и при реализации принятых решений.

Процессный подход в теории управления

Процессный подход развивает идеи о существовании некоторых основных и универсальных функций управления. Причем с точки зрения процессного подхода эти функции рассматриваются не как взаимонезависимые, а как органически взаимосвязанные и образующие в своей совокупности единый процесс управления.

Управление — это система непрерывных и взаимосвязанных действий, группирующихся в управленческие функции. Процесс управления в целом рассматривается как хронологически упорядоченная и циклически организованная система управленческих функций. Следовательно, важным условием успешного управления является не только эффективность управленческих функций самих по себе, но и правильная их соорганизация в рамках единого процесса.

Широкое распространение приобрела точка зрения, согласно которой все разнообразие управленческих функций может быть сгруппировано в четыре базовые категории: планирование, организацию, мотивирование, контроль и две так называемые связующие функции: принятие решения и коммуникация. Последние направлены на согласование базовых функций.

Планирование — система методов и средств, посредством которых руководство обеспечивает единое направление усилий всех членов организации к достижению ее общих целей.

Функция организации — выбор или создание определенной структуры, упорядочивающей группу (группы) совместно работающих людей, а также саму их работу. Задача функции мотивирования состоит в том, чтобы члены организации действительно выполняли работу в соответствии с делегированными обязанностями и сообразуясь с планом.

Контроль — это процесс обеспечения того, что организация действительно достигает своих целей. Он предполагает установление контрольных стандартов; измерение достигнутого в действительности; сравнение достигнутого с ожидаемым; действия для коррекции отклонений от первоначального плана.

Принятие решения в роли «связующей функции» — это выбор того, как и что планировать, мотивировать, организовывать и исполнять.

Коммуникация — это процесс обмена информацией, ее смысловым значением между двумя или более людьми. Без него невозможна организация совместной деятельности. Следовательно, в организации должна быть обеспечена система эффективных коммуникаций.

Справочное издание

Для старшего школьного возраста

СУПЕРМОБИЛЬНЫЙ СПРАВОЧНИК

Панова Светлана Юрьевна

ИНФОРМАТИКА

Ответственный редактор *А. Жилинская*
Ведущий редактор *Т. Судакова*
Художественный редактор *Е. Брынчик*

ООО «Издательство «Эксмо»
127299, Москва, ул. Клары Цеткин, д. 18/5. Тел. 411-68-86, 956-39-21.
Home page: **www.eksmo.ru** E-mail: **info@eksmo.ru**

Оптовая торговля книгами «Эксмо»:
ООО «ТД «Эксмо». 142702, Московская обл., Ленинский р-н, г. Видное, Белокаменное ш., д. 1, многоканальный тел. 411-50-74.
E-mail: **reception@eksmo-sale.ru**

По вопросам приобретения книг «Эксмо» зарубежными оптовыми покупателями *обращаться в отдел зарубежных продаж ТД «Эксмо»*
E-mail: **international@eksmo-sale.ru**

International Sales: *International wholesale customers should contact Foreign Sales Department of Trading House «Eksmo» for their orders.*
international@eksmo-sale.ru

По вопросам заказа книг корпоративным клиентам, в том числе в специальном оформлении, *обращаться по тел. 411-68-59, доб. 2299, 2205, 2239, 1251.*
E-mail: **vipzakaz@eksmo.ru**

Оптовая торговля бумажно-беловыми и канцелярскими товарами для школы и офиса «Канц-Эксмо»: Компания «Канц-Эксмо»: 142700, Московская обл., Ленинский р-н, г. Видное-2, Белокаменное ш., д. 1, а/я 5. Тел./факс +7 (495) 745-28-87 (многоканальный). e-mail: **kanc@eksmo-sale.ru**, сайт: **www.kanc-eksmo.ru**

Подписано в печать 15.10.2012. Формат 70x90$^{1}/_{32}$.
Печать офсетная. Усл. печ. л. 5,83.
Тираж экз. Заказ

ISBN 978-5-699-59586-0

ВПЕРВЫЕ!

Супермобильный справочник с удобной навигацией и использованием QR-кодов!

ИНФОРМАТИКА

Весь курс информатики представлен в конспективной форме. К каждой теме приводится уникальный QR-код. Достаточно установить на телефоне специальное приложение, навести камеру телефона на QR-код, и вы мгновенно получите доступ к интернет-ресурсу с дополнительной информацией по изучаемой теме.

ВСЯ ШКОЛЬНАЯ ПРОГРАММА ПО ИНФОРМАТИКЕ С СОБОЙ!

ДОМА

В ШКОЛЕ

В ДОРОГЕ

Использование ресурсов Интернета сделает процесс обучения более увлекательным, эффективным и позволит сэкономить время!