# План учебника информатики для 10 класса

## (профильный уровень, 140 часов)

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

- Техника безопасности.

## 1. *Информация и информационные процессы*

*Виды информационных процессов. Процесс передачи информации. Восприятие, запоминание и обработка информации человеком, пределы чувствительности и разрешающей способности органов чувств.*

1.1. Информатика и информация.
1.2. Информационные процессы.
1.3. Измерение информации.
1.4. Структура информации.

## 2. *Кодирование информации*

*Сигнал, кодирование, декодирование,*

2.1. Язык, алфавит.
2.2. Кодирование.
2.3. Двоичное кодирование.
2.4. Декодирование.
2.5. Алфавитный подход к оценке количества информации.

*Системы счисления.*

2.6. Системы счисления.
2.7. Позиционные системы счисления.
2.8. Двоичная система счисления.
2.9. Восьмеричная система счисления.
2.10. Шестнадцатеричная система счисления.
2.11. Другие системы счисления.

*Дискретное (цифровое) представление текстовой, графической, звуковой информации и видеоинформации.*

2.12. Кодирование символов.

*Форматы графических и звуковых объектов.*

2.13. Кодирование графической информации.
2.14. Кодирование звуковой информации.
2.15. Кодирование видеоинформации.

*Скорость передачи информации.*
*Искажение информации. Кодирование с исправлением ошибок.*

2.16. Передача информации.
2.17. Сжатие информации.

## 3. *Логические основы компьютеров*

*Логика и алгоритмы. Высказывания, логические операции, кванторы, истинность высказывания.*

3.1. Логика и компьютер.
3.2. Логические операции.
3.3. Диаграммы Эйлера-Вена.
3.4. Упрощение логических выражений.
3.5. Синтез логических выражений.
3.6. Предикаты и кванторы.

3.7. Логические устройства компьютера.
3.8. Логические задачи.

## 4. *Компьютерная арифметика*

*Программная и аппаратная организация компьютеров.*

4.1. Хранение в памяти целых чисел.
4.2. Арифметические и логические (битовые) операции. Маски.
4.3. Хранение в памяти вещественных чисел.
4.4. Выполнение арифметических операций с нормализованными числами.

## 5. *Устройство компьютера*

*Архитектура компьютеров. Программная и аппаратная организация компьютеров.*

5.1. История развития вычислительной техники.
5.2. Перспективы развития компьютерной техники.
5.3. Архитектура компьютеров. Принципы фон Неймана.
5.4. Магистрально-модульный принцип построения ПК.
5.5. Процессор.
5.6. Внутренняя и внешняя память.
5.7. Устройства ввода.
5.8. Устройства вывода.

## 6. *Программное обеспечение*

*Виды программного обеспечения. Операционные системы.*

6.1. Что такое программное обеспечение?
6.2. Прикладные программы.
6.3. Системное программное обеспечение.

*Система программирования.*

6.4. Системы программирования.
6.5. Инсталляция программ.

*Информационная этика и право, информационная безопасность. Правовые нормы, относящиеся к информации, правонарушения в информационной сфере, меры их предотвращения.*

6.6. Правовая охрана программ и данных.

## 7. *Компьютерные сети*

*Архитектура компьютерных сетей. Программная и аппаратная организация компьютерных систем. Понятие о системном администрировании.*
*Представления о средствах телекоммуникационных технологий: электронная почта, чат, телеконференции, форумы, телемосты, интернет-телефония. Специальное программное обеспечение средств телекоммуникационных технологий. Использование средств телекоммуникаций в коллективной деятельности.*

7.1. Основные понятия.
7.2. Структура (топология) сетей.
7.3. Локальные сети.
7.4. Сеть Интернет.
7.5. Адреса в Интернете.
7.6. Всемирная паутина.
7.7. Электронная почта.
7.8. Другие службы Интернета.
7.9. Электронная коммерция.
7.10. Интернет и право.
7.11. Нетикет.

## 8. *Алгоритмизация и программирование*

*Язык программирования. Типы данных. Основные конструкции языка программирования.*

8.1. Алгоритм и его свойства.
8.2. Структура программы.
8.3. Вывод на экран.
8.4. Переменные. Типы данных.
8.5. Оператор присваивания. Арифметические выражения.
8.6. Условный оператор. Сложные условия.
8.7. Цикл со счетчиком.
8.8. Цикл с условием. Циклы с постусловием.
8.9. Множественный выбор.
8.10. Графические примитивы.

*Псевдослучайные последовательности.*

8.11. Случайные события. Датчики случайных чисел. Броуновское движение.
8.12. Метод Монте-Карло. Вычисление площадей и объемов фигур.
8.13. Построение графиков функций. Масштабирование.
8.14. Процедуры.

*Индуктивное определение объектов.*

8.15. Рекурсия. Рекурсивные фигуры.
8.16. Анимация. Моделирование вращения.
8.17. Функции. Логические функции.

*Матрицы (массивы).*

8.18. Массивы. Ввод и вывод. Заполнение случайными числами.
8.19. Поиск минимального элемента в массиве.
8.20. Алгоритмы обработки массивов.

*Сортировка.*

8.21. Сортировка массивов. Методы пузырька и вставки.
8.22. Быстрая сортировка массивов.
8.23. Поиск в массиве (линейный, двоичный).
8.24. Массивы в процедурах и функциях.
8.25. Символьные строки.
8.26. Рекурсивный перебор. Сочетания. Перестановки.
8.27. Матрицы.

*Основные этапы разработки программ. Разбиение задачи на подзадачи.*

8.28. Основные этапы разработки программ. Структурное программирование.

## 9. *Решение вычислительных задач на компьютере*

*Обработка числовой информации*
*Использование динамических (электронных) таблиц для выполнения учебных заданий из различных предметных областей: обработка результатов естественнонаучного и математического эксперимента, экономических и экологических наблюдений, социальных опросов, учета индивидуальных показателей учебной деятельности.*

9.1. Точность вычислений.
9.2. Решение уравнений.
9.3. Дискретизация.
9.4. Оптимизация.

*Математическая обработка статистических данных, результатов эксперимента, в том числе с использованием компьютерных датчиков.*
*Использование инструментов решения статистических и расчетнографических задач.*
*Обработка числовой информации на примерах задач по учету и планированию.*

9.5. Статистические расчеты.
9.6. Обработка результатов эксперимента.

## 10. ***Информационная безопасность***

*Технологии и средства защиты информации в глобальной и локальной компьютерных сетях от разрушения, несанкционированного доступа.*

10.1. Основные понятия.

10.2. Вредоносные программы.

*Правила подписки на антивирусные программы и их настройка на автоматическую проверку сообщений.*

10.3. Защита от вредоносных программ.

10.4. Что такое шифрование?

10.5. Хэширование и пароли.

10.6. Современные алгоритмы шифрования.

10.7. Стеганография.

10.8. Безопасность в Интернете.

# Глава I. Информация и информационные процессы

## I.1. Информатика и информация

### I.1.1. Информатика

Задачи, связанные с хранением, передачей и обработкой информации человеку приходилось решать во все времена: требовалось передавать знания из поколения в поколения, искать нужные книги в хранилищах, шифровать секретную переписку. К концу XIX века количество документов в библиотеках стало настолько велико, что появилась необходимость как-то систематизировать накопленную информацию, для того чтобы было удобно ее хранить и искать нужные данные. В конце XIX века зародилось новое научное направление, в котором изучалась *документальная* информация, то есть информации в виде документов (книг, журналов, статей и т.п.). В английском языке оно получило название «*information science*» («информационная наука», «наука об информации»).

Применение компьютерной техники значительно увеличило возможности людей в области работы с информацией. Слово *«информатика*[1]*»* в современном значении образовано в результате объединения двух слов: «информация» и «автоматика». Таким образом, получается «автоматическая работа с информацией». В английском языке существует близкое по значению выражение *«computer science»* (наука о компьютерах).

Современная информатика, которая стала самостоятельной наукой в 1970-х годах, включает следующие научные направления:

- **теоретическую информатику** (теорию информации, теорию кодирования, математическую логику, теорию систем и др.);
- **кибернетику** (теорию управления в природе, технике и обществе);
- **искусственный интеллект** (распознавание образов, понимание речи, машинный перевод, логические выводы, алгоритмы самообучения);
- **вычислительную технику** (устройство компьютеров и компьютерных сетей)**;**
- **программирование** (методы создания новых программ)**;**
- **прикладную информатику** (персональные компьютеры, прикладные программы, информационные системы и т.д.).

Раньше эти вопросы частично рассматривались в других науках – математике, лингвистике (науке о языке), электронике и др.

В понятие «информатика» часто включают и *новые информационные технологии*, связанные с применением компьютеров. Они используются практически во всех областях современной жизни: для оформления документов; для подготовки книг и журналов к печати; для расчета зарплаты; для продажи билетов на поезда и самолеты; для автоматизации производства; при проектировании зданий, кораблей, станков и т.д. Кажется, что все это – совершенно разные сферы деятельности, однако они «связаны» в единое целое понятием *«информация»*.

[1] Впервые это слово использовал немецкий ученый К. Штейнбух в 1957 году (в немецком языке – *Informatik*). Затем оно было введено во французский язык (фр. *informatique*) и переведено на английский (англ. *informatics*).

## I.1.2. Что такое информация?

Латинское слово *«informatio»* переводится как *«разъяснение»*, *«сведения»*. В быту под информацией мы обычно понимаем любые сведения или данные об окружающем нас мире и о нас самих. Однако дать общее определение информации весьма непросто. Более того, в каждой области знаний слово «информация» имеет свой смысл.

Философы говорят о том, что информация, как зеркало, *отражает* мир (реальный или вымышленный). Биологи чаще всего связывают понятие «информация» только с живой природой. Социологи изучают ценность и полезность информации в человеческом обществе. Специалистов по компьютерной технике в первую очередь интересует представление информации в виде знаков.

Попробуем посмотреть на информацию с разных сторон и попытаться выявить некоторые ее свойства. Прежде всего, информация сама по себе «бестелесна» или *нематериальна*, она не имеет формы, размеров, массы. С этой точки зрения информация – это содержание, которое человек с помощью своего сознания «выделяет» из окружающей среды.

**Информация** нематериальна.

Давайте сравним два изображения одинакового размера (рисунок справа). На первом из них пусто, а на втором мы видим фотографию. Вряд ли кто-то способен долго разглядывать чистый лист, в то же время можно долго смотреть на фотографию, открывая все новые и новые детали. Почему так?

1 2

Можно считать, что на первом рисунке нет информации, поэтому разглядывать его неинтересно. Там все одинаково – везде белый цвет. На втором рисунке есть *разнообразие*, и он перестал быть однородным.

**Информация** характеризует разнообразие (неоднородность) в окружающем мире.

Зачем вообще нам нужна информация? Дело в том, что наши знания всегда в чем-то неполны, в них есть *неопределенность*. Например, вы стоите на остановке автобуса и не знаете, на каком именно автобусе вам нужно ехать в гости к другу (его адрес известен). Неопределенность мешает вам решить свою задачу. Нужный номер автобуса можно определить, например, по карте с маршрутами транспорта. Очевидно, что при этом вы получите новую информацию, которая увеличила знание (уменьшила неопределенность).

**Информация** уменьшает неопределенность знаний.

Многие выдающиеся ученые XX века (Н. Винер, У. Эшби, К. Шеннон, А. Урсул, А. Моль, В. Глушков) давали свое определение информации, но ни одно из них не стало общепринятым. И этому есть свое объяснение.

Дело в том, что мы всегда определяем новые понятия через уже известные. И, в конечном счете, приходим к понятиям, которые определить невозможно. Например, все термины в геометрии определяются через *базовые* понятия *«прямая», «точка»* и *«плоскость»*.

В современной науке информация считается одним из базовых понятий, так же, как *материя* и *энергия*. Поэтому дать строгое определение информации не удается, можно только объяснить значение этого слова на примерах и сравнить с другими понятиями. Норберт Винер, создатель *кибернетики* – науки об управлении и связи – писал: «Информация есть информация, а не материя и не энергия».

## I.1.3. Виды информации

Человек получает информацию через свои органы чувств: глаза, уши, рот, нос и кожу. Поэтому всю получаемую нами информацию можно разделить на следующие виды:

- *зрительная информация* (визуальная, от англ. *visual*)**,** которая поступает через глаза (по разным оценкам, 80-90% всей получаемой нами информации);
- *звуковая информация* (аудиальная*,* от англ. *audio*);
- *вкусовая информация* (вкус);
- *запахи* (обонятельная информация);
- *тактильная информация*, которую мы получаем с помощью осязания, «на ощупь».

Зафиксированная (закодированная) каким-то способом информация может быть представлена в различных *формах*:

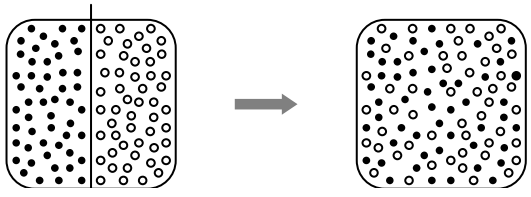
- *символ* (буква, цифра, знак) – самый простой вид информации;
- *текст*, который строится из символов; в отличие от набора символов, в тексте важен порядок их расположения, например, КОТ и ТОК – два разных текста, хотя они состоят из одинаковых символов;
- *числовая информация* (иногда ее не считают отдельным видом информации, полагая, что число – это текст специального вида, состоящий из цифр);
- *графическая информация* (рисунки, картины, чертежи, карты, схемы, фотографии);
- *звуковая информация* (звучание голоса, мелодии, шум, стук, шорох и т.п.);
- *комбинированная информация*, которая объединяет несколько видов информации (например, видеоинформация).

Обратим внимание, что одна и та же информация может быть представлена по-разному. Например, результаты измерения температуры в течение недели можно сохранить в виде текста, таблицы, графика, диаграммы, видеофильма и т.д.

## I.1.4. Информация в природе

В науках, изучающих **неживую природу** (прежде всего, в физике), информацию связывают со сложностью объекта. Чем разнообразнее и сложнее объект, тем больше информации он содержит и тем большее количество знаков необходимо для того, чтобы его описать.

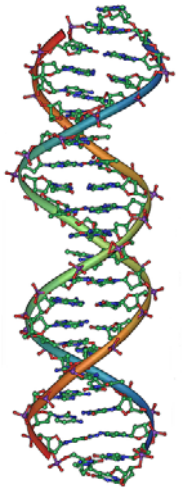
В замкнутых системах, которые не обмениваются веществом и энергией с окружающей средой, постепенно устанавливается состояние равновесия или хаоса. Все становится однородным, поэтому количество информации уменьшается. Например, если соединить две части сосуда, заполненного разными газами, эти газы постепенно перемешаются, и произойдет переход от порядка к хаосу[2].




[2] Если считать, что наша Вселенная – замкнутая система, можно сделать вывод о том, что ее ждет «тепловая смерть» – молекулы вещества равномерно распределятся в пространстве, и наступит полный хаос, развитие закончится. Однако современная теоретическая физика считает, что такого не произойдет благодаря действующим силам тяготения, которые заставляют Вселенную расширяться или сжиматься.

Порядок (и количество информации) не может увеличиваться «просто так», для этого необходим приток энергии и вещества. Системы, обладающие этим свойством, называются *открытыми*. Например, планета Земля – это открытая система, она получает энергию (и вместе с ней информацию) от Солнца. Согласно одной из теорий[3], молекулы вещества – это природные ячейки памяти. Под действием солнечного света молекулы переходят в возбуждённое состояние (в ячейки памяти записывается информация), это и служит причиной всех процессов, происходящих на Земле. При этом увеличивается сложность строения вещества (например, из атомов водорода и кислорода образуются молекулы воды). Это значит, что увеличивается также и количество накопленной информации.

Еще бóльшую роль играет информация в **живой природе**. Даже простейшие растения (синие водоросли) и животные (амебы) могут обрабатывать информацию о химическом составе и температуре окружающей среды и приспосабливаться к изменяющимся условиям. Более высокоразвитые животные обмениваются звуковой информацией (например, токование глухаря), зрительной (позы собаки, кошки), обонятельной (запахи, в том числе на больших расстояниях). Животные используют информацию на уровне инстинктов для того, чтобы выжить, избежать опасности, продолжить род.

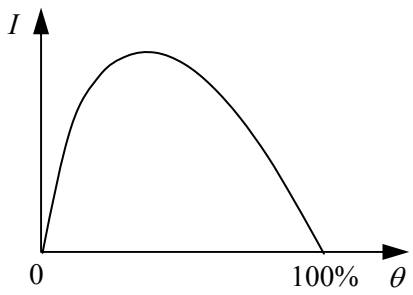

Молекула ДНК

Наследственная информация растений и животных, определяющая строение, внешний вид, предрасположенность к болезням, хранится и передается из поколения в поколение с помощью молекул ДНК (дезоксирибонуклеиновой кислоты). В этих молекулах информация закодирована в виде цепочек, составленных из четырех химических веществ (они называются *аденин*, *гуанин*, *тимин* и *цитозин*). Современный уровень развития биологии позволяет с помощью ДНК клонировать организмы, то есть создавать точные копии из одной клетки-образца.

## I.1.5. Человек, информация, знания

Обо всех изменениях в окружающем мире человек узнает с помощью своих органов чувств: сигналы от них («первичная» информация) постоянно поступают в мозг. Там они обрабатываются с учетом уже имеющегося у этого человека опыта; он дополняет уже имеющиеся у него представления о мире, накапливает *знания* – свои представления о природе, обществе, самом себе. Знания позволяют человеку принимать решения, определяют его поведение и отношения с другими людьми.

Всегда ли полученная информация увеличивает наши знания? Очевидно, что нет. Например, информация о том, что 2·2 = 4 вряд ли увеличит ваши знания, потому что вы это уже знаете, эта информация не нова. Однако, она будет новой для тех, кто изучает таблицу умножения. Это значит, что изменение знаний при получении сообщения зависит от того, что человек знал до этого момента. Если он знает все, что было в полученном сообщении, знания не изменяются.




С другой стороны, сообщение о том, что «учет вибронных взаимодействий континуализирует моделирование диссипативных структур» (или сообщение на неизвестном языке) также не увеличивает знания, потому что эта фраза вам не-

[3] http://thd.pnpi.spb.ru/~makariev/ec02.pdf

понятна. Иначе говоря, имеющихся знаний не хватило для того, чтобы воспринять новую информацию.

Эти идеи послужили основой *семантической* (смысловой) теории информации, предложенной в 1960-х годах советским математиком Ю.А. Шрейдером. На рисунке показано, как зависит количество полученных знаний $I$ от того, какая доля информации $\theta$ в сообщении уже известна получателю.

Сообщение увеличивает знания человека, если оно понятно и содержит новые сведения.

К сожалению, «измерить» смысл информации, оценить его числом, довольно сложно. Поэтому для оценки количества информации используют другие подходы, о которых вы узнаете чуть позже.

Когда человек хочет поделиться с кем-то своим знанием, он может сказать «Я знаю, что…» или «Я знаю, как…». Это говорит о том, что есть два разных вида знаний. В первом случае знания – это некоторый известный факт, например, «я знаю, что Луна вращается вокруг Земли». Такие знания называются **декларативными**, человек выражает их словами (*декларирует*). Декларативные знания – это факты, законы, принципы.

Второй тип знаний («Я знаю, как…») называют **процедурными**. Они выражаются в том, что человек *умеет* выполнять некоторые действия. Самое интересное, что объяснить эти знания бывает довольно сложно (например, попробуйте объяснить, как кататься на велосипеде).

Для того, чтобы сохранить знания и передать другим людям, нужно выразить их на каком-то языке (например, рассказать, записать, нарисовать и т.п.). Такие сведения можно хранить, обрабатывать, передавать, причем с этим может справиться и компьютер. В научной литературе информацию, зафиксированную (закодированную) в какой-то форме, называют *данными*, имея в виду, что компьютер может выполнять с ними какие-то операции, но не способен понимать смысл.

Данные – это просто какие-то зафиксированные сигналы, которые никак и никем не используются, не помогают решать какие-то задачи. Для того, чтобы данные стали информацией, их нужно понять и осмыслить, а на это способен пока только человек. Если человек, получающий сообщение, знает язык, на котором оно записано, он может понять смысл этого сообщения, то есть, получить информацию. Обрабатывая и упорядочивая информацию, он получает знания.

Мы увидели, что в науке существуют достаточно тонкие различия между понятиями «данные», «информация», «знания». Тем не менее, на практике чаще всего все это называется общим термином «информация».

## I.1.6. Информация в обществе

Очень велика роль информации в человеческом обществе. Информация, получаемая нами из разных источников, позволяет принимать решения и во многом определяет всю нашу жизнь. Огромно влияние на общество средств массовой информации (СМИ) – газет, телевидения, изданий в Интернете. Их часто называют «четвертой властью», сравнивая с остальными тремя ветвями власти: законодательной, исполнительной и судебной.

В идеале информация должна быть

- *объективной* (не зависящей от чьего-либо мнения)**;**
- *понятной* для получателя;
- *полезной* (позволяющей получателю решать свои задачи) ;
- *достоверной* (истинной, соответствующей реальной ситуации);
- *актуальной* (значимой в данный момент);
- *полной* (достаточной для принятия решения).

К сожалению, информация не всегда обладает всеми этими свойствами. Сообщение «В стакане мало молока» необъективно (для пессимиста полстакана – это мало, а для оптимиста – много). Сообщение 私は散歩に行った。непонятно для нас (оно означает «Я пошел гулять», только по-японски).

Полезность информации определяется для каждого человека в конкретной ситуации. Например, информация о том, как древние люди добывали огонь, для большинства городских жителей бесполезна, поскольку она никак не помогает им решать свои жизненные проблемы. С другой стороны, в экстремальной ситуации, когда человек оказывается один на один с природой, такие знания очень полезны, потому что сильно увеличивают шансы на выживание, то есть, помогают достичь цели.

Слухи, байки, искаженная информация (в том числе дезинформация) – это примеры недостоверной информации. Сообщение «10 лет назад тут был ларек с мороженым» неактуально, эта информация устарела. Информация «Сегодня будет концерт» неполна, потому что не указано время и участники концерта, поэтому мы не можем принять решение (идти или не идти?). Необъективная, неполная и недостоверная информация нередко используется в СМИ для продвижения идей и интересов отдельных личностей и групп.

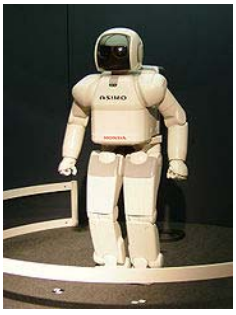
В будущем, по-видимому, нас ждет переход к информационному обществу, где бóльшая часть населения будет заниматься сбором, обработкой и распространением информации, поэтому высказывание немецкого банкира Н. Ротшильда «Кто владеет информацией, тот владеет миром[4]» становится актуально как никогда.

Развитие глобальной сети Интернет, в которую ежеминутно вносится огромное количество самых разнообразных данных, во многом перевернуло привычные представления о работе с информацией. Например, основным источником для поиска учебных материалов теперь фактически является Интернет, а не библиотеки. Однако при использовании информации из Интернета необходимо относиться к ней критически, так как ее достоверность никто не гарантирует.

## I.1.7. Информация в технике

Практически все современные технические устройства (телевизоры, телефоны, стиральные машины, системы управления самолетами и судами) строятся на микропроцессорах, которые обрабатывают информацию: анализируют сигналы с датчиков, выбирают нужный режим работы. Широко используются *системы программного управления*, например, станки, обрабатывающие детали по программе, заложенной в памяти. Эту программу очень легко поменять и настроить станок на изготовление другой детали.

Во многих ситуациях, особенно при выполнении опасных, тяжелых и утомительных работ, человека могут заменить роботы, которые имеют датчики, заменяющие органы чувств. Одним из наиболее совершенных считается человекоподобный робот (*андроид*) *Asimo*, разработанный фирмой *Honda*. Он умеет распознавать предметы, жесты, звуки, узнавать лица, разговаривать через домофон, передавать информацию через Интернет.



Робот Asimo

Наиболее универсальным устройством для обработки информации можно считать компьютер. Хотя современные компьютеры пока не умеют работать с вкусовой и обонятельной информацией (запахом), работы в этом направлении ведутся. Уже существуют экспериментальные приборы, названные

[4] http://www.newsfrol.com/internet/page_3.html

«электронный нос»[5] и «электронный язык»[6]; они построены на основе химических датчиков.

Сейчас в теоретической информатике считается, что компьютер может хранить и обрабатывать только *данные*, но не информацию. Многие ученые считают, что машина принципиально не может научиться понимать смысл информации и делать выводы. В пользу такой точки зрения свидетельствуют фактический провал проекта «компьютеров пятого поколения»[7] (Япония, 1980-е гг.), в ходе которого планировалось создать машины, общающиеся с человеком на естественном языке. Тем не менее, ученые уделяют этим проблемам огромное внимание. Например, возникло целое научное направление *data mining* («добыча данных»), в котором изучаются методы извлечения информации («смысла», закономерностей, связей, знаний) из огромных наборов данных. В некоторых случаях действительно удается использовать огромные вычислительные мощности компьютеров для того, чтобы найти неизвестные ранее закономерности, которые можно использовать на практике.

## ? *Контрольные вопросы*

1. Что изучает информатика?
2. Какие научные направления обычно включают в информатику?
3. Чем занимается кибернетика?
4. Что такое искусственный интеллект?
5. Как связана неопределенность наших знаний с получением информации?
6. Как связана информация и сложность объекта?
7. Объясните, почему термин «информация» трудно определить?
8. Какие еще базовые понятия науки вы знаете?
9. Основоположником какого научного направления был Норберт Винер?
10. Согласны ли вы с определением информации, которое дал Н. Винер?
11. Как человек воспринимает информацию?
12. Как можно классифицировать информацию? Какие существуют подходы?
13. Чем отличается текст от набора символов?
14. Почему числовую информацию иногда не выделяют как отдельный вид?
15. К какому виду информации относятся видеофильмы?
16. Что такое тактильная информация?
17. Что такое «информация» в неживой природе?
18. В чем разница между замкнутыми и открытыми системами?
19. Что такое «тепловая смерть Вселенной»? Что думают современные физики по этому поводу?
20. Какую роль играет информация в живой природе?
21. Как передается наследственная информация растений и животных?
22. Как кодируется наследственная информация в молекуле ДНК?
23. Что такое «клонирование»?
24. Как вы думаете, почему клонирование человека запрещено во многих странах, в том числе и в России?
25. Всякая ли информация увеличивает знания? Почему?
26. Что такое семантическая теория информации?
27. Приведите примеры своих декларативных и процедурных знаний.

[5] http://www.rg.ru/2007/06/26/nos.html
[6] http://www.business-magazine.ru/ideas/tech/pub282851
[7] http://ru.wikipedia.org/wiki/Компьютеры_пятого_поколения

28. В чем, на ваш взгляд, разница между понятиями «данные», «информация», «знания»?
29. Почему считают, что компьютер может работать только с данными?
30. Какова роль информации в жизни общества?
31. Какими свойствами должна обладать «идеальная» информация?
32. Приведите примеры необъективной, непонятной, бесполезной, недостоверной, неактуальной и неполной информации.
33. Может ли информация быть достоверной, но бесполезной? достоверной, но необъективной? объективной, но недостоверной? актуальной, но непонятной?
34. Какое общество называют информационным?
35. Какие изменения произошли в жизни общества в результате широкого распространения Интернета?
36. Приведите примеры анализа и обработки информации в технических устройствах.
37. Что умеет робот *Asimo*? Какую информацию он обрабатывает?
38. Что такое «электронный нос» и «электронный язык»?
39. Как вы считаете, смогут ли компьютеры научиться понимать смысл данных?

# I.2. Информационные процессы

Как мы уже знаем, информация сама по себе нематериальна. Но она может существовать только тогда, когда связана с каким-то объектом или средой, то есть с *носителем*.

**Материальный носитель** – это объект или среда, которые могут содержать информацию.

Изменения, происходящие с информацией (то есть изменения свойств носителя), называются *информационными процессами.* Обычно выделяют **три основных информационных процесса**:

- *хранение информации,*
- *передача информации,*
- *обработка информации.*

Часто информационными процессами называют также и многие другие операции с информацией (например, получение, представление, копирование, удаление и др.), но они, в конечном счете, сводятся к трем названным процессам.

## I.2.1. Хранение информации

Для хранения информации человек, прежде всего, использует свою память. Можно считать, что мозг – это одно из самых совершенных хранилищ информации, во многом превосходящее компьютерные средства. Для запоминания и поиска информации используются нервные клетки мозга – *нейроны*, их более ста миллиардов.

К сожалению, человек многое забывает. Поэтому в древности он записывал информацию на камне, папирусе, бересте, пергаменте, а сейчас – на бумаге, магнитной ленте, электронных носителях. Это нужно еще и для того, чтобы передать знания другим людям, в том числе и следующим поколениям.

В XX веке появились новые средства хранения информации: перфокарты и перфоленты, магнитные ленты и магнитные диски, лазерные диски, флэш-память.

В любом случае информация хранится на каком-то *носителе*, который обладает «памятью», то есть способен сохранять свое состояние. При записи информации свойства носителя меняются: на бумагу наносятся текст и рисунки; на магнитных дисках и лентах намагничиваются отдельные участки; на лазерных дисках образуются области, по-разному отражающие свет. Во время хранения эти свойства сохраняются, что позволяет «читать» записанную информацию.

Запись и чтение информации – это *процессы*, потому что при этом изменяются свойства носителя. При самом хранении никаких изменений не происходит, поэтому, строго говоря, процессом его назвать нельзя.

## I.2.2. Передача информации

При **передаче информации** всегда есть два объекта – источник и приемник информации. Эти роли могут меняться, например, во время диалога каждый из участников выступает то в роли источника, то в роли приемника информации.

Информация проходит от источника к приемнику через канал связи, в котором она должна быть связана с каким-то *материальным носителем*. Для передачи информации свойства этого носителя должны изменяться со временем. Так лампочка, которая все время горит, передает информацию только о том, что какой-то процесс идет. Если же включать и выключать лампочку, можно передавать самую разную информацию, например, с помощью азбуки Морзе.

При разговоре людей носитель информации – это звуковые волны в воздухе. В компьютерах информация передается с помощью электрических сигналов или радиоволн (в беспроводных устройствах). Информация может передаваться с помощью света, лазерного луча, системы телефонной или почтовой связи, компьютерной сети и др.


Информация поступает по каналу связи в виде сигналов, которые приемник может обнаружить с помощью своих органов чувств (или датчиков) и «понять» (раскодировать).

**Сигнал** – это изменение свойств носителя, которое используется для передачи информации.

Примеры сигналов – это изменение частоты и громкости звука, вспышки света, изменение напряжения на контактах и т.п.

Человек может принимать сигналы только с помощью своих органов чувств. Чтобы передавать информацию, например, с помощью радиоволн, нужны вспомогательные устройства: радиопередатчик, преобразующий звук в радиоволны, и радиоприемник, выполняющий обратное преобразование. Они позволяют расширить возможности человека.

С помощью одного сигнала невозможно передать много информации. Поэтому чаще всего используется не одиночный сигнал, а последовательность сигналов, то есть *сообщение*. Важно понимать, что сообщение – это только «оболочка» для передачи информации, а информация – это *содержание* сообщения. Приемник должен сам «извлечь» информацию из полученной последовательности сигналов. Можно принять сообщение, но не принять информацию, например, услышав речь на незнакомом языке или перехватив шифровку.

Одна и та же информация может быть передана с помощью разных сообщений, например, через устную речь, с помощью записки или с помощью флажного семафора, который используется на флоте. В то же время одно и то же сообщение может нести разную информацию для разных приемников. Так фраза «В Сантьяго идет дождь», переданная в 1973 году на военных радиочастотах, для сторонников генерала А. Пиночета послужила сигналом к началу государственного переворота в Чили.

К сожалению, в реальном канале связи всегда действуют помехи: посторонние звуки при разговоре, шумы радиоэфира, электрические и магнитные поля. Помехи могут полностью или частично искажать сообщение, вплоть до полной потери информации (вспомните телефонные разговоры при перегрузке сети).

Чтобы содержание сообщения, искаженного помехами, можно было восстановить, оно должно быть *избыточным*, то есть, в нем должны быть «лишние» элементы, без которых смысл все равно восстанавливается. Например, в сообщении «Влг впдт в Кспск мр» многие угадают фразу «Волга впадает в Каспийское море», из которой убрали все гласные. Этот пример говорит о том, что естественные языки содержат много «лишнего», их избыточность оценивается в 60-80%.

## I.2.3. Обработка информации

**Обработка** – это любое изменение информации, причем изменяться может как содержание информации, так и ее форма представления.

Можно выделить четыре важнейших вида обработки:

- *создание новой информации*, например, решение задачи с помощью вычислений или логических рассуждений;
- *кодирование*, когда меняется *форма* (внешний вид), но не содержание информации; например, перевод текста на другой язык; один из видов кодирования – шифрование, цель которого – скрыть смысл (содержание) информации от посторонних;
- *поиск информации*, например, в книге, в библиотечном каталоге, на схеме или в Интернете;
- *сортировка* – расстановка элементов списка в заданном порядке, например, расстановка чисел по возрастанию или убыванию, расстановка слов или фамилий по алфавиту; одна из задач сортировки – облегчить поиск информации.

Для обработки информации человек использует, в первую очередь, свой мозг. Нейроны коры головного мозга «переключаются» примерно 200 раз в секунду – значительно медленнее, чем элементы памяти компьютеров. Однако человек практически безошибочно отличает собаку от кошки, а для компьютеров эта задача пока не разрешима. Дело, по-видимому, в том, что мозг решает такие задачи не «в лоб», не путем сложных вычислений, а как-то иначе (как – пока никто до конца не знает).

Компьютер позволяет «усилить» возможности человека в тех задачах обработки информации, решение которых требует длительных расчетов по известным алгоритмам. Однако, в отличие от человека, компьютер не может «мыслить» образами, поэтому для него недоступно фантазии, размышления, творчество.

### ? *Контрольные вопросы*

1. К каким из основных информационных процессов можно, на ваш взгляд, отнести получение, представление, копирование, удаление информации?
2. Зачем человек записывает информацию?
3. В чем преимущества и недостатки человеческой памяти в сравнении с компьютерной?
4. В каких задачах компьютер не может соревноваться с человеком? Почему? В каких ситуациях человек явно уступает компьютеру?
5. Какие средства хранения информации используются в компьютерной технике? Какие из них уже вышли или выходят из употребления? Почему?
6. Кто (что) может быть источником (приемником) информации? Приведите примеры.
7. Какие каналы связи могут использоваться для передачи информации? Приведите примеры из жизни, из литературных произведений.
8. Что такое сигнал? Приведите примеры сигналов.
9. Что такое сообщение? Чем отличается получение информации от получения сообщения?
10. Приведите примеры, когда прием сообщения не означает прием информации.

11. Приведите примеры, когда одна и та же информация может быть передана с помощью разных сообщений.
12. Приведите примеры, когда одно и то же сообщение несет разную информацию для разных людей.
13. Расскажите, как помехи влияют на передачу информации. Приведите примеры.
14. Что такое избыточность? Почему она полезна при передаче информации?
15. Представьте, что придумали язык, в котором нет избыточности. В чем будет его недостаток?
16. Как вы думаете, какой вариант русского языка обладает наибольшей избыточностью: разговорный, литературный, юридический, язык авиадиспетчеров? (Оценки[8]: 72%, 76%, 83%, 96%)
17. Какие основные виды обработки информации вы знаете?
18. При каких видах обработки информации меняется ее содержание? форма?
19. При каких видах обработки информации меняется только ее форма представления?
20. К какому виду обработки можно отнести шифрование? Почему?
21. Работники удаленной метеостанции каждый 3 часа измеряют температуру и влажность воздуха, и передают данные по рации в районный метеоцентр. Там эти данные сводят в таблицу и отправляют по электронной почте в Гидрометцентр, где мощные компьютеры составляют прогноз погоды. Выделите здесь информационные процессы.
22. Вася Пупкин нашел в старой книге сведения о населении Москвы в XIX веке, составил таблицу по этом данным, построил диаграмму и сделал доклад на школьной конференции. Выделите здесь информационные процессы.

# I.3. Измерение информации

Любая наука рано или поздно приходит к необходимости как-то измерять то, что она изучает. Измерение информации – это одна из важнейших задач теоретической информатики.

Для человека информация – это, прежде всего, смысл, заключенный в сигналах и данных. Как измерить смысл? На этот вопрос пока нет однозначного ответа.

Вспомним, что компьютеры не могут обрабатывать смысл, они работают только с данными. При этом возникают чисто практические задачи – определить, сколько места займет на диске текст, рисунок или видеофильм; сколько времени потребуется на передачу файла по компьютерной сети и т.п. Поэтому чаще всего используется *объемный* подход к измерению информации. Он заключается в том, что количество информации оценивается просто по числу символов, используемых для ее кодирования. С этой точки зрения стихотворение А.С. Пушкина и случайный набор букв могут содержать одинаковое количество информации. Конечно, этот подход далек от идеала, но он позволяет успешно решать все задачи, связанные с компьютерной обработкой и хранением данных.

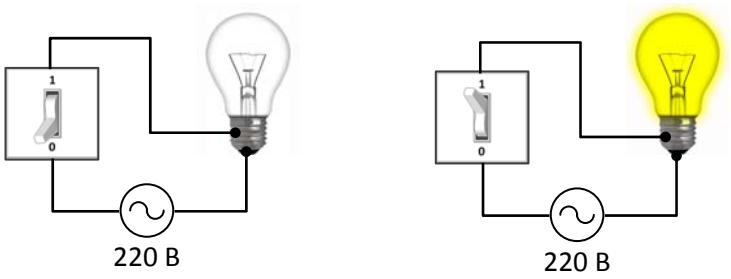
## I.3.1. Что такое бит?

Чтобы измерить информацию, нужно выбрать какую-то единицу измерения, эталон. В качестве такого эталона принимают информацию, полученную при выборе одного из двух вариантов.

Например, электрическая лампочка может находиться в двух состояниях: «горит» и «не горит». Тогда на вопрос «Горит ли сейчас лампочка» есть два возможных варианта ответа, которые можно обозначить цифрами 1 («горит») и 0 («не горит»). Поэтому ответ на этот вопрос может быть записан как 0 или 1[9].

[8] http://habrahabr.ru/blogs/popular_science/50643/

[9] Конечно, вместо 0 и 1 можно использовать два любых знака.




Цифры 0 и 1 называют *двоичными* (в отличие, например, от набора десятичных цифр), и с этим связано название единицы измерения количества информации – *бит*. Английское слово *bit* – это сокращение от выражения *binary digit*, «двоичная цифра». Впервые слово *бит* в этом значении использовал американский инженер и математик Клод Шеннон в 1948 г.

**Бит** – это количество информации, соответствующее выбору одного из двух равновозможных вариантов.

Например, в сообщении «подброшенная монета упала гербом» содержится 1 бит информации, потому что до опыта было два возможных варианта: монета могла упасть гербом или «решкой». Ответ на вопрос «Дверь открыта?» тоже содержит 1 бит, если считать, что возможны только два ответа, «да» или «нет». Вот пример диалога, в котором получена информация в 1 бит:

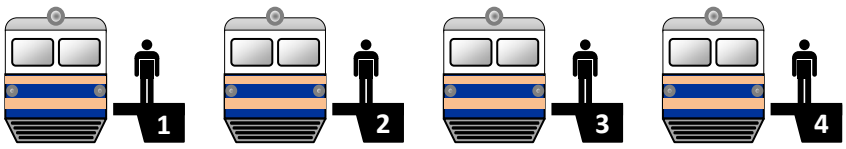
– Вы будете чай или кофе?

– Кофе, пожалуйста.

Слово «равновозможных» означает, что ни одному из этих вариантов заранее нельзя отдать предпочтение (как говорят математики, они *равновероятны*). Это условие будет нарушено, например, когда летом в Воронеже кто-то отвечает на вопрос «Идет ли сейчас снег?» (можно почти с полной уверенностью заранее сказать, что летом в Воронеже снега нет).

## I.3.2. 2 бита, 3 бита, …

А если возможных вариантов не два, а больше? Понятно, что в этом случае информация будет больше, чем 1 бит. Представим себе, что на вокзале стоят 4 одинаковых поезда, причем только один из них идет в Москву. В справочном окне сидит робот, который умеет отвечать на вопросы, но может сказать только «да» или «нет». Как за наименьшее количество вопросов найти поезд в Москву?




Понятно, что можно перебрать все поезда, задавая вопросы типа «Поезд на платформе 1 идет в Москву?» и т.д. В лучшем случае мы угадаем с первого раза, а в худшем (если нужный поезд стоит на платформе 3 или 4) придется задать три вопроса.

Однако есть другой способ, который позволяет за два вопроса гарантированно найти поезд в Москву. Разделим все поезда на две равные группы (например, включим в первую поезда на платформах 1 и 2, а во вторую – все остальные) и спросим: «В первой группе есть поезд в Москву?». В ответ мы получим 1 бит информации, поскольку сделан выбор из двух равновозможных вариантов. При любом ответе («да» или «нет») количество вариантов уменьшается вдвое. Теперь

останется одним вопросом выбрать один из двух оставшихся поездов, при этом мы тоже получим 1 бит информации. Таким образом, всего за 2 ответа получено 2 бита информации[10].

Теперь предположим, что в справочном окне работает человек, способный дать полный ответ. Тогда на вопрос «Какой поезд идет в Москву?» он может ответить, например, «В Москву идет поезд с платформы № 2». Сколько информации мы получили в этом сообщении? Очевидно, что мы не узнали ничего нового в сравнении с первым случаем (когда робот отвечал «да» или «нет»). Поэтому количество информации будет равно 2 бита, независимо от того, как мы ее получили. Таким образом, при выборе одного из четырех вариантов (любым способом) мы получаем 2 бита информации.

Для большего количества вариантов тоже можно применить принцип «деления на два». При поиске одного из 8 вариантов мы сначала выбираем «четверку», где находится нужный вариант (1 бит информации), потом – пару (еще один бит), а затем находим сам этот вариант (третий бит). Поэтому всего мы получаем 3 бита информации.

Таким образом, при каждом удвоении количества возможных вариантов мы должны задать дополнительный вопрос и, следовательно, получить еще один бит информации. При выборе одного из 16 вариантов мы получаем 4 бита информации, при выборе из 32 вариантов – 5 бит и т.д. (см. таблицу).

| $I$, бит | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $N$, вариантов | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |

Наверное, вы заметили, что все числа в нижней строчке таблицы – это степени числа 2.

Осталось выяснить, чему равно количество информации, если выбор делается, скажем, из 6 возможных вариантов (или **вообще** для любого числа, не являющегося степенью числа 2). Несложно догадаться, что это не целое число. Например, при выборе из 6 вариантов количество информации будет между 2 и 3 битами. Более точно мы определим это значение в 11 классе, когда вы изучите логарифмы.

## I.3.3. Другие единицы

Считать большие объемы информации в битах неудобно, хотя бы потому, что придется работать с очень большими числами (миллиардами, триллионами, и т.д.). Поэтому стоит ввести более крупные единицы.

Измерение количества информации тесно связано с устройством компьютерной памяти. Память строится из элементов, которые могут находиться в двух состояниях (0 или 1, включено или выключено). Поэтому информация о состоянии такого элемента равна 1 биту. Чтобы обращаться к ячейкам памяти, нужно каждой из них присвоить адрес (номер). Если каждый отдельный бит будет иметь свой адрес, адреса будут очень большие, и для их хранения потребуется много места. Кроме того, реальные данные состоят из нескольких битов, и каждый раз «собирать» число или символ из нескольких отдельных ячеек памяти неудобно. Поэтому группы соседних битов памяти объединяют в ячейки, каждая из которых имеет свой адрес и считывается (или записывается) как единое целое[11]. Такие ячейки называются **байтами**.

[10] Важно, что мы каждый раз разбиваем оставшиеся поезда на две равные группы. Если это не так, количество информации в битах не будет совпадать с количеством ответом «да»-«нет», потому что варианты не будут равновозможными. Нужный нам поезд, скорее всего, окажется в той группе, где поездов больше.

[11] Компьютеры первых поколений обрабатывали только числа, для хранения которых выделялись ячейки размером от 24 до 60 бит (они часто назывались *машинными словами*). Позднее, когда на компьютерах начали обрабатывать в основном текстовую информацию, большие ячейки стало неудобно использовать,

**Байт** – это группа битов, имеющая собственный адрес в памяти. В современных компьютерах[12]

**1 байт = 8 бит**

Слово «байт» (англ. *byte*) впервые использовал американский инженер В. Бухгольц в 1956 г. для обозначения наименьшего объема данных, который компьютер может «откусить» (англ. *bite*) за один раз.

Байт – тоже очень небольшая единица количества информации. Для того, чтобы не работать с большими числами, часто используют более крупные единицы, образованные с помощью приставок:

**1 Кбайт (килобайт) = 1024 байта = $2^{10}$ байта = $2^{13}$ бит**
**1 Мбайт (мегабайт) = 1024 Кбайта = $2^{10}$ Кбайта = $2^{20}$ байта = $2^{23}$ бит**
**1 Гбайт (гигабайт) = 1024 Мбайта**
**1 Тбайт (терабайт) = 1024 Гбайта**

Так сложилось исторически, что при измерении количества информации приставка «кило-» обозначает, в отличие от международной системы единиц СИ, увеличение не в 1000 раз, а в $1024 = 2^{10}$ раза. Аналогично «мега-» – это увеличение в $1024^2 = 2^{20} = 1048576$ раза, а не в 1 млн $= 1000^2$ раз.

Строго говоря, нужно называть такие кило-(мега-, гига-, …)байты *двоичными*, поскольку множитель 1024 – это $2^{10}$. Стандарт Международной электротехнической комиссии (МЭК) предлагает называть их «кибибайт», «мебибайт», «гибибайт» и «тебибайт», но эти названия не прижились.

Для перевода количества информации из одних единиц в другие нужно использовать приведенные выше соотношения. При переводе из крупных единиц в мелкие числа умножают на соотношение между единицами. Например,

2 Кбайта = 2·(1 Кбайт) = 2·1024 байта = 2048 байт
= 2048·(1 байт) = 2048·8 бит = 16384 бита
= $2 \cdot 2^{10}$ байт = $2^{11}$ байт = $2^{11} \cdot 2^3$ бит = $2^{14}$ бит

В последней строке все расчеты сделаны через степени числа 2, очень часто так бывает проще.

При переводе из мелких единиц в крупные нужно делить на соотношение между единицами. Например,

8192 бита = 8192·(1/8 байта) = 8192:8 байт = 1024 байта
= 1024·(1/1024 Кбайта) = 1024:1024 Кбайт = 1 Кбайт
= $2^{13}$ бита = $2^{13} \cdot (1/2^3$ байта) = $2^{10}$ байт
= $2^{10} \cdot (1/2^{10}$ Кбайта) = 1 Кбайт

## ? *Контрольные вопросы*

1. Какую информацию принимают за единицу измерения количества информации?
2. Приведите примеры информации, равной 1 биту.
3. Что такое двоичные цифры?
4. Что такое равновозможные варианты? Приведите примеры, когда этой свойство нарушено.
5. Объясните, почему все числа во второй строке таблицы на предыдущей странице – это степени числа 2.

так как приходилось «разбирать» слова на отдельные символы («байты»). Поэтому изменили систему адресации, дав каждому байту свой адрес.

[12] Так было не всегда: см. http://ru.wikipedia.org/wiki/Байт.

6. Чему равен байт в современных компьютерах?
7. Какие единицы используют для измерения больших объемов информации?
8. Что означают приставки «кило-», «мега-», «гига-» и «тера-» при измерении количества информации?
9. Какие приставки рекомендует МЭК для обозначения двоичных килобайта и мегабайта?

## *Задачи*

1. Вася Пупкин не знает, какой из 8 поездов, стоящих на вокзале, идет в Санкт-Петербург. В справочном бюро он задает 8 вопросов: «Поезд на 1-ой платформе идет в Санкт-Петербург?», «Поезд на 2-ой платформе идет в Санкт-Петербург?» и т.д. На первые 7 вопросов он получает ответ «нет», а на последний – «да». Вася считает, что он получил 8 бит информации. Прав он или нет? Почему? (Ответ: не прав, в самом деле, он получил 3 бита информации)
2. В зоопарке содержится 10 обезьян, причем одна из них– альбинос (вся белая). Обезьяны сидят в двух вольерах, в первом – 8 штук, а во втором – две. Вася Пупкин считает, что сообщение «Обезьяна-альбинос сидит во втором вольере» содержит 1 бит информации. Прав он или нет? (Ответ: нет, варианты не равновероятны.)
3. На лугу пасутся 12 коров. Дедушка говорит Васе Пупкину, приехавшему из города в гости: «Вон та, крайняя справа – это наша Зорька». Сколько информации (в битах) получил Вася? (Ответ: больше 3, но меньше 4 бит)
4. Известно, что дверь открывается двумя из 4-х имеющихся ключей. Оцените количество информации в сообщении «Дверь открывается ключами № 2 и № 4». (Ответ: всего есть 6 вариантов, поэтому количество информации больше 2, но меньше 3 бит)
5. Известно, что дверь открывается двумя из 5 имеющихся ключей. Оцените количество информации в сообщении «Верхний замок открывается ключом № 1, а нижний – ключом № 4». (Ответ: всего есть 20 вариантов, поэтому количество информации больше 4, но меньше 5 бит)
6. Вася Пупкин задумал число от 1 до 100. Нужно отгадать это число за наименьшее число попыток, задавая Васе вопросы, на которые он отвечает только «да» и «нет». За сколько вопросов вы беретесь угадать число? Как нужно задавать вопросы, чтобы их число было минимальным даже в худшем случае?
7. Вася Пупкин задумал число от 20 до 83. Сколько бит информации содержится в сообщении «Вася задумал число 77»? (Ответ: 6 бит)
8. Двое играют в «крестики-нолики» на поле 4 на 4 клетки. Какое количество информации получил второй игрок, узнав ход первого игрока? (Ответ: 4 бита)
9. Что такое байт? Всегда ли в байте 8 бит?
10. Переведите 1 Мбайт во все изученные единицы измерения количества информации.
11. Переведите $2^{26}$ бит во все изученные единицы измерения количества информации.
12. Сколько килобайт содержится в 32768 битах?
13. Сколько бит в 8 Кбайтах?
14. Сколько бит содержит 1/16 Кбайта?
15. Сколько бит содержит 1/512 Мбайта?

# I.4. Структура информации

## I.4.1. Зачем структурировать информацию?

Давайте сравним четыре информационных сообщения.

Первое:

*«Для того, чтобы добраться до села Васино, нужно сначала долететь на самолете до Ивановска. Затем на электричке доехать до Ореховска. Там на пароме переправиться через реку Слоновую в поселок Ольховка, и оттуда ехать в Васино на попутной машине».*

Второе:

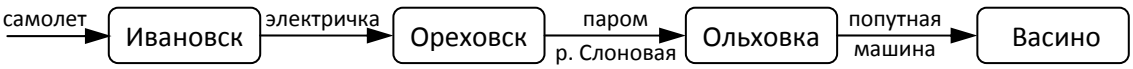
*Как ехать в Васино?*

1) *На самолете до Ивановска.*
2) *На электричке до Ореховска.*
3) *На пароме через р. Слоновую в пос. Ольховка.*
4) *На попутной машине до с. Васино.*

Третье:

| Откуда | Куда | Транспорт |
|---|---|---|
| Москва | Ивановск | самолет |
| Ивановск | Ореховск | электричка |
| Ореховск | пос. Ольховка | паром (р. Слоновая) |
| пос. Ольховка | с. Васино | попутная машина |

Четвертое:




Можно считать, что все эти (такие разные по форме!) сообщения содержат одну и ту же информации. Какие из них проще воспринимать? Очевидно, что «вытащить» полезную информацию из простого текста (первое сообщение) сложнее всего. Во втором случае мы сразу видим все этапы поездки. Третье сообщение (таблицу) и четвертое (схему) можно понять сразу, с первого взгляда. Второй, третий и четвертый варианты лучше и быстрее воспринимаются, потому что в них выделена *структура* информации, в которой самое главное – этапы поездки в Васино.

**Структура** (лат. *structura* — строение) – это внутреннее устройство чего-либо.

Почему книгу разбивают на главы и разделы, а не пишут сплошной текст? Зачем в тексте выделяют абзацы? Прежде всего, для того, чтобы подчеркнуть основные мысли – каждая глава, раздел, абзац содержат определенную идею. При таком выделении структуры улучшается передача информации от автора к читателю.

Кроме того, есть еще и другая причина – облегчить **поиск** нужной информации. В книгах чаще всего есть оглавление, позволяющее быстро найти нужный раздел. Слова в словарях всегда расставлены в алфавитном порядке (представьте себе, что было бы, если бы они были расположены произвольно!). В больших книгах используют *индексы* – списки основных терминов с указанием страниц, на которых они встречаются.

**Оглавление**:



**Словарь:**

автомат – *automaton*
автор – *author*
адрес – *address*
алгебра – *algebra*
алгоритм – *algorithm*
архив – *archive*
архитектура – *architecture*

**Индекс:**

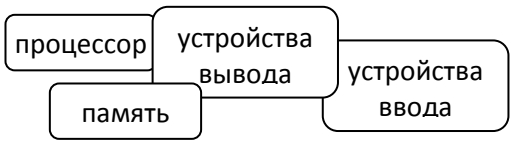
**Структурирование** – это выделение важных элементов в информационных сообщениях и установление связей между ними. Цели структурирования – облегчение восприятия и поиска информации, выявление закономерностей.

## I.4.2. Простые структуры

Простейшая структура – это **множество**, то есть, некоторый набор элементов. Чтобы определить множество, мы должны перечислить все его элементы (например, множество, состоящее из Васи, Пети и Коли) или определить характерный признак, по которому элементы включаются в это множество (множество драконов с пятью зелеными хвостами).

Множество может состоять из конечного числа элементов (множество букв русского алфавита), бесконечного числа элементов (множество натуральных чисел) или вообще быть пустым (множество слонов, живущих на Северном полюсе). В документах конечное множество часто оформляют в виде маркированного списка, например:

- процессор;
- память;
- устройство ввода;
- устройства вывода.

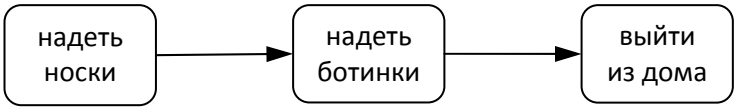



В таком списке порядок элементов не важен, от перестановки элементов множество не меняется.

Если множество состоит из конечного числа элементов и его элементы должны быть расположены в строго определенном порядке, мы получаем **линейный список**. Список обычно упорядочен (отсортирован) по какому-то правилу, например, по алфавиту, по важности, по последовательности действий и т.д. В тексте он оформляется как нумерованный список, например:

1) надеть носки;
2) надеть ботинки;
3) выйти из дома.

Переставить местами элементы такого списка нельзя (это будет уже другой список). Линейный список может быть представлен в виде цепочки связанных элементов:

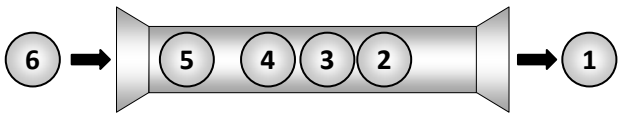



Список можно задать перечислением элементов, с первого до последнего:

(надеть носки, надеть ботинки, выйти из дома)

Теперь предположим, что нам нужно добавлять и удалять элементы в линейном списке. Знакомый вам пример – это *очередь* в кассу в магазине. Действительно, очередь – это цепочка, в которой элементы нельзя переставлять местами (если так случается, то кто-то лезет без очереди).

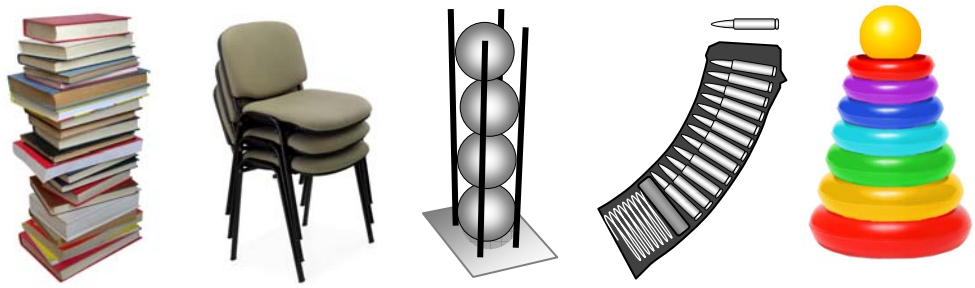
**Очередь** – это линейный список, в котором элементы добавляются с одного конца, а удаляются с другого («первым пришел – первым ушел»[13]).




[13] Англ. FIFO = *First In – First Out*.

Очередь можно представить в виде трубы, с одного конца которой добавляются шарики, а с другого – вынимаются. Трамваи, стоящие на перекрестке, – это тоже пример очереди, они не могут обогнать друг друга.

Возможен и другой вариант, когда элементы добавляются и удаляются с одного и того же конца списка. Это значит, что тот, кто пришел последним, уйдет первым. Такая структура называется **стек**. Например, стопка с книгами или автоматный магазин:



Во всех этих примерах для того, чтобы добраться до нужного объекта, нам нужно сначала удалить все те, что расположены выше него.

**Стек** – это линейный список, в котором элементы добавляются и удаляются только с одного конца («последний пришел – первым ушел»[14]).

Более общий случай – это список, в котором добавление и удаление элементов разрешается с обоих концов. Такая структура называется **дек.**

Еще одна знакомая вам структура – **таблица**. С помощью таблиц устанавливается связь между несколькими элементами. Например, в следующей таблице элементы в каждой строке связаны между собой – это свойства некоторого объекта (человека):

| Фамилия | Имя | Рост, см | Вес, кг | Год рождения |
|---|---|---|---|---|
| Иванов | Иван | 175 | 67 | 1996 |
| Петров | Петр | 164 | 70 | 1998 |
| Сидоров | Сидор | 168 | 63 | 2000 |

Именно так хранится информация в базах данных: строка таблицы, содержащая информацию об одном объекте, называется *записью*, а столбец (название свойства) – *полем*.

Возможен и другой вариант таблицы, когда роли строк и столбцов меняются. В первом столбце записываются названия свойств, а данные в каждом из следующих столбцов описывают свойства какого-то объекта. Например, вот таблица с характеристиками разных марок автомашин:

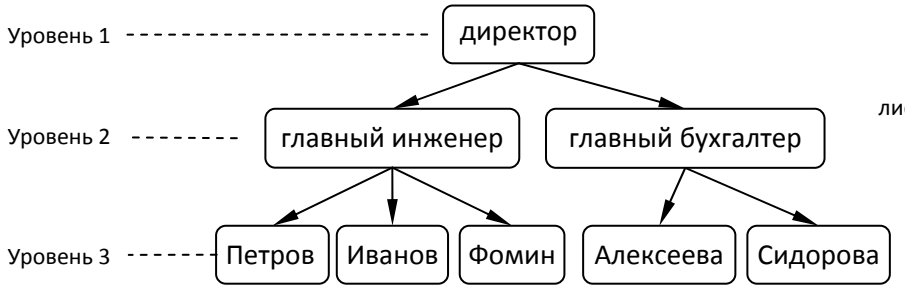
| Марка | Лада Приора | Лада Калина | ВАЗ 2110 | ВАЗ 21099 |
|---|---|---|---|---|
| Мощность, л.с. | 98 | 89 | 89 | 81 |
| Максимальная скорость, км/ч | 183 | 165 | 187 | 167,5 |
| Время разгона до 100 км/ч, с | 12 | 12,5 | 12 | 13,5 |

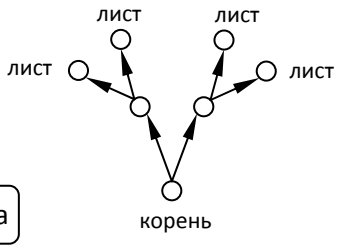
В математике и программировании таблицы обычно называют *матрицами*.

## I.4.3. Иерархия (дерево)

Линейных списков и таблиц иногда недостаточно для того, чтобы представить все связи между элементами. Например, в некоторой фирме есть директор, ему подчиняются главный инженер и главный бухгалтер, у каждого из них есть свои подчиненные. Если мы захотим нарисовать схему управления этой фирмы, она получится *многоуровневой*.

[14] Англ. LIFO = *Last In – First Out*.



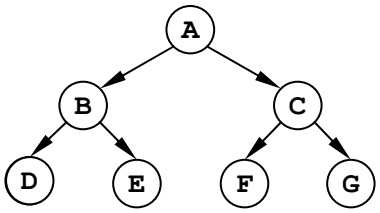




Такая структура, в которой одни элементы «подчиняются» другим, называется *иерархия* (от древнегреческого ἱεραρχία – «священное правление»). В информатике иерархию называют **деревом**. Дело в том, что если перевернуть эту схему вверх ногами, она становится похожа на дерево (точнее, на куст, см. рисунок справа). Несколько деревьев образуют **лес**.

Дерево состоит из узлов и связей между ними (они называются дугами). Самый первый узел, расположенный на верхнем уровне (в него не входит ни одна стрелка-дуга) – это *корень дерева.* Конечные узлы, из которых не выходит ни одна дуга, называются *листьями*. Все остальные узлы, кроме корня и листьев – это промежуточные узлы.

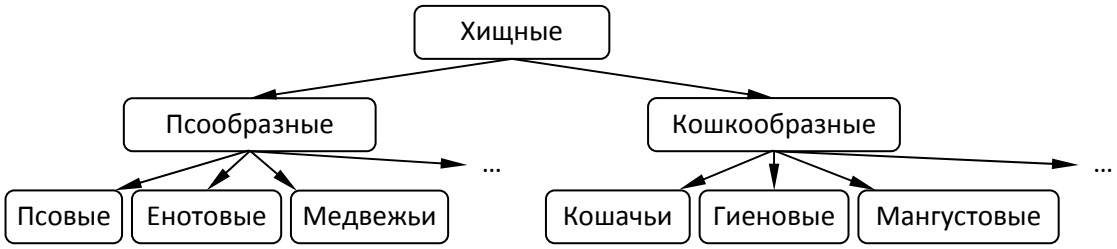
Из двух связанных узлов тот, который находится на более высоком уровне, называется «*родителем*», а другой – «*сыном*». Корень – это единственный узел, у которого нет «родителя»; у листьев нет «сыновей».

Используются также понятия «предок» и «потомок». «Потомок» какого-то узла – это узел, в который можно перейти по стрелкам от узла-предка. Соответственно, «предок» какого-то узла – это узел, из которого можно перейти по стрелкам в данный узел.




В дереве на рисунке справа родитель узла E – это узел B, а предки узла E – это узлы A и B, для которых узел E – потомок. Потомками узла A (корня) являются все остальные узлы.

Типичный пример иерархии – различные *классификации* (животных, растений, минералов, химических соединений). Например, отряд *Хищные* делится на два подотряда: *Псообразные* и *Кошкообразные*. В каждом из них выделяют несколько семейств:




Конечно, на этой схеме показаны не все семейства, остальные обозначены многоточием.

В текстах иерархию часто оформляют в виде многоуровневого списка. Например, оглавление книги о хищниках может выглядеть так:

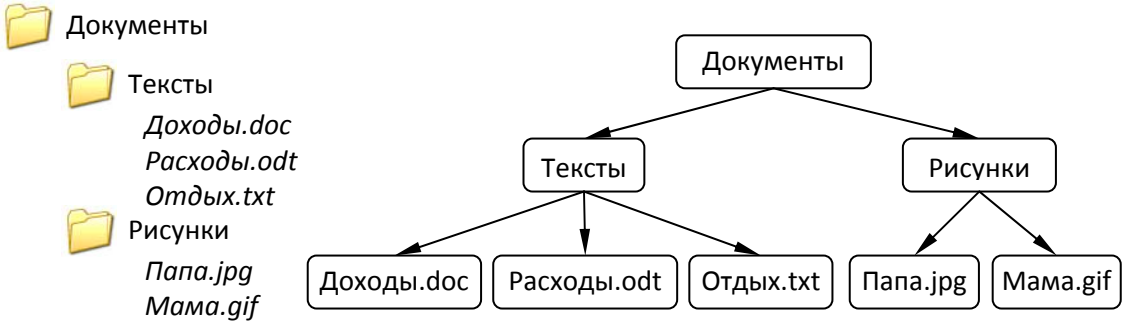
Глава 1. Псообразные
1.1 Псовые
1.2 Енотовые
1.3 Медвежьи
…
Глава 2. Кошкоообразные
2.1 Кошачьи

2.2 Гиеновые

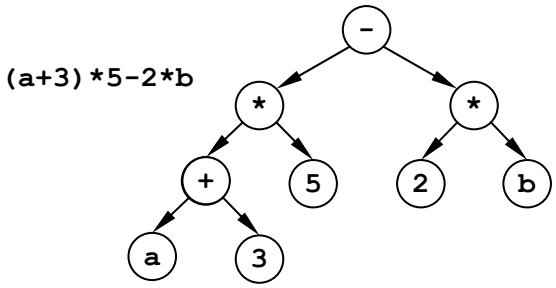
2.3 Мангустовые

…

С иерархией мы встречаемся, работая с файлами и папками: классическая файловая система имеет древовидную структуру[15]. Вход в папку – это переход на следующий (более низкий) уровень иерархии:




Алгоритм вычисления арифметического выражения тоже может быть представлен в виде дерева:




Здесь листья – это числа и переменные, тогда как корень и промежуточные узлы – знаки операций. Вычисления идут «снизу вверх», от листьев – к корню. Показанное дерево можно записать так:

```
(-(*(+(a,3),5),*(2,b)))
```

Самое интересное, что скобки здесь не обязательны; если их убрать, то выражение все равно может быть однозначно вычислено:

```
- * + a 3 5 * 2 b
```

Такая запись, которая называется *префиксной* (операция записывается *перед* данными), просматривается с конца. Как только встретится знак операции, эта операция выполняется с двумя значениями, записанными справа. В рассмотренном выражении сначала выполняется умножение:

```
- * + a 3 5 (2*b)
```

затем – сложение:

```
- * (a+3) 5 (2*b)
```

и еще одно умножение

```
- (a+3)*5 (2*b)
```

и, наконец, вычитание: `(a+3)*5-(2*b)`.

Для того, чтобы вычислять выражение не с конца, а с начала, префиксную форму «разворачивают» задом наперед, тогда получается *постфиксная* форма (операция *после* данных). Например, рассмотренное выше выражение может быть записано в виде

```
b 2 * 5 3 a + * -
```

[15] В современных файловых системах файл может «принадлежать» нескольким каталогам одновременно. При этом древовидная структура, строго говоря, нарушается.

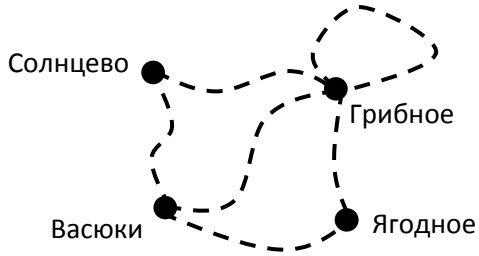
Для вычисления такого выражения скобки также не нужны, и это очень удобно для автоматических расчетов. Когда программа на языке программирования высокого уровня переводится в машинные коды, все выражения записываются в бесскобочной постфиксной форме и именно так и вычисляются.

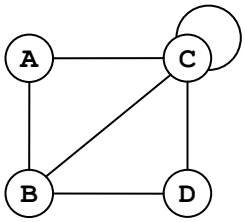
## I.4.4. Графы

Подумайте, как можно структурировать такую информацию:

*«От пос. Васюки три дороги идут в Солнцево, Грибное и Ягодное. Между Солнцевым и Грибным и между Грибным и Ягодным также есть дороги. Кроме того, есть дорога, которая идет из Грибного в лес и возвращается обратно в Грибное».*

Можно, например, нарисовать схему дорог:




В информатике такие схемы называются *графами*.

**Граф** – это набор узлов (вершин) и связей между ними (рёбер).

Для хранения информации об узлах и связях показанного выше графа (см. правый рисунок) можно использовать таблицу (матрицу) такого вида,

| | A | B | C | D |
|---|---|---|---|---|
| **A** | 0 | 1 | 1 | 0 |
| **B** | 1 | 0 | 1 | 1 |
| **C** | 1 | 1 | 1 | 1 |
| **D** | 0 | 1 | 1 | 0 |

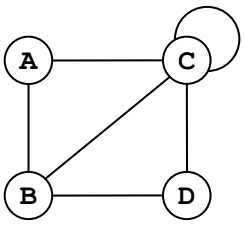
Единица на пересечении строки A и столбца B означает, что между узлами A и B есть связь. Ноль указывает на то, что связи нет. Такая таблица называется **матрицей смежности**. Она симметрична относительно главной диагонали (серые клетки в таблице).

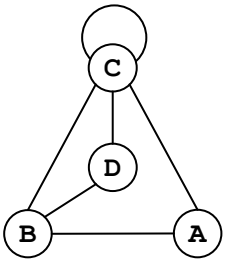
На пересечении строки C и столбца C стоит единица, которая говорит о том, что в графе есть **петля** – ребро, которое начинается и заканчивается в одной и той же вершине.

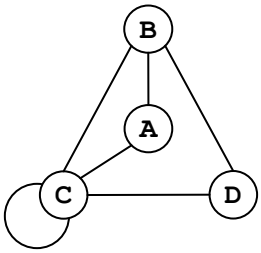
Можно поступить иначе: для каждого узла перечислить все узлы, с которыми связан данный узел. В этом случае мы получим **список смежности**. Для рассмотренного графа список смежности выглядит так:

(A (B, C), B (A, C, D), C (A, B, C, D), D (B, C))

Матрица смежности (и список смежности) не дают никакой информации о том, как именно расположены узлы друг относительно друга. Для таблицы, приведенной выше, возможны, например, такие варианты:



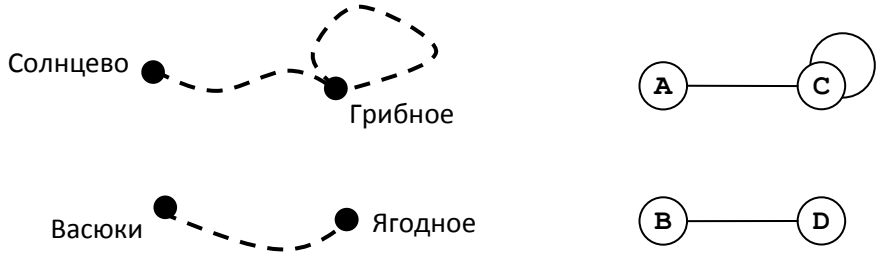



В рассмотренном примере все узлы связаны, то есть, между любой парой узлов существует **путь** – последовательность ребер, по которым можно перейти из одного узла в другой. Такой граф называется *связным*.

**Связный граф** – это граф, в котором все узлы связаны.

Теперь представьте себе, что дороги Васюки – Солнцево, Васюки – Грибное и Грибное – Ягодное завалило снегом (ли размыло дождем) так, что по ним не пройти и не проехать.




Эту схему тоже можно считать графом (она подходит под определение), но в таком графе есть две несвязанные части, каждая из которых – связный граф. Такие части называют **компонентами связности**.

Вспоминая материал предыдущего пункта, можно сделать вывод, что дерево – это частный случай связного графа. Но у него есть одно важное свойство – в дереве нет замкнутых путей (*циклов*). Граф в последнем примере не является деревом, потому что в нем есть циклы: ABCA, BCDB, ABCDA.

**Дерево** – это связный граф, в котором нет циклов.

Если в первом примере с дорогами нас интересуют еще и расстояния между поселками, каждой связи нужно сопоставить число (вес).


Такой граф называется **взвешенным**, поскольку каждое ребро имеет свой **вес**. В реальных задачах это может быть не только расстояние, но и, например, стоимость проезда или другая величина.

Как хранить информацию о таком графе? Ответ напрашивается сам собой – нужно в таблицу записывать не 1 или 0, а вес ребра. Если связи между двумя узлами нет, на бумаге можно оставить ячейку таблицы пустой, а при хранении в памяти компьютера записывать в нее условный код, например, –1. Такая таблица называется *весовой матрицей,* потому что содержит веса ребер. В данном случае она выглядит так:

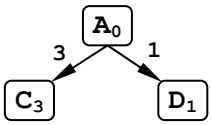
| | A | B | C | D |
|---|---|---|---|---|
| **A** | | 12 | 8 | |
| **B** | 12 | | 5 | 6 |
| **C** | 8 | 5 | 2 | 4 |
| **D** | | 6 | 4 | |

Также как и матрица смежности, весовая матрица симметрична относительно диагонали. Две пустые ячейки говорят о том, что между узлами A и D нет связи.

Если в графе немного узлов, весовая матрица позволяет легко определить наилучший маршрут из одного узла в другой простым перебором вариантов. Рассмотрим граф, заданный весовой матрицей (числа определяют стоимость поездки между соседними пунктами):

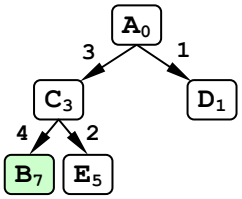
| | A | B | C | D | E |
|---|---|---|---|---|---|
| A | | | 3 | 1 | |
| B | | | 4 | 5 | 1 |
| C | 3 | 4 | | | 2 |
| D | 1 | 5 | | | 1 |
| E | | 1 | 2 | 1 | |

Найдем наилучший путь из A в B – такой, при котором общая стоимость поездки минимальная. Сначала видим, что из пункта A напрямую в B ехать нельзя, а можно ехать только в C и D. Изобразим это на схеме:

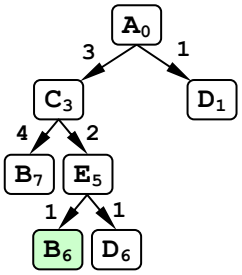



Числа около каждого ребер показывают стоимость поездки по этому участку, а индексы у названий узлов показывают общую стоимость проезда в данный узел из узла A.

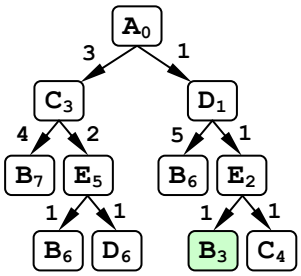
Теперь разберем варианты дальнейшего движения из узла C (узел A уже не нужно рассматривать, так как мы из него пришли).




Видим, что из C сразу можно попасть в B, стоимость проезда в этом случае равна 7. Но, возможно, это не самый лучший вариант, и нужно проверить еще путь через узел E. Действительно, оказывается, что можно сократить стоимость до 6:




Исследовать дальше маршрут, содержащий цепочку ACED, нет смысла, потому что его стоимость явно будет больше 6. Аналогично строим вторую часть схемы (вы догадались, что она представляет собой дерево!).

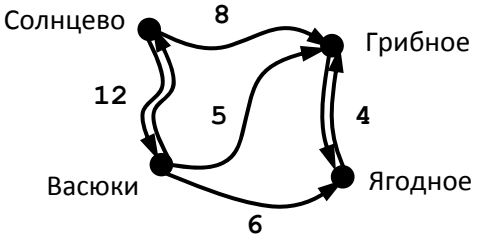


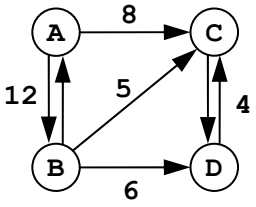

Таким образом, **оптимальный (наилучший) маршрут – ADEB**, его стоимость – 3. Маршруты ACED и ADEC, не дошедшие до узла B, далее проверять не нужно, они не улучшат результат.

Конечно, для более сложных графов метод перебора работает очень долго, поэтому используются более совершенные (но более сложные) методы, о которых мы поговорим в 11-м классе.

Наверное, вы заметили, что при изображении деревьев, которые описывают иерархию (подчинение), мы ставили стрелки от верхних уровней к нижним. Это означает, что для каждого ребра указывается направление, и двигаться можно только по стрелкам, но не наоборот. Такой граф называется *ориентированным* (или коротко *орграфом*). Он может служить, например, моделью системы дорог с односторонним движением. Матрица смежности и весовая матрица для орграфа уже не обязательно будут симметричными.

На этой схеме всего две дороги с двусторонним движением, по остальным можно ехать только в одну сторону:




| | A | B | C | D |
|---|---|---|---|---|
| A | | 12 | 8 | |
| B | 12 | | 5 | 6 |
| C | | | | 4 |
| D | | | 4 | |

Ребра в орграфе называют *дугами*. Дуга, в отличие от ребра, имеет начало и конец.

## *Контрольные вопросы*

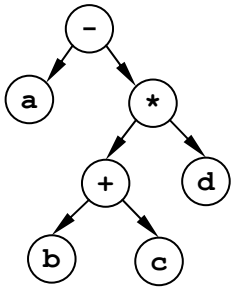
1. Что такое структура?
2. Что такое структурирование информации? Зачем оно нужно?
3. Что такое алфавитный порядок? Как поступают, если начальные символы слов совпали?
4. Расскажите, как используются оглавление, словарь и индекс для быстрого поиска нужной информации. Чем эти средства отличаются друг от друга?
5. Какими способами можно задать множество? Что такое пустое множество?
6. Приведите примеры множеств.
7. Чем отличаются множество и линейный список?
8. Что такое очередь? Приведите примеры.
9. Что такое стек? Чем он отличается от очереди? Приведите примеры.
10. Расшифруйте английские сокращения LIFO и FIFO.
11. Какая структура позволяет объединить возможности стека и очереди?
12. Что такое матрица?
13. Как можно записать табличные данные в виде списка?
14. Что такое иерархия? Приведите примеры.
15. Вспомните известные вам классификации, которые вы изучали на других предметах.
16. Как называется соответствующая структура в информатике?
17. Что такое «корень», «лист», «родитель», «сын», «предок», «потомок»?
18. В дереве 4 потомка и все они являются листьями. Нарисуйте это дерево. Сколько в нем узлов?
19. Что такое «лес»?
20. В чем разница между понятиями «ребро» и «дуга»?
21. В чем отличие понятий «дерево», «лес», «граф»?
22. Какой граф называется связным?
23. Что такое компонента связности?
24. Что такое петля? Как по матрице смежности определит, есть ли петли в графе?

25. Выберите наиболее подходящий способ структурирования информации для хранения
    а) данных по крупнейшим озерам мира;
    б) рецепта приготовления шашлыка;
    в) схемы железных дорог;
    г) схемы размещения файлов на флэш-диске.
26. Что такое орграф? Когда для представления данных используются орграфы? Приведите примеры.

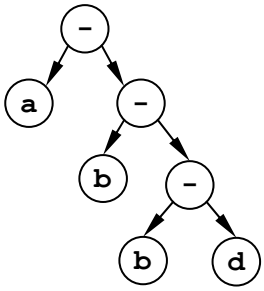
## *Задачи*

1. Определите выражения, соответствующие каждому из деревьев, в «нормальном» виде со скобками (эту форму называют *инфиксной* – операция записывается *между* данными). Постройте для каждого из них постфиксную форму.

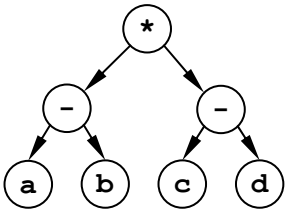
а)




б)




в)



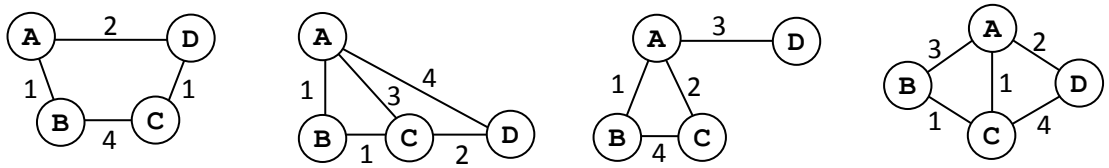

2. Постройте деревья, соответствующие следующим арифметическим выражениям. Запишите эти выражения в префиксной и постфиксной формах.
    а) **(a+b)*(c+2*d)**
    б) **(2*a-3*d)*c+2*b**
    в) **(a+b+2*c)*d**
    г) **3*a-(2*b+c)*d**
3. Вычислите выражение, записанное в постфиксной форме
    **а) 12 6 + 7 3 - 1 - * 12 +**
    **б) 12 10 – 5 7 + * 7 – 2 ***
    **в) 5 6 7 8 9 + - + -**
    **г) 5 4 3 2 1 - - - -**

    Запишите его в инфиксной и в префиксной формах. Постройте дерево, соответствующее этому выражению. Единственно ли такое дерево? В этом дереве назовите корень, листья и промежуточные узлы.

    (Ответ: 66; 34; 9; 3)
4. Нарисуйте граф, в котором 5 вершин и три компоненты связности. Постройте его матрицу смежности.
5. Структурируйте эту информацию разными способами: «Между поселками Верхние Васюки и Нижние Васюки есть проселочная дорога длиной 10 км. Село Сергеево соединяется двумя асфальтовыми шоссе с Нижними Васюками (22 км) и Верхними Васюками (16 км). В Солнечное можно доехать только из Сергеева по грунтовой дороге (5 км)». Можно ли сказать, как точно расположены эти пункты?
6. Для графа, полученного в предыдущей задаче, постройте матрицу смежности, список смежности, весовую матрицу. Является ли этот граф деревом?

7. Постройте матрицы смежности и весовые матрицы для каждого графа:

а) б) в) г)




8. Постройте графы, соответствующие каждой из матриц смежности:

а)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | 0 | 1 | 1 | 0 |
| **B** | 0 | | 1 | 0 | 1 |
| **C** | 1 | 1 | | 0 | 1 |
| **D** | 1 | 0 | 0 | | 0 |
| **E** | 0 | 1 | 1 | 0 | |

б)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | 0 | 1 | 1 | 1 |
| **B** | 0 | | 1 | 0 | 0 |
| **C** | 1 | 1 | | 0 | 1 |
| **D** | 1 | 0 | 0 | | 0 |
| **E** | 1 | 0 | 1 | 0 | |

в)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | 0 | 1 | 1 | 1 |
| **B** | 0 | | 1 | 0 | 1 |
| **C** | 1 | 1 | | 0 | 1 |
| **D** | 1 | 0 | 0 | | 0 |
| **E** | 1 | 1 | 1 | 0 | |

г)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | 0 | 0 | 1 | 0 |
| **B** | 0 | | 1 | 0 | 1 |
| **C** | 0 | 1 | | 1 | 1 |
| **D** | 1 | 0 | 1 | | 0 |
| **E** | 0 | 1 | 1 | 0 | |

9. Постройте графы, соответствующие каждой из весовых матриц:

а)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | 4 | 3 | | 7 |
| **B** | 4 | | | 2 | |
| **C** | 3 | | | 6 | |
| **D** | | 2 | 6 | | 1 |
| **E** | 7 | | | 1 | |

б)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | 2 | 5 | | 6 |
| **B** | 2 | | | 3 | |
| **C** | 5 | | | | |
| **D** | | 3 | | | 1 |
| **E** | 6 | | | 1 | |

в)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | | 2 | 2 | 6 |
| **B** | | | | 2 | |
| **C** | 2 | | | 2 | |
| **D** | 2 | 2 | 2 | | |
| **E** | 6 | | | | |

г)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | 5 | 2 | | 6 |
| **B** | 5 | | | 5 | |
| **C** | 2 | | | 2 | |
| **D** | | 5 | 2 | | 3 |
| **E** | 6 | | | 3 | |

10. Стоимость перевозок между пунктами, которые для краткости обозначены буквами A, B, C, D и E, задается таблицей (весовой матрицей графа). Нужно перевезти груз из пункта A в пункт B. Для каждого из четырех вариантов определите оптимальный маршрут и полную стоимость перевозки.

а)

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

б)

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

в)

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

г)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | | | 1 | |
| **B** | | | 4 | | 1 |
| **C** | | 4 | | 4 | 2 |
| **D** | 1 | | 4 | | |
| **E** | | 1 | 2 | | |

11. Постройте орграф, соответствующий каждой из этих таблиц.

а)

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

б)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| **A** | | | 5 | 1 | 1 |
| **B** | | | 6 | 4 | |
| **C** | 3 | 4 | | | 2 |
| **D** | | 2 | | | |
| **E** | | | 3 | | |

в)

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

г)

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

# Глава II. Кодирование информации

## 2.1 Язык и алфавит

### *2.1.1 Как записать информацию?*

Для того, чтобы хранить, передавать и обрабатывать информацию, ее необходимо как-то зафиксировать с помощью символов (знаков), то есть записать на каком-то *языке*.

**Язык** – это система знаков, используемая для хранения, передачи и обработки информации.

Естественные языки (русский, английский и др.) сформировались в результате развития человеческого общества и используются для общения людей.

Сначала древние люди овладели устной речью. Поскольку человек может издавать и различать на слух не так много звуков, он стал комбинировать их, составляя слова, каждому из которых приписывался некоторый смысл.

Затем появилась необходимость записывать информацию, например, для передачи потомкам. Сначала жизненный опыт пытались зафиксировать в виде рисунков животных и предметов, затем пиктограмм (схематических изображений предметов и явлений), иероглифов (см. рисунок).

| Египетское письмо | |
|---|---|
|  | рука |
|  | дом |
|  | кобра |
|  | лев |
|  | вода |
|  | рот |
|  | мужчина |
|  | женщина |

| Иероглифы (Китай) | |
|---|---|
| 日 | солнце |
| 月 | луна |
| 雨 | дождь |
| 山 | гора |
| 马 | лошадь |
| 鱼 | рыба |
| 人 | человек |
| 女 | женщина |

Сейчас мы используем *алфавитное письмо*, где каждый знак (или сочетание знаков) обозначает некоторый звук, так что с помощью небольшого набора знаков (*алфавита*) можно записать любые слова устной речи.

**Алфавит** – это набор знаков, который используется в языке.

Чаще всего подразумевается, что символы в алфавите расположены в определенном порядке.

Обычно алфавит естественных языков содержит около 30 знаков. Например, в русском языке 33 буквы, в английском – 26. К алфавиту языка, вообще говоря, нужно отнести пробел (пропуск между словами), цифры (знаки для записи чисел), знаки препинания, скобки.

**Мощность алфавита** – это количество знаков в алфавите.

Например, алфавит, состоящий из 33 русских букв, 10 цифр, пробела и 12 знаков препинания (точка, запятая, вопросительный и восклицательный знаки, тире, двоеточие, многоточие, кавычки, круглые скобки) имеет мощность 56.

**Слово** – это последовательность символов алфавита, которая используется как самостоятельная единица и имеет определенное значение.

Из слов составляются *предложения*, каждое из которых выражает определенную законченную мысль (сообщение, порцию информации). В любом языке определяются правила построения слов (*грамматика*), правила построения предложений (*синтаксис*) и правила расстановки знаков препинания (*пунктуация*).

С точки зрения теории информации, сообщение – это любой набор знаков некоторого алфавита. Определим, сколько различных сообщений можно построить с помощью заданного количества знаков. Например, используя только знаки из набора @#$%, можно записать 4 сообщения из одного символа: @, #, $ и %. Теперь рассмотрим сообщения из двух знаков. Первый знак можно выбрать четырьмя способами, и для каждого из них есть 4 варианта выбора второго знака. Поэтому сообщений, состоящих из двух знаков, будет $4^2 = 16$:

| | | | |
|---|---|---|---|
| @@ | #@ | $@ | %@ |
| @# | ## | $# | %# |
| @$ | #$ | $$ | %$ |
| @% | #% | $% | %% |

Рассуждая аналогично, получим, что трехсимвольных сообщений будет $4^3 = 64$, а четырехсимвольных – $4^4 = 256$ и т.д.

Если алфавит языка состоит из $M$ символов (имеет мощность $M$), количество различных сообщений длиной $N$ знаков вычисляется как

$$Q = M^N$$

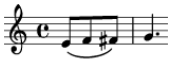
## 2.1.2 Естественные и формальные языки

Вы знаете, что в естественных языках кроме правил есть и исключения. Кроме того, одно слово может иметь различный смысл в зависимости от *контекста*, то есть от предложения, в котором оно употребляется. Существует некоторая «свобода понимания»: слова и выражения могут пониматься немного по-разному в зависимости от множества условий: опыта человека, его культуры, уровня образованности, настроения и т.п. Так многие кулинарные рецепты содержат совет «добавить соль по вкусу», который каждый выполняет по-своему.

Часто, например, в научных публикациях, такая ситуация недопустима, потому что смысл текста должен быть понят однозначно. В таких случаях используют языки специального типа, в которых каждое слово и словосочетание имеет четко определенное единственное значение, и нет никаких исключений.

**Формальный язык** – это язык, в котором однозначно определяется значение каждого слова, а также правила построения предложений и придания им смысла.

Вот некоторые примеры формальных языков:

- математические формулы: $y = f(x)$;
- химические формулы и правила записи реакций: $2H_2 + O_2 = 2H_2O$;
- системы счисления (правила записи чисел с помощью специальных знаков – цифр): 12345, XXI;
- нотная запись: 
- язык записи шахматных партий: 1. **e2-e4 e7-e5**…
- язык программирования Паскаль:

```
program qq;
begin
  write('Привет, Вася!');
end.
```

Все формальные языки – *искусственные*. В отличие от естественных языков, которые формировались в течение многих веков и неотделимы от истории каждого народа и его культуры,

формальные языки разрабатываются людьми для общения в специальных областях знаний. Например, нотная запись позволяет сохранить и передать музыкальное произведение.

В следующей таблице проведено сравнение естественных и формальных языков:

| Естественные языки | Формальные языки |
|---|---|
| • сформировались в результате развития общества; | • созданы искусственно; |
| • используются для общения в быту; | • используются для общения в специальных областях знаний; |
| • часто встречаются слова с неточным и неясным содержанием; | • не допускаются слова с неточным и неясным содержанием; |
| • значения отдельных слов и предложений зависят не только от них самих, но и от их окружения (контекста); | • значения отдельных слов и предложений не зависит от контекста |
| • встречаются *синонимы* (разные слова имеют одинаковый смысл) | • как правило, синонимов нет |
| • встречаются *омонимы* (одно слово может иметь несколько значений); | • омонимов нет |
| • нет строгих правил образования предложений | • правила образования предложений строго определены |
| • для многих правил существуют исключения | • нет исключений из правил |

### *Контрольные вопросы*

1. Что такое язык?
2. Зачем нужны языки?
3. Какие языки называются естественными?
4. Что такое алфавит языка?
5. Как вы думаете, почему алфавит большинства современных языков содержит небольшое число знаков?
6. Чем отличается алфавитное письмо от использования иероглифов?
7. Какие правила существуют в языке? Как они называются?
8. В каких областях требуется использование формальных языков?
9. Чем отличается формальный язык от естественного?
10. Что такое *контекст*? Почему меню, которое появляется при нажатии правой кнопки мыши над объектом, называют *контекстным*?
11. Приведите примеры формальных языков, о которых не упоминалось в тексте учебника.
12. Объясните, почему любой язык программирования – это формальный язык?
13. Как вы думаете, почему люди не отказываются от естественных языков и не переходят на формальные во всех областях?
14. Как вы думаете, почему любой формальный язык не является универсальным и хорошо подходит для записи информации только в определенной области?
15. Выберите из таблицы те свойства естественных языков, которые затрудняют и не позволяют полностью автоматизировать перевод с одного языка на другой. Приведите примеры.

# 2.2 Дискретность

## *2.2.1 Аналоговые и дискретные сигналы*

Одна из важнейших задач любой информационной системы – обеспечить надежную передачу информации в условиях, когда на сигнал действуют помехи. Элементы электронных устройств, как правило, обмениваются данными с помощью электрических сигналов; для получения информации приемник должен измерить этот сигнал (чаще всего – напряжение на

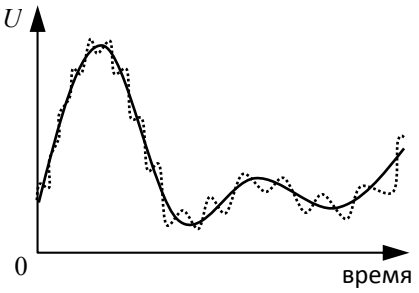
контактах). Изменение электрического сигнала может произойти в любой момент и быть любым (в пределах допустимого диапазона). Такие сигналы называют *аналоговыми*.

**Аналоговый сигнал** – это сигнал, который в любой момент времени может принимать любые значения в заданном диапазоне.

Органы чувств человека воспринимают информацию в аналоговой форме: свет, звуковые волны, вкус, запах и т.п. Поэтому раньше большинство технических устройств для работы с информацией (радио, телефоны, магнитофоны, фотоаппараты) тоже были аналоговыми.

В 60-х годах XX века были широко распространены *аналоговые компьютеры*, которые выполняли вычисления с аналоговыми сигналами (сложение, вычитание, умножение, извлечение квадратного корня). Однако они решали достаточно узкий круг задач (моделирование законов движения), и их точность была невысока.

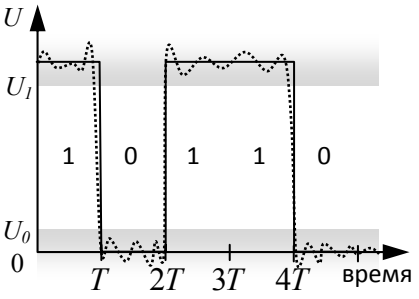
Дело в том, что при передаче сигнала всегда есть помехи, которые искажают его форму. В большинстве случаев эти искажения – случайные ошибки, не поддающиеся учету. Фактически приемник получает не исходный сигнал, посланный источником (сплошная линия на рисунке), а искаженный (точечная линия).



Аналоговые сигналы

Вспомним, что аналоговый сигнал может принимать любые значения в некотором диапазоне. «Очистить» его от помех в общем случае нельзя, потому что невозможно понять, искажен он или в самом деле имеет такое значение. Кроме того, дополнительные ошибки (погрешности) вносятся при измерении сигнала.

Если использовать аналоговые компьютеры, мы будем при каждом расчете с одинаковыми исходными данными получать несколько отличающиеся результаты. Кроме того, при копировании аналоговая информация искажается (например, при каждом копировании звукозаписи на магнитной ленте качество копии ухудшается).




Эта ситуация не устраивала ученых и инженеров, и они нашли интересное решение: если не удается точно измерить сигнал, нужно вообще отказаться от его измерения, а просто через некоторый интервал времени $T$ определять, есть сигнал в данный момент или нет (эти состояние можно обозначить как 1 и 0)[1]. При использовании такого подхода мы получаем огромное преимущество: при небольших помехах искажение формы сигнала не влияет на передачу данных – если напряжение выше некоторого порога $U_1$, считается, что сигнал равен 1, а все сигналы, меньшие другого порога $U_0$, считаются равными нулю (см. рисунок).

Сигналы, с которыми работает компьютер, называются *дискретными* или *цифровыми*. Они обладают двумя важными свойствами:

- изменяются только в отдельные моменты времени (*дискретность по времени*);
- принимают только несколько возможных значений (*дискретность по уровню*).

**Дискретный (цифровой) сигнал** – это последовательность значений, каждое из которых принадлежит некоторому множеству.

[1] Может быть и наоборот: 0 обозначает, что сигнал есть, а 1 – сигнала нет.

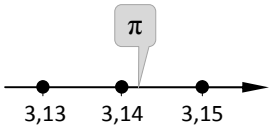
Обратите внимание на важный момент – мы естественно пришли к необходимости использования дискретных сигналов, когда стало необходимо точно и однозначно воспринимать передаваемую информацию с учетом неизбежных помех. Этот принцип применим не только к компьютерам. Переход от наскальных рисунков к алфавитному письму, где каждый знак имеет четко определенное значение, – это тоже переход от аналоговых сигналов к дискретным, цель которого – максимально исключить неоднозначное понимание смысла.

## *2.2.2 Дискретизация*

Дискретность означает, что мы представляем целое (непрерывное) в виде набора отдельных элементов. Например, картина художника – это аналоговая (непрерывная) информация, а мозаика, сделанная на ее основе (рисунок из кусочков разноцветного стекла) – дискретная.

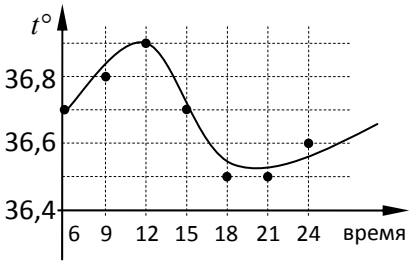
**Дискретизация** – это представление единого объекта в виде множества отдельных элементов.

Всем известное иррациональное число $\pi$ содержит бесконечное количество знаков в дробной части. Если мы хотим записать, чему равно $\pi$, необходимо остановиться на каком-то знаке, отбросив остальные, например, $\pi \approx 3{,}14$. Таким образом, мы перешли к дискретной информации, потому что рассматриваем только числа с шагом 0,01 – точки на числовой оси.




Изменение высоты столбика термометра – это аналоговая информация, а *записанная* температура 36,6° – дискретная, потому что ее округляют до десятых долей градуса.

аналоговая информация



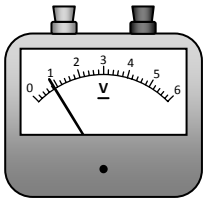
дискретизация

6 ч. 36,7°
9 ч. 36,8°
12 ч. 36,9°
15 ч. 36,7°
18 ч. 36,5°
21 ч. 36,5°
24 ч. 36,6°

дискретная информация

Дискретность состоит в том, что записанные значения температуры изменяются скачкообразно (через 0,1°) – это дискретизация по уровню. Кроме того, обычно температуру больного измеряют не непрерывно, а несколько раз в день – появляется дискретизация по времени.

Заметим, что при дискретизации, как правило, происходит *потеря информации*. В данном случае мы, во-первых, потеряли информацию об изменении температуры между моментами измерений и, во-вторых, исказили измеренные значения, округлив их до десятых (каждая дискретизация, и по времени, и по уровню, вносит свою ошибку). Чтобы уменьшить ошибки, нужно уменьшать шаг дискретизации – измерять температуру чаще, записывать показания термометра до тысячных долей градуса. Однако, в любой практической задаче есть некоторый предел, после которого увеличение точности уже никак не влияет на конечный результат.



аналоговая информация

дискретная информация

Из приведенного примера понятно, что непрерывность и дискретность – это не свойство самой информации, а свойство ее *представления*. В данном случае информация – это сведения об изменении температуры человека в течение дня. Если бы она измерялась постоянно и записывалась самописцем (в виде графика), можно было бы говорить о

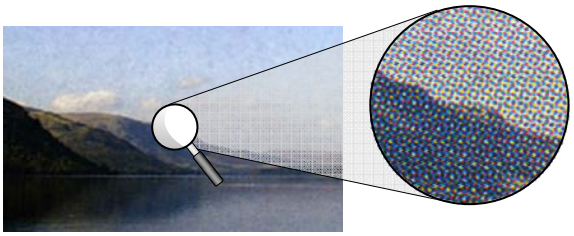
том, что эта информация представлена в аналоговой (непрерывной) форме.

Еще один пример – аналоговые («стрелочные») и цифровые вольтметры, которые измеряют одну и ту же величину, но выводят результат измерения в разном виде.

Теперь подумаем, как записать аналоговую величину, которая может принимать бесконечное множество значений. Вы уже знаете, что с помощью $N$ знаков алфавита, состоящего из $M$ символов, можно закодировать $Q = M^N$ разных сообщений. Поэтому теоретически для записи аналоговой величины придется использовать бесконечное число знаков.

Таким образом, когда мы хотим записать (зафиксировать) информацию с помощью какого-то алфавита, нужно переходить к дискретному представлению. С одной стороны, это делает более надежной передачу данных (если обе стороны одинаково понимают используемые знаки). С другой стороны, при дискретизации часть информации теряется. Например, «живой» рассказ человека (со всеми интонациями, паузами и т.п.) обычно оказывает большее впечатление, чем чтение этого же текста, записанного на бумаге.

Хотя аналоговую информацию невозможно точно представить в дискретном виде, при увеличении точности дискретизации свойства непрерывной и дискретной информации практически совпадают. Например, для точной записи числа π требуется бесконечное количество цифр, но в расчетах чаще всего достаточно знать это значение с точностью не более 10 знаков.



Идеальная непрерывность существует только в теории. Мы считаем дерево, пластмассу, металл непрерывным, но на самом деле они состоят из отдельных молекул, расположенных на некотором расстоянии друг от друга – это значит, что вещество дискретно. Иллюстрация в книге кажется нам сплошной, но при сильном увеличении видно, что она строится из отдельных точек (имеет «растр»). Классическая («пленочная») фотография считается аналоговой, но при увеличении снимка с фотопленки нельзя бесконечно получать все новые и новые детали – предел «уточнения» определяется величиной зерна светочувствительного материала.

Мы часто воспринимаем дискретные объекты как непрерывные, потому что наши органы чувств не позволяют различить отдельные элементы. Например, *разрешающая способность* глаза составляет около одной угловой минуты (1' = 1/60 часть градуса), это значение определяется размером элементов сетчатки. Поэтому человек не может различить два объекта, если направления на них отличаются меньше, чем на 1'. Для того, чтобы повысить разрешающую способность при наблюдении, применяют бинокли.

## ? *Контрольные вопросы*

1. Что такое аналоговый сигнал?
2. Какие бытовые устройства работают с аналоговыми сигналами?
3. Какие системы связи используют аналоговые сигналы, а какие – дискретные?
4. Что такое аналоговые компьютеры? Почему они вышли из употребления?
5. Почему при использовании аналоговой техники передача информации всегда происходит с искажениями?
6. *Как вы думаете, почему аналоговый сигнал нельзя полностью «очистить» от помех? Какую дополнительную информации нужно иметь, чтобы эта задача могла быть частично решена?
7. Что такое дискретный сигнал?
8. Почему с помощью дискретного сигнала можно передавать информацию практически без искажений? Искажается ли форма такого сигнала под воздействием помех?

9. Сигнал изменяется в моменты времени, кратные 1 секунде, и может принимать одно из 16 возможных значений. Является ли такой сигнал дискретным? (Ответ: да)
10. Что такое дискретизация? Приведите примеры.
11. Что такое дискретизация по времени и дискретизация по уровню?
12. Приведите пример дискретизации сигнала только по уровню, только по времени. Нарисуйте графики таких процессов.
13. Почему при дискретизации, как правило, происходит потеря информации?
14. Как можно уменьшить потери информации при дискретизации? Почему не стоит стремиться максимально уменьшить эти потери?
15. Приведите примеры, когда одна и та же информация может быть представлена в аналоговой и дискретной форме.
16. *Выясните, какие музыкальные инструменты позволяют извлекать только дискретные звуки (заранее определенные ноты), а какие – звук любой частоты.
17. Объясните фразу «при увеличении точности дискретизации свойства непрерывной и дискретной информации практически совпадают».

## 2.3 Кодирование

Для передачи и обработки информации ее всегда *кодируют*, то есть записывают в другой знаковой системе (на другом языке).

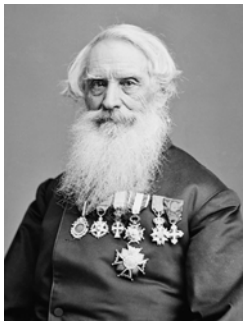
**Кодирование** – это преобразование информации в форму, удобную для ее хранения, передачи и обработки. Правило такого преобразования называется **кодом**.

В зависимости от конкретной задачи информация может кодироваться разными способами. Например, предложение «Привет, Вася!» может быть закодировано *транслитом* (так сокращенно называют транслитерацию – русский текст, записанный латинскими буквами): «Privet, Vasya!». Такой метод используют в электронных письмах, когда у одного собеседника (или у обоих) на компьютере нет поддержки русского языка. То же самое сообщение можно просто перевести на английский (или другой) язык, например, если собеседник не знает русского. А можно даже зашифровать: «Рсйгжу-!Гбта”». Это один из способов кодирования, при котором нужно скрыть смысл сообщения от посторонних[2].

Для кодирования чисел в разных ситуациях тоже используют разные способы. Например, число 21 можно записать как XXI (в римской системе счисления) или «Двадцать один» (в финансовых документах).

### *2.3.1 Код Морзе*

Долгое время для передачи сообщений по телеграфу и радио использовался код Морзе[3] (или азбука Морзе), предложенный американским художником и изобретателем Самюэлем Морзе. В этом коде все буквы и цифры кодируются в виде различных последовательностей точек и тире.



Самюэль Морзе
(1791-1872)

[2] Это сообщение зашифровано с помощью шифра Цезаря. Попробуйте разгадать этот шифр и сформулировать правила кодирования и декодирования.

[3] Код Морзе применялся в британском флоте с 1865 г. для передачи сообщений с помощью флажков (днем) и фонарей (ночью). Для этой же цели использовали прожектора, у которых закрывали и открывали специальные жалюзи, а также сирены (для звуковой связи). С начала XX века код Морзе начали применять в радиосвязи.

**Код Морзе для русских букв и цифр**

| | | | | | |
|---|---|---|---|---|---|
| А | •– | О | ––– | Э | ••–•• |
| Б | –••• | П | •––• | Ю | ••–– |
| В | •–– | Р | •–• | Я | •–•– |
| Г | ––• | С | ••• | | |
| Д | –•• | Т | – | 1 | •———— |
| Е | • | У | ••– | 2 | ••––– |
| Ж | •••– | Ф | ••–• | 3 | •••–– |
| З | ––•• | Х | •••• | 4 | ••••– |
| И | •• | Ц | –•–• | 5 | ••••• |
| Й | •––– | Ч | –––• | 6 | –•••• |
| К | –•– | Ш | ———— | 7 | ––••• |
| Л | •–•• | Щ | ––•– | 8 | –––•• |
| М | –– | Ь | –••– | 9 | ————• |
| Н | –• | Ы | –•–– | 0 | ————— |

Легко понять, что это дискретный код, и здесь дискретность введена для того, чтобы надежно принимать информацию при наличии помех.

Код Морзе – *неравномерный*, то есть коды символов могут быть разной длины. Для сокращения общего времени передачи буквы, которые встречаются чаще, должны иметь более короткие коды. Частоты встречаемости букв английского алфавита Морзе оценил по количеству литер в типографской машине. Поэтому английская буква «E», которая встречается в текстах чаще всего, получила код •. Коды Морзе для русских букв совпадают с кодами похожих по звучанию английских букв, например коды букв «Л» и «L» одинаковы[4].

Чтобы отделить последовательности (коды букв) друг от друга, вводят еще один символ – пробел (пауза). Например, имя «Вася», закодированное с помощью кода Морзе, выглядит так:

•––  •–  •••  •–•–

## *2.3.2 Двоичное кодирование*

Неизменный во времени сигнал не может передавать информацию, поэтому обязательно нужно, чтобы свойства носителя как-то изменялись. Самый простой используемый код должен содержать, по крайней мере, два разных знака.

Мы видели, что сигналы, которые используются в компьютерах для обмена данными – это дискретные сигналы, принимающие два возможных значения, которые условно обозначают как 0 и 1. Поэтому вся хранимая, передаваемая и обрабатываемая информация (числа, текст, рисунки, звуки, видео) должна быть закодирована в виде последовательностей нулей и единиц. Такое кодирование называют *двоичным* (от слова «два»).

**Двоичное кодирование** – это кодирование с помощью двух знаков.

Например, сообщение АБАВГБ может быть закодировано с помощью кодовой таблицы

| А | Б | В | Г |
|---|---|---|---|
| 00 | 01 | 10 | 11 |

следующим образом: 000100101101.

Кодирование чисел с помощью нулей и единиц впервые применил в своей (механической) вычислительной машине немецкий мыслитель Готфрид Вильгельм Лейбниц в конце XVII века.

[4] Поэтому код Морзе для русских букв менее эффективен.

Затем, уже в середине XX века, двоичное кодирование информации стало повсеместно применяться для электронных компьютеров.

Чаще всего используется равномерный код, когда все символы исходного сообщения кодируются с помощью одинакового количества двоичных знаков (или, что то же самое, битов, потому что каждый знак фактически соответствует выбору между 0 и 1, то есть несет 1 бит информации). Длина кода определяется количеством вариантов, которые нужно закодировать. Поскольку алфавит двоичного кода содержит 2 символа, применяя общую формулу, получаем количество различных сообщений длиной $N$ бит: $Q = 2^N$.

Если заданное количество вариантов не равно степени числа 2, нужно выбирать длину кода с запасом. Например, для кодирования номера спортсмена в интервале от 1 до 200 нужно использовать не меньше, чем 8 бит, поскольку

$$2^7 = 128 < 200 \le 2^8 = 256 .$$

Если нужно передать информацию о номерах первых 20 спортсменов, пришедших к финишу, информационный объем такого сообщения будет равен 8·20 = 160 бит = 20 байт.

## ? *Контрольные вопросы*

1. Что такое кодирование?
2. Зачем кодируют информацию?
3. Что такое код?
4. Какой алфавит используется в коде Морзе?
5. Какие буквы в коде Морзе имеют самые короткие коды? Почему?
6. Предложите, как можно изменить азбуку Морзе, чтобы сообщения на русском языке стали более короткими.
7. Запишите свое имя с помощью кода Морзе.
8. Почему в коде Морзе необходим символ-разделитель (пауза)?
9. В каком случае используется транслитерация?
10. Где сейчас используются числа, записанные в римской системе?
11. Как вы думаете, зачем в финансовых документах денежные суммы пишут прописью?
12. Какое кодирование называют двоичным?
13. Можно ли при двоичном кодировании использовать не 0 и 1, а другие знаки (например, буквы А и Б)?

## *Задачи*

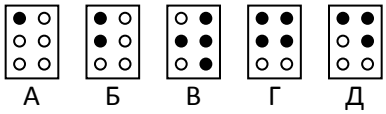
1. Сколько существует в коде Морзе различных последовательностей из точек и тире, длина которых равна 4 символа? 6 символов?
2. Сколько различных пятизначных чисел можно записать с помощью цифр 4 и 2? (Ответ: 32)
3. В алфавите языка племени «тамба-амба» две буквы: Й и Ы. Сколько различных 11-буквенных слов можно образовать в этом языке? (Ответ: 2048)
4. Алфавит языка «амба-карамба» состоит из 5 букв. Сколько различных четырехбуквенных слов можно образовать в этом языке? (Ответ: 625)
5. В языке племени «тумба-юмба» разрешены только четырехбуквенные слова, которые можно образовывать из букв алфавита в любых комбинациях. Известно, что словарный запас языка составляет 81 слово. Какова мощность алфавита? (Ответ: 3)
6. Алфавит некоторого языка содержит только трехбуквенные слова, которые можно образовывать из букв алфавита в любых комбинациях. Известно, что словарный запас языка составляет 216 слов. Какова мощность алфавита? (Ответ: 6)
7. Какое наименьшее число символов должно быть в алфавите, чтобы при помощи всевозможных трехбуквенных слов, состоящих из символов данного алфавита, можно было передать не менее 9 различных сообщений? (Ответ: 3)

8. Световое табло состоит из лампочек. Каждая лампочка может находиться в одном из трех состояний («включено», «выключено» или «мигает»). Какое наименьшее количество лампочек должно находиться на табло, чтобы с его помощью можно было передать 18 различных сигналов? (Ответ: 3)
9. Некоторое сигнальное устройство за одну секунду передает один из трех сигналов. Сколько различных сообщений длиной в четыре секунды можно передать при помощи этого устройства? (Ответ: 81)
10. Световое табло состоит из светящихся элементов, каждый из которых может гореть одним из двух различных цветов (или не гореть вообще). Сколько различных сигналов можно передать с помощью табло, состоящего из пяти таких элементов? (Ответ: 243)
11. Для передачи сигналов на флоте используются специальные сигнальные флаги, вывешиваемые в одну линию (последовательность важна). Какое количество различных сигналов может передать корабль при помощи пяти сигнальных флагов, если на корабле имеются флаги четырех различных видов (флагов каждого вида неограниченное количество)? (Ответ: 1024)
12. Вася и Петя передают друг другу сообщения, используя синий, красный и зеленый фонарики. Это они делают, включая по одному фонарику на одинаковое короткое время в некоторой последовательности. Количество вспышек в одном сообщении – 3 или 4, между сообщениями – паузы. Сколько различных сообщений могут передавать мальчики? (Ответ: 108)
13. Для кодирования 300 различных сообщений используются 5 последовательных цветовых вспышек. Вспышки одинаковой длительности, для каждой вспышки используется одна лампочка определенного цвета. Лампочки скольких цветов должны использоваться при передаче (укажите минимально возможное количество)? (Ответ: 4)
14. *Некоторый алфавит содержит 4 различных символа. Сколько трехбуквенных слов можно составить из символов этого алфавита, если символы в слове не могут повторяться? (Ответ: 24)
15. *В текстовом процессоре есть 5 кнопок, с помощью которых можно включать и выключать следующие режимы: жирный шрифт, курсив, подчеркивание, верхний индекс, нижний индекс. Сколько различных стилей оформления текста можно использовать? (Ответ: 24)
16. Используя кодовую таблицу

| А | Б | В | Г |
|---|---|---|---|
| 0 | 01 | 100 | 11 |

закодируйте сообщение ГАВВАБ.

17. Шрифт Брайля[5] – это специальный шрифт, с помощью которого незрячие люди могут читать. Для кодирования используются 6 точек, расположенных в два столбца. В каждой из них может быть выпуклость, которую человек воспринимает на ощупь. Вот коды Брайля первых букв русского алфавита (черная точка обозначает выпуклость):




Сколько различных символов можно закодировать с помощью кода Брайля. (Ответ: 63)

18. Предложите какой-нибудь способ перехода от шрифта Брайля к двоичному кодированию.
19. В чем преимущества использования двоичного кодирования информации в современных компьютерах?
20. Сколько существует различных последовательностей из символов «плюс» и «минус» длиной ровно в пять символов? (Ответ: 32)
21. На хранение целого числа отвели 12 бит. Сколько различных чисел можно закодировать таким образом? (Ответ: 4096)

[5] http://ru.wikipedia.org/wiki/Шрифт_Брайля

22. Разведчик кодирует секретные сообщения, расставляя крестики и нолики в ячейки таблицы. Всего он может закодировать 512 сообщений. Сколько ячеек в таблице у разведчика? (Ответ: 9)
23. Шахматная доска состоит 8 столбцов и 8 строк. Какое минимальное количество бит потребуется для кодирования координат одного шахматного поля? (Ответ: 6)
24. Какое минимальное количество бит потребуется для кодирования положительных чисел, меньших 60? (Ответ: 6)
25. Для кодирования значений температуры воздуха (целое число в интервале от –50 до 40) используется двоичный код. Какова минимальная длина двоичного кода? (Ответ: 7 бит)
26. Метеорологическая станция ведет наблюдение за влажностью воздуха. Результатом одного измерения является целое число от 0 до 100 процентов, которое записывается при помощи минимально возможного количества бит. Станция сделала 80 измерений. Определите информационный объем результатов наблюдений. (Ответ: 70 байт)
27. Обычный дорожный светофор без дополнительных секций подает шесть видов сигналов (непрерывные красный, желтый и зеленый, мигающие желтый и зеленый, красный и желтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 100 сигналов светофора. Определите информационный объем этого сообщения. (Ответ: 37,5 байт)
28. В некоторой стране автомобильный номер длиной 6 символов составляется из заглавных букв (всего используется 12 букв) и десятичных цифр в любом порядке. Каждый символ кодируется одинаковым и минимально возможным количеством бит, а каждый номер – одинаковым и минимально возможным количеством байт. Определите объем памяти, необходимый для хранения 32 автомобильных номеров. (Ответ: 128 байт)
29. В базе данных хранятся записи, содержащие информацию о датах. Каждая запись содержит три поля: год (число от 1 до 2100), номер месяца (число от 1 до 12) и номер дня в месяце (число от 1 до 31). Каждое поле записывается отдельно от других полей с помощью минимально возможного числа бит. Определите минимальное количество бит, необходимое для кодирования одной записи. (Ответ: 21)
30. В велокроссе участвуют 678 спортсменов. Специальное устройство регистрирует прохождение каждым из участников промежуточного финиша, записывая его номер с использованием минимально возможного количества бит, одинакового для каждого спортсмена. Каков информационный объем сообщения (в байтах), записанного устройством, после того как промежуточный финиш прошли 200 велосипедистов? (Ответ: 250 байт)

## 2.4 Декодирование

**Декодирование** – это восстановление сообщения из последовательности кодов.

Например, закодированное сообщение

•−−•  ••−  •−−•  −•−  ••  −•

можно восстановить, используя код Морзе «в обратную сторону»: в этой строке закодирована фамилия «Пупкин».

В некоторых случаях даже при использовании неравномерного кода не требуется вводить символ-разделитель. Для этого достаточно выполнение условия Фано: ни один код не является началом другого кода. Такой код называют **префиксным**. Например, для кодирования первых 5 букв русского алфавита используется таблица

| А | Б | В | Г | Д |
|---|---|---|---|---|
| 000 | 10 | 01 | 110 | 001 |

Это неравномерный код, поскольку в нем есть двух- и трехсимвольные коды. Однако условие Фано выполняется, поэтому любую правильную кодовую последовательность можно однозначно

декодировать. Например, рассмотрим цепочку 1100000100110. Букв с кодами 1 и 11 в таблице нет, поэтому сообщение начинается с буквы Г – она имеет код 110:

| Г | |
|---|---|
| 110 | 0000100110 |

Следующий (единственно возможный) код – 000, это буква А:

| Г | А | |
|---|---|---|
| 110 | 000 | 0100110 |

Аналогично декодируем все сообщение:

| Г | А | В | Д | Б |
|---|---|---|---|---|
| 110 | 000 | 01 | 001 | 10 |

Рассмотрим другую кодовую таблицу:

| А | Б | В | Г | Д |
|---|---|---|---|---|
| 000 | 01 | 10 | 011 | 100 |

Здесь условие Фано не выполняется, поскольку код буквы Б (01) является началом кода буквы Г (011), а код буквы Д (100) начинается с кода буквы В (10). Тем не менее, можно заметить, что выполнено «обратное» условие Фано: ни один код не является окончанием другого кода (такой код называют **постфиксным**). Поэтому закодированное сообщение можно однозначно декодировать с конца. Например, рассмотрим цепочку 011000110110. Последней буквой в этом сообщении может быть только В (код 10):

| | В |
|---|---|
| 0110001101 | 10 |

Вторая буква с конца – Б (код 01):

| | Б | В |
|---|---|---|
| 01100011 | 01 | 10 |

и так далее:

| Б | Д | Г | Б | В |
|---|---|---|---|---|
| 01 | 100 | 011 | 01 | 10 |

В общем случае декодировать сообщение удается только перебором вариантов. Например, декодируем сообщение 010100111101, закодированное с помощью кодовой таблицы

| А | Б | В | Г | Д |
|---|---|---|---|---|
| 01 | 010 | 011 | 11 | 101 |

Здесь не выполняется ни «прямое», ни «обратное» условие Фано, поэтому декодировать сразу, скорее всего, не удастся. На первом месте может быть, буква А или буква Б. Сначала предположим, что это буква А:

А0100111101

Тогда второй буквой также может быть буква А:

АА00111101.

Дальше декодировать не получается, потому что в таблице нет кодов 0, 00 и 001. Поэтому проверяем второй вариант: вторая буква – Б:

АБ0111101.

Третьей буквой может быть А:

АБА11101,

Тогда четвертая и пятая буквы определяются однозначно – это буквы Г и Д. Таким образом, один из подходящих вариантов – АБАГД.

Посмотрим, есть ли другие варианты. После сочетания АБ может стоять буква В:

АБВ1101,

тогда оставшиеся буквы – это ГА, а полное сообщение – АБВГА. Этот вариант тоже подходит.

Кроме того, на первом месте может стоять буква Б:

Б100111101,

но дальше декодировать не удается, потому что в таблице нет кодов 1, 10 и 100. Таким образом, сообщение может быть декодировано двумя способами: АБАГД и АБВГА.

Иногда при кодировании и декодировании происходит искажение сообщения. Например, известно, что перевод художественных текстов (особенно стихов) на другой язык и обратный перевод может изменить их до неузнаваемости[6].

## *Контрольные вопросы*

1. Что такое декодирование?
2. Всегда ли удается однозначно декодировать сообщение? В каких случаях это может быть не так?
3. Перечислите достаточные условия, при которых можно однозначно декодировать сообщение с неравномерным кодом.
4. В каких случаях для декодирования приходится использовать перебор вариантов?

## *Задачи*

1. Расшифруйте сообщение, записанное с помощью кода Морзе, которое используется как международный сигнал бедствия: ••• ——— •••.
2. Для кодирования сообщения используется таблица

| А | Б | В | Г | Д |
|---|---|---|---|---|
| 10 | 11 | 001 | 010 | 011 |

   Найдите все способы декодирования сообщения 1111001011.
   (Ответ: ББВД)
3. Для кодирования сообщения используется таблица

| А | Б | В | Г | Д |
|---|---|---|---|---|
| 01 | 11 | 100 | 010 | 110 |

   Найдите все способы декодирования сообщения 1111001001100.
   (Ответ: БДГАВ)
4. Для кодирования сообщения используется таблица

| А | Б | В | Г | Д |
|---|---|---|---|---|
| 0 | 11 | 101 | 110 | 111 |

   Найдите все способы декодирования сообщения 1111001010.
   (Ответ: ББААВА, БГАВА)
5. Для кодирования сообщения используется таблица

| А | Б | В | Г | Д |
|---|---|---|---|---|
| 0 | 10 | 1 | 110 | 111 |

   Найдите все способы декодирования сообщения 01110011.
   (Ответ: АВВБАВВ, АВВВААВВ, АВГАВВ, АДААВВ)
6. Для кодирования сообщения используется таблица

[6] http://www.stihi.ru/2007/01/10-1192

| A | B | C | D | E |
|---|---|---|---|---|
| 000 | 01 | 100 | 10 | 011 |

Декодируйте сообщение 0110100011000. (Ответ: BDCEA)

7. Для кодирования сообщения, состоящего только из букв A, B, C, D и E, используется неравномерный двоичный код:

| A | B | C | D | E |
|---|---|---|---|---|
| 000 | 11 | 01 | 001 | 10 |

Какие из этих сообщений были переданы без ошибок:

1) 110000010011110
2) 110000011011110
3) 110001001001110
4) 110000001011110

(Ответ: 1)

8. *Для передачи по каналу связи сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный код: А = 0, Б = 10, В = 110. Как нужно закодировать букву Г, чтобы длина кода была минимальной и допускалось однозначное разбиение кодированного сообщения на буквы? (Ответ: 111)
9. *Для передачи по каналу связи сообщения, состоящего только из букв А, Б, В, Г, решили использовать неравномерный код: А = 0, Б = 100, В = 101. Как нужно закодировать букву Г, чтобы длина кода была минимальной и допускалось однозначное разбиение кодированного сообщения на буквы? (Ответ: 11)

## 2.5 Алфавитный подход к оценке количества информации

Представьте себе, что вы много раз бросаете монету и записываете результат очередного броска как 1 (если монета упала гербом) или 0 (если она упала «решкой»). В результате получится некоторое *сообщение* – цепочка нулей и единиц: 0101001101001110. Вы наверняка поняли, что здесь используется *двоичное* кодирование – это сообщение написано на языке, *алфавит* которого состоит из двух символов, 0 и 1.

Сколько информации в таком сообщении? Чтобы ответить на этот вопрос, рассмотрим один символ, который может быть единицей или нулем. Следовательно, получив такой символ, мы выберем один из двух вариантов, иначе говоря, получим информацию в 1 бит. Полная информация в сообщении 0101001101001110 равна 16 бит, потому что каждый из 16 знаков несет 1 бит.

Теперь представим себе, что нужно закодировать программу для Робота, который умеет выполнять команды «вперед», «назад», «влево» и «вправо». Для этого можно использовать алфавит, состоящий из 4 символов: ↑↓→←. Сколько информации содержится в сообщении ↑←↑↑→↓↓↓↓→←? Каждый полученный знак может быть любым из 4-х символов алфавита. При выборе одного из 4-х вариантов мы получаем 2 бита информации, поэтому полное сообщение (из 11 символов) содержит 22 бита информации.

Алфавитный подход к оценке количества информации состоит в следующем:

1) определяем мощность алфавита $M$ (количество символов в алфавите);
2) по таблице степеней числа 2 определяем количество бит информации $i$, приходящихся на каждый символ сообщения:

| $M$, символов | 2 | 4 | 8 | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
|---|---|---|---|---|---|---|---|---|---|---|
| $i$, бит | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

3) умножаем $i$ на число символов в сообщении, это и есть полное количество информации:

$$I = N \cdot i .$$

Обратим внимание на две важные особенности алфавитного подхода.

При использовании алфавитного подхода не учитывается, что некоторые символы могут встречаться в сообщении чаще других. Считается, что каждый символ несет одинаковое количество информации.

Алфавитный подход не учитывает также частоты появления *сочетаний символов* (например, после гласных букв никогда не встречается мягкий знак).

Кроме того, никак не учитывается смысл сообщения, оно представляет собой просто набор знаков, которые приемник, возможно, даже не понимает.

При использовании алфавитного подхода смысл сообщения не учитывается. Количество информации определяется только длиной сообщения и мощностью алфавита.

Во многих задачах такой подход очень удобен. Например, для устройств, передающих информацию по сети, ее содержание не имеет никакого значения, важен только объем. Почтальону все равно, что написано в письмах, важно только их количество, которое влияет на вес сумки. Для компьютера все данные – это последовательности нулей и единиц, их смысла он не понимает.

Для вычисления информационного объема текста чаще всего применяют именно алфавитный подход. Например, пусть требуется оценить количество информации в 10 страницах текста (на каждой странице 32 строки по 64 символа) при использовании алфавита из 256 символов. Задача решается так:

- определяем информационную емкость одного символа: так как $256 = 2^8$, один символ несет $i = 8$ бит или 1 байт информации;
- считаем количество символов на одной странице, в данном случае удобно использовать степени числа 2 ($32 = 2^5$, $64 = 2^6$): $2^5 \cdot 2^6 = 2^{11}$ символов на странице;
- находим общее количество символов на 10 страницах: $N = 10 \cdot 2^{11}$ символов;
- определяем информационный объем всего текста:

$$I = N \cdot i = 10 \cdot 2^{11} \cdot 1 \text{ байт} = 10 \cdot 2^{11} \text{ байт} = 10 \cdot 2^{11} \cdot (1/2^{10} \text{ Кбайт}) = 20 \text{ Кбайт}.$$

## ? *Контрольные вопросы*

1. В чем состоит алфавитный подход к оценке количества информации?
2. Приведите примеры ситуаций, когда смысл информации особого значения не имеет, а важен только ее объем.
3. Учитывается ли при алфавитном подходе частота встречаемости символов в тексте?
4. Отметьте все утверждения, справедливые для алфавитного подхода:
   а) количество информации зависит от длины сообщения;
   б) количество информации зависит от мощности алфавита;
   в) чем больше мощность алфавита, тем больше количество информации;
   г) важен смысл сообщения;
   д) сообщение должно быть понятно для приемника;
   е) разные символы могут нести разное количество информации.

   (Ответ: а, б, в)
5. Технический документ перевели с одного языка на другой (считаем, что это было сделано максимально близко к тексту). Изменился ли смысл документа? Изменился ли его объем?
6. Как вы думаете, почему компьютеру легко извлечь несколько предложений с конкретных страниц документа, но трудно составить аннотацию к нему?

### *Задачи*

1. Сообщение состоит из 100 символов, используется алфавит, состоящий из 64 знаков. Каков информационный объем этого сообщения? (Ответ: 600 бит)
2. Дан текст из 600 символов. Известно, что символы берутся из таблицы размером 16 на 32, в которой все ячейки заполнены разными знаками. Определите информационный объем текста в битах. (Ответ: 5400)
3. Для записи текста использовался алфавит, состоящий из 32 символов. Каждая страница текста содержит 32 строки. Информационный объем сообщения, состоящего из 5 страниц, составил 6400 байт. Сколько символов в каждой строке текста? (Ответ: 64)
4. Страница текста содержит 30 строк по 60 символов в каждой. Сообщение, состоящее из 4 страниц текста, имеет информационный объем 6300 байт. Какова мощность алфавита? (Ответ: 128)
5. Мощность алфавита равна 256. Сколько Кбайт памяти потребуется для сохранения 160 страниц текста, содержащего в среднем 192 символа на каждой странице? (Ответ: 30)
6. Мощность алфавита равна 64. Сколько Кбайт памяти потребуется, чтобы сохранить 128 страниц текста, содержащего в среднем 256 символов на каждой странице? (Ответ: 24)
7. Секретарь может набирать текст со скоростью 256 символов в минуту. Сколько Кбайт информации он сможет ввести в компьютер за 10 минут, если используется алфавит из 256 символов? (Ответ: 2,5 Кбайт)
8. Для кодирования секретного сообщения используются 12 специальных знаков. При этом символы кодируются одним и тем же минимально возможным количеством бит. Чему равен информационный объем сообщения длиной в 256 символов? (Ответ: 128 байт)
9. Для кодирования нотной записи используется 7 знаков-нот. Каждая нота кодируется одним и тем же минимально возможным количеством бит. Чему равен информационный объем сообщения, состоящего из 180 нот (в битах)? (Ответ: 540 бит)
10. Объем сообщения – 7,5 Кбайт. Известно, что данное сообщение содержит 7680 символов. Какова мощность алфавита? (Ответ: 256)
11. Объем сообщения равен 12 Кбайт. Сообщение содержит 16384 символа. Какова мощность алфавита? (Ответ: 64)
12. Объем сообщения, содержащего 4096 символов, равен 1/512 части Мбайта. Какова мощность алфавита, с помощью которого записано это сообщение? (Ответ: 16)
13. Два текста содержат одинаковое количество символов. Первый текст составлен в алфавите мощностью 16 символов, а второй текст – в алфавите из 256 символов. Во сколько раз количество информации во втором тексте больше, чем в первом? (Ответ: в 2 раза)
14. Алфавит языка первого племени содержит 8 знаков, а алфавит языка второго племени – 16. Племена обменялись сообщениями, состоящими из одинакового количества символов. Известно, что сообщение второго племени содержало 128 байт информации. Каков информационный объем сообщения первого племени? (Ответ: 96 байт)
15. *Два текста содержат одинаковое количество символов, но информационный объем второго текста в 1,5 раза больше, чем первого. Определите мощности алфавитов, если известно, что в обоих текстах число символов меньше 10, и на каждый символ приходится целое число бит. (Ответ: 4 и 8)
16. *Два текста имеют одинаковый информационный объем, но количество символов во втором тексте в 3,5 раза больше, чем в первом. Определите мощности алфавитов, если известно, что в обоих текстах число символов меньше 200, и на каждый символ приходится целое число бит. (Ответ: 128 и 4)

## 2.6 Системы счисления

**Система счисления** – это правила записи чисел с помощью специальных знаков – цифр, а также

соответствующие правила выполнения операций с этими числами.

Первоначально люди считали на пальцах – это самый простой способ, который используется по сей день. Один загнутый (или отогнутый) палец обозначал единицу (один день, одного человека, одного барана и т.п.). Такая система счисления называется *унарной* (от латинского слова *unus* – один). В качестве цифры можно использовать камешки, узелки, счетные палочки (как в начальной школе), зарубки на дереве (как Робинзон Крузо) или на кости, черточки на бумаге, точки и другие одинаковые знаки или предметы.

С помощью унарной системы можно записывать только натуральные числа, причем запись больших чисел получается очень длинной (представьте себе, как записать миллион?). Любая цифра в унарной системе всегда обозначает единицу, поэтому это одна из *непозиционных* систем счисления.

**Непозиционная система счисления** – это такая система, в которой значение цифры не зависит от ее места в записи числа.

К непозиционным относится и десятичная египетская система счисления. Египтяне ввели 7 знаков-иероглифов, которые обозначали степени числа 10 (черточка, хомут, веревка, лотос, палец, лягушка, человек)[7]:

1 10 $10^2$ $10^3$ $10^4$ $10^5$ $10^6$

В этой системе, например, число 235 записывалось как 

В *римской системе* (она также считается непозиционной) в качестве цифр используются латинские буквы: I обозначает 1, V — 5, X — 10, L — 50, C — 100, D — 500, и M — 1000. Единицы, десятки, сотни и тысячи кодировались отдельно, например,

2368 = 2000 + 300 + 60 + 8 =

= (1000 + 1000) + (100 + 100 + 100) + (50 + 10) + (5 + 1 + 1 + 1) = MMCCCLXVIII

Больше трех цифр одинаковых подряд не ставили, поэтому число 4 записывали как IV. В такой записи меньшая цифра (I) стоит перед большей (V), поэтому она вычитается из нее. То есть

IV = 5 – 1 = 4.

Аналогично записывались числа 9, 40, 90, 400 и 900:

IX = 10 – 1 = 9, XL = 50 – 10 = 40, XC = 100 – 10 = 90,

CD = 500 – 100 = 400, MC = 1000 – 100 = 900.

Часы Московского Кремля

Из-за этой особенности римскую систему нельзя считать полностью непозиционной, потому что значение меньшей цифры, стоящей слева от большей, меняется на отрицательное.

У римской системы есть несколько серьезных недостатков:

- можно записывать только натуральные числа (что делать с дробными и отрицательными?);
- чтобы записывать большие числа, необходимо вводить все новые и новые цифры (иногда использовались цифры с подчеркиванием или чертой сверху, что обозначало увеличение в 1000 раз: V — 5000, X — 10000 и т.д.);
- сложно выполнять арифметические действия.

Сейчас римская система применяется для нумерации веков (XXI век), глав в книгах, на циферблатах часов (например, на Спасской башне Кремля).

[7] http://en.wikipedia.org/wiki/Egyptian_numeral_system

В славянской системе счисления в качестве цифр использовались буквы алфавита, над которыми ставился знак ҃ («титло»):



Часы Суздальского Кремля

Если в ряд стояло несколько цифр, знак титло ставился только у первой. Старшие цифры записывались *справа* от младших, например, число 11 записывалось как **а҃і** (см. циферблат часов Суздальского Кремля на рисунке).

## *Контрольные вопросы*

1. Как можно закончить фразу: «Система счисления – это…»:
   а) правила записи чисел с помощью знаков-цифр;
   б) правила записи чисел с помощью цифр 0..9;
   в) набор цифр;
   г) множество натуральных чисел.
2. Что такое унарная система счисления? Приведите примеры.
3. Какие недостатки присущи унарной системе счисления?
4. Что такое непозиционная система счисления?
5. Какие системы счисления относятся к непозиционным?
6. Какое наибольшее число можно записать в классической римской системе?
7. Какие цифры использовались в римской системе? Что они означают?
8. Можно ли называть римскую систему непозиционной? Обоснуйте ответ.
9. Где сейчас используется римская система?
10. Перечислите недостатки римской системы счисления. Как вы думаете, почему ее не используют в компьютерах?

## *Задачи*

1. Переведите в римскую систему числа 12345, 2999, 2444, 2888, 3777.
2. Переведите в десятичную систему числа MCDXCIX, MMDCCXLVII, MDCXCIX.
3. Запишите в славянской системе числа 15, 25, 38, 137, 596.

# 2.7 Позиционные системы счисления

## *2.7.1 Основные понятия*

**Позиционная система счисления** – это такая система, в которой значение цифры полностью определяется ее местом (позицией) в записи числа.

Пример позиционной системы счисления – привычная нам десятичная система. В числе 6375 цифра 6 обозначает тысячи (то есть 6000), цифра 3 – сотни (300), цифра 7 – десятки (70), а цифра 5 – единицы:

$$6375 = 6 \cdot 1000 + 3 \cdot 100 + 7 \cdot 10 + 5 \cdot 1$$

**Алфавит** – это набор цифр, используемых в системе счисления.
**Основание** – это количество цифр в алфавите (мощность алфавита).

В десятичной системе основание – 10, используется *алфавит* из 10 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9. Число 10, вероятно, было выбрано потому, что люди сначала использовали для счета свои 10 пальцев на руках.

**Разряд** – это позиция цифры в записи числа. Разряды в записи целых чисел нумеруются с нуля справа налево.

В числе 6375 цифра 6 стоит в третьем разряде (тысячи, $10^3$), 3 – во втором разряде (сотни, $10^2$), 7 – в первом (десятки, $10^1$), а 5 – в нулевом (единицы, $10^0$). Не забывайте, что любое число (кроме нуля!) в нулевой степени равно 1. Поэтому

$$\begin{matrix}\textit{разряды} \rightarrow & 3 & 2 & 1 & 0 \\ 6375 & = & 6\cdot10^3 & + & 3\cdot10^2 + 7\cdot10^1 + 5\cdot10^0\end{matrix}$$

Это так называемая *развернутая форма* записи числа. Из этой записи видно, что последняя цифра 5 – это остаток от деления числа на 10 (все остальные слагаемые делятся на 10); число, составленное из двух последних цифр (75) – это остаток от деления исходного числа на $100 = 10^2$ и т.д. Поэтому все числа, делящиеся на 100 без остатка, оканчиваются на два нуля.

Чтобы определить число, записанное в позиционной системе счисления, нужно значение каждой цифры умножить на основание системы счисления в степени, равной разряду, и сложить полученные величины.

Число 6375 можно представить в другой форме (*схема Горнера*):

$$6375 = ((6\cdot10 + 3)\cdot10 + 7)\cdot10 + 5$$

Она позволяет найти число, используя только умножение и деление (без возведения в степень).

Кроме десятичной, на практике используются еще несколько позиционных систем:

- двоичная, восьмеричная и шестнадцатеричная в компьютерной технике;
- двенадцатеричная английская система мер (1 фут = 12 дюймов, 1 шиллинг = 12 пенсов);
- шестидесятеричная в часах (1 час = 60 минут, 1 минута = 60 секунд).

## *2.7.2 Целые числа*

Теперь можно записать аналогичные выражения для системы счисления с любым натуральным основанием $p > 1$. Ее алфавит состоит из $p$ цифр[8] от 0 до $p - 1$, то есть «старшая» (наибольшая) цифра в позиционной системе счисления на единицу меньше, чем основание.

Рассмотрим четырехзначное число $a_3a_2a_1a_0$, записанное в системе счисления с основанием $p$. Здесь $a_3$, $a_2$, $a_1$ и $a_0$ – это отдельные цифры, стоящие соответственно в третьем, втором, первом и нулевом разрядах. Это число может быть записано в развернутой форме

$$\begin{matrix}\textit{разряды} \rightarrow & 3 & 2 & 1 & 0 \\ a_3a_2a_1a_0 & = & a_3\cdot p^3 & + & a_2\cdot p^2 + a_1\cdot p^1 + a_0\cdot p^0\end{matrix}$$

или с помощью схемы Горнера:

$$a_3a_2a_1a_0 = ((a_3\cdot p + a_2)\cdot p + a_1)\cdot p + a_0$$

Оба способа можно использовать для **перевода числа в десятичную систему**. Например, пусть число $1234_5$ записано в пятеричной системе счисления (с основанием 5). Нижний индекс «5» в записи $1234_5$ обозначает основание системы счисления (для десятичной системы основание не указывают). Тогда

[8] При $p > 10$ используются также и латинские буквы, но об этом далее.

$$1234_5 = 1\cdot5^3 + 2\cdot5^2 + 3\cdot5^1 + 5\cdot5^0 = 125 + 2\cdot25 + 3\cdot5 + 4 = 194$$

$$1234_5 = ((1\cdot5 + 2)\cdot5 + 3)\cdot5 + 4 = (7\cdot5 + 3)\cdot5 + 4 = 38\cdot5 + 4 = 194$$

Развернутую запись числа можно использовать для обратного перехода, **от десятичной системы к системе с основанием** $p$. Действительно из формулы

$$a_3a_2a_1a_0 = a_3\cdot p^3 + a_2\cdot p^2 + a_1\cdot p + a_0$$

следует, что $a_0$ – это остаток от деления исходного числа на основание $p$. Если мы разделим исходное число на $p$ и отбросим остаток, мы «отбросим» последнюю цифру числа и получим

$$a_3a_2a_1 = a_3\cdot p^2 + a_2\cdot p + a_1$$

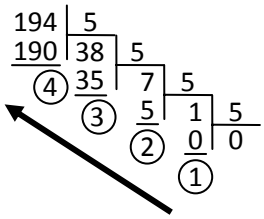
Теперь легко найти $a_1$ – это последняя цифра получившегося числа, которая, как мы знаем, равна остатку от его деления на $p$. Снова разделив на $p$ и отбросив остаток, получим число

$$a_3a_2 = a_3\cdot p + a_2$$

из которого найдем $a_2$ как остаток от деления на $p$. Разделив на $p$ еще раз, получаем последнюю цифру $a_3$.

Например, возьмем число 194 и переведем его в пятеричную систему счисления ( $p = 5$ ). Найдем остаток от деления на 5:

$$194 = 38\cdot5 + 4.$$




Таким образом, мы нашли последнюю цифру – 4. Частное равно 38, повторяем ту же операцию:

$$38 = 7\cdot5 + 3.$$

Следующая (с конца) цифра числа – 3. Дальше получаем

$$7 = 1\cdot5 + 2,$$

третья с конца цифра – 2, а четвертая – 1 (единица уже не делится на 5). Обратим внимание, что с помощью этого способа мы находим цифры числа, начиная с последней. Поэтому полученные остатки нужно выписать в обратном порядке (см. схему на рисунке). Ответ: $1234_5$.

Для перевода числа из десятичной системы в систему счисления с основанием $p$ нужно делить это число на $p$, отбрасывая остаток на каждом шаге, пока не получится 0. Затем остается выписать найденные остатки в обратном порядке.

Можно было заметить, что такой алгоритм фактически использует схему Горнера, «раскручивая» ее в обратном порядке. При каждом делении частное и остаток определяются однозначно, поэтому представление числа в любой позиционной системе единственно.

**Пример 1.** Зная десятичное число и его запись в некоторой позиционной системе счисления, можно найти основание этой системы. Пусть, например, число 71 в некоторой системе с основанием $x$ записывается как $56_x$. Представим это число в развернутой форме:

$$71 = 56_x = 5\cdot x^1 + 6\cdot x^0 = 5\cdot x + 6.$$

Решая уравнение $71 = 5\cdot x + 6$ относительно неизвестного $x$, получаем $x = 13$. Значит, искомое основание системы – 13.

**Пример 2.** В более сложных случаях может получиться алгебраическое уравнение второй (или еще более высокой) степени. Например, то же число 71 в в некоторой системе с основанием $x$ записывается как $155_x$. Представим это число в развернутой форме:

$$71 = 155_x = 1\cdot x^2 + 5\cdot x^1 + 5\cdot x^0 = x^2 + 5\cdot x + 5.$$

Решая уравнение $71 = x^2 + 5\cdot x + 5$ относительно неизвестного $x$, получаем два решения, $x_1 = -11$ и $x_2 = 6$. Искомое основание положительно, поэтому выбираем ответ 6.

**Пример 3.** Если запись числа в другой системе счисления задана не полностью, решений может быть несколько. Например, найдем все основания систем счисления, в которых запись числа 24 оканчивается на 3. Здесь удобно использовать схему Горнера, из которой сразу следует

$$24 = k \cdot x + 3,$$

где $x$ – неизвестное основание системы счисления, а $k$ – некоторое натуральное число или 0. Отсюда сразу получаем $21 = k \cdot x$, то есть все интересующие нас основания являются делителями числа 21. Это могут быть 3, 7 и 21. Поскольку последняя цифра числа – 3, основание не может быть равно 3 (в троичной системе нет цифры 3), поэтому условию задачи удовлетворяют только основания 7 и 21.

**Пример 4.** Найдем все десятичные числа, не превосходящие 40, запись которых в системе счисления с основанием 4 оканчивается на 11. Используя схему Горнера, находим, что все интересующие нас числа имеют вид

$$N = k \cdot 4^2 + 1 \cdot 4 + 1 = k \cdot 16 + 5,$$

где $k$ – некоторое натуральное число или 0. Подставляя $k$ = 0, 1, 2, 3, …, находим соответствующие числа $N$ = 5, 21, 37, 53, …. Из них только 5, 21 и 37 удовлетворяют условию (не больше 40).

## ? *Контрольные вопросы*

1. Какие системы счисления называют позиционными?
2. Каким термином называется количество цифр в алфавите системе счисления?
3. Что такое разряд? Как нумеруются разряды?
4. Как связан в позиционной системе «вес» цифры и разряд, в котором она стоит?
5. Почему в быту мы чаще всего используем десятичную систему?
6. Какие позиционные системы счисления используются сейчас на практике?
7. Чем хороша схема Горнера с точки зрения вычислений?
8. Как перевести число из любой позиционной системы в десятичную?
9. Какие цифры входят в алфавит девятеричной системы?
10. Как вы думаете, можно ли использовать систему счисления с основанием 1000000? В чем могут быть проблемы?
11. Сформулируйте алгоритм перевода числа из семеричной системы в десятичную.
12. Сформулируйте алгоритм перевода числа из десятичной системы в семеричную.
13. Как по записи числа в пятеричной системе сразу увидеть, делится ли оно на 5? на 25? на 125?

## *Задачи*

1. Запишите число 12345 в развернутой форме и в виде схемы Горнера.
2. Какое минимальное основание должно быть у системы счисления, чтобы в ней существовали числа 123, 463, 153 и 455? Ответ обоснуйте. (Ответ: 7)
3. Выберите наибольшее число: $11_2$, $11_7$, 11, $11_{12}$, $11_{16}$, $11_{25}$.
4. Переведите числа $345_6$, $345_7$, $345_8$ и $345_9$ в десятичную систему. (Ответ: 137, 180, 229, 284)
5. Переведите число 194 в троичную, шестеричную, семеричную и восьмеричную системы счисления. (Ответ: $21012_3$, $522_6$, $365_7$, $302_8$)
6. Какие из чисел $1230_7$, $124_7$, $600_7$, $530_7$ делятся на 7? на 49?
7. Десятичное число, переведенное в восьмеричную и в девятеричную систему, в обоих случаях заканчивается на цифру 0. Какое минимальное десятичное число удовлетворяет этому условию? (Ответ: 72)
8. Сколько всего раз встречается цифра 2 в записи чисел 10, 11, 12, …, 17 в системе счисления с основанием 5? (Ответ: 7)

9. Сколько всего раз встречается цифра 3 в записи чисел 19, 20, 21, …, 33 в системе счисления с основанием 6? (Ответ: 8)
10. В системе счисления с некоторым основанием $x$ число 12 записывается в виде $110_x$. Найдите это основание. (Ответ: 3)
11. Найдите все основания систем счисления, в которых запись числа 29 оканчивается на 5. (Ответ: 6, 8, 12, 24)
12. В системе счисления с некоторым основанием десятичное число 129 записывается как 1004. Найдите это основание. (Ответ: 5)
13. Найдите все десятичные числа, не превосходящие 25, запись которых в двоичной системе счисления оканчивается на 101. (Ответ: 5, 13, 21)
14. Найдите все основания систем счисления, в которых запись числа 30 оканчивается на 8. (Ответ: 11, 22)
15. Найдите все основания систем счисления, в которых запись числа 31 оканчивается на 11. (Ответ: 2, 3, 5, 30)
16. Найдите все основания систем счисления, в которых запись числа 63 оканчивается на 23. (Ответ: 5, 30)
17. Найдите наименьшее основание системы счисления, в которой запись числа 30 трехзначна. (Ответ: 4)
18. Найдите наименьшее основание системы счисления, в которой запись числа 70 трехзначна. (Ответ: 5)
19. Найдите все десятичные числа, не превосходящие 26, запись которых в троичной системе счисления оканчивается на 22? (Ответ: 8, 17, 26)
20. Найдите все десятичные числа, не превосходящие 30, запись которых в четверичной системе счисления оканчивается на 31? (Ответ: 13, 29)
21. *Найдите все десятичные числа, не превосходящие 25, запись которых в системе счисления с основанием 6 начинается на 4. (Ответ: 4, 24, 25)
22. *Найдите все десятичные числа, не превосходящие 30, запись которых в системе счисления с основанием 5 начинается на 3. (Ответ: 3, 15, 16, 17, 18, 19)
23. *Найдите основание системы счисления $x$, для которого выполняется равенство

    а) $32_x + 64_x = 106_x$ в) $42_x + 41_x = 133_x$

    б) $45_x + 55_x = 122_x$ г) $A1_x + A3_x = 184_x$

    (Ответ: 9, 8, 5, 12)

## *2.7.3 Дробные числа*

Дробные числа сначала рассмотрим на примере десятичной системы. Число 0,6375 можно представить в виде

$$0{,}6375 = 6\cdot 0{,}1 + 3\cdot 0{,}01 + 7\cdot 0{,}001 + 5\cdot 0{,}0001.$$

Все множители, на которые умножаются значения цифр, представляют собой *отрицательные* степени числа 10 – основания системы счисления. То есть можно использовать развернутую форму записи, вводя отрицательные разряды:

*разряды* → -1 -2 -3 -4

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

Это число можно представить также с помощью схемы Горнера:

$$0{,}6375 = 10^{-1}\cdot(6 + 10^{-1}\cdot(3 + 10^{-1}\cdot(7 + 10^{-1}\cdot 5))).$$

Рассмотрим дробное число $0{,}a_1a_2a_3a_4$, записанное в системе счисления с основанием $p$. Здесь $a_1$, $a_2$, $a_3$ и $a_4$ – это отдельные цифры, стоящие соответственно в разрядах -1, -2, -3 и -4. Это число может быть записано в развернутой форме

*разряды* → -1 -2 -3 -4

$$0,a_1a_2a_3a_4 = a_1 \cdot p^{-1} + a_2 \cdot p^{-2} + a_3 \cdot p^{-3} + a_4 \cdot p^{-4}$$

или с помощью схемы Горнера:

$$0,a_3a_2a_1a_0 = p^{-1} \cdot (a_1 + p^{-1} \cdot (a_2 + p^{-1} \cdot (a_3 + p^{-1} \cdot a_4))).$$

Умножив это число на $p$, получаем $a_3,a_2a_1a_0$. Если взять целую часть результата, мы получим цифру $a_3$. Таким же способом можно найти оставшиеся цифры дробной части: на каждом шаге берем дробную часть, умножаем ее на $p$ и запоминаем *целую часть* результата – это и будет очередная цифра записи числа в системе с основанием $p$. Например, переведем число 0,9376 в пятеричную систему:

| Вычисления | Целая часть | Дробная часть |
|---|---|---|
| 0,9376 · 5 = 4,688 | ④ | 0,688 |
| 0,688 · 5 = 3,44 | ③ | 0,44 |
| 0,44 · 5 = 2,2 | ② | 0,2 |
| 0,2 · 5 = 1 | ① | 0 |

Чтобы получить ответ, нужно выписать все целые части результатов, полученные на каждом шаге:

$$0,9376 = 0,4321_5.$$

Вычисления заканчиваются, когда при очередном умножении дробная часть результата равна нулю. Это означает, что все остальные цифры дробной части – нули. Всегда ли это произойдет? К сожалению, нет. Чтобы убедиться в этом вы можете перевести в пятеричную систему число 0,3 (должна получиться бесконечная дробь). Такая ситуация может случиться в любой системе счисления (например, вспомните, что число ⅓ записывается в виде бесконечной десятичной дроби).

Если нужно перевести в другую систему число, в котором есть целая и дробная части, эти части переводят отдельно, а потом соединяют. Например, переведем число 25,375 в шестеричную систему:

$$25,375 = 25 + 0,375$$

$$25 = 41_6, \quad 0,375 = 0,213_6 \quad \Rightarrow \quad 25,375 = 41,213_6.$$

## ? *Контрольные вопросы*

1. Сформулируйте алгоритм перевода дробной части в шестеричную систему счисления.
2. Как вы думаете, почему не все конечные десятичные дроби можно представить в виде конечных дробей в других системах счисления?
3. Какие дробные числа можно записать в виде *конечной* дроби в шестеричной системе счисления? Ответ обоснуйте.

## *Задачи*

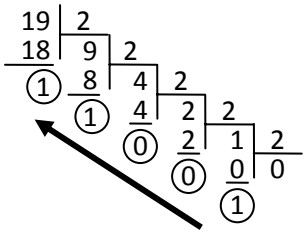
1. Запишите число $0,12321_4$ в развернутой форме и с помощью схемы Горнера.
2. Переведите число 15,125 в двоичную, четверичную, шестеричную и восьмеричную системы. (Ответ: $1111,001_2$; $33,02_4$; $23,043_6$; $17,1_8$)
3. Какие из этих чисел больше, чем ½: $0,011_2$; $0,12_3$; $0,21_4$; $0,22_5$; $0,25_6$; $0,35_7$; $0,35_8$? (Ответ: $0,12_3$; $0,21_4$; $0,35_7$)
4. Переведите числа 11,125; 15,75; 22,6875 и 30,375 в систему счисления с основанием 4. (Ответ: $23,02_4$; $33,3_4$; $112,23_4$; $132,12_4$)

# 2.8 Двоичная система счисления

## *2.8.1 Основные понятия*

В двоичной системе счисления, то есть в системе с основанием 2, алфавит состоит из двух цифр: 0 и 1. Вся числовая информация в компьютерных устройствах хранится и обрабатывается в двоичной системе счисления.

Для перевода натуральных чисел[9] из десятичной системы в двоичную можно использовать общий алгоритм, описанный в предыдущем параграфе (деление на 2 и выписывание остатков в обратном порядке). Например, переведем в двоичную систему число 19:




$19 = 10011_2$

Кроме того, можно использовать так называемый *метод подбора* или *табличный метод* (разложение на сумму степеней двойки). Так в числе 77 старшая степень двойки – это $64 = 2^6$ (следующая степень, $128 = 2^7$, уже больше, чем 77), поэтому

$$77 = 2^6 + 13.$$

Теперь выделяем старшую степень двойки в числе 13: это $8 = 2^3$, так что

$$77 = 2^6 + 2^3 + 5.$$

Выделяем старшую степень двойки в числе 5: это $4 = 2^2$, получаем

$$77 = 2^6 + 2^3 + 2^2 + 1 = 2^6 + 2^3 + 2^2 + 2^0.$$

Мы разложили число на сумму степеней двойки. Для «полного комплекта» здесь не хватает $2^5$, $2^4$ и $2^2$, но можно считать, что эти степени умножаются на ноль:

$$77 = \textcircled{1}\cdot 2^6 + \textcircled{0}\cdot 2^5 + \textcircled{0}\cdot 2^4 + \textcircled{1}\cdot 2^3 + \textcircled{1}\cdot 2^2 + \textcircled{0}\cdot 2^1 + \textcircled{1}\cdot 2^0.$$

Это – развернутая запись числа в двоичной системе счисления, поэтому краткая запись состоит из цифр, обведенных кружками. Единицы стоят в шестом, третьем, первом и нулевом разрядах:

$$\begin{matrix} & 6\,5\,4\,3\,2\,1\,0 \leftarrow \textit{разряды} \\ 77 = & 1001101_2. \end{matrix}$$

Для перевода из двоичной системы в десятичную можно использовать сложение степеней двойки, соответствующих единичным разрядам:

$$\begin{matrix} \textit{разряды} \rightarrow & 6\,5\,4\,3\,2\,1\,0 \\ & 1001101_2 = 2^6 + 2^3 + 2^2 + 2^0 = 64 + 8 + 4 + 1 = 77. \end{matrix}$$

Кроме того, иногда удобно применять схему Горнера. В первом столбце таблицы записывают разряды двоичного числа, начиная со старшего. Вычисления начинаются с 1 (старший разряд всегда равен 1, если число – не ноль). В каждой из следующих строчек результат, полученный в предыдущей строчке, умножается на 2 и к нему добавляется очередной разряд двоичного числа (из первой ячейки той же строки):

| | Вычисления | Результат |
|---|---|---|
| 1 | 1 | |
| 0 | 1·2+0 | 2 |
| 0 | 2·2+0 | 4 |
| 1 | 4·2+1 | 9 |
| 1 | 9·2+1 | 19 |

[9] О хранении отрицательных чисел в памяти компьютера будет рассказано в главе 4.

| | | |
|---|---|---|
| 0 | 19·2+0 | 36 |
| 1 | 38·2+1 | 77 |

## *2.8.2 Арифметические операции*

Двоичные числа, как и десятичные, можно складывать в столбик, начиная с младшего разряда (без перехода к десятичной системе). При этом используют следующие правила:

$$0 + 0 = 0, \quad 1 + 0 = 1, \quad 1 + 1 = 10_2, \quad 1 + 1 + 1 = 11_2.$$

В двух последних случаях, когда получается сумма $2 = 10_2$ или $3 = 11_2$, происходит перенос в следующий разряд.

Например, сложим в столбик $10110_2$ и $111011_2$. Единицы сверху обозначают перенос из предыдущего разряда:

$$\begin{array}{r} {\scriptstyle 1\,1\,1\,1\,1} \\ 10110_2 \\ +\ 111011_2 \\ \hline 1010001_2 \end{array}$$

Вычитание выполняется почти так же, как и в десятичной системе. Вот основные правила:

$$0 - 0 = 0, \quad 1 - 0 = 1, \quad 1 - 1 = 0, \quad 10_2 - 1 = 1.$$

В последнем случае приходится брать заем из предыдущего разряда. Именно этот вариант представляет наибольшие сложности, поэтому мы рассмотрим его подробно.

Чтобы понять принцип, временно вернемся к десятичной системе. Вычтем в столбик из числа 21 число 9:

$$\begin{array}{r} 21 \\ -\ 9 \\ \hline ? \end{array}$$

Поскольку из 1 нельзя вычесть 9, нужно взять заем из предыдущего разряда, в котором стоит 2. В результате к младшему разряду добавляется 10, а в следующем 2 уменьшается до 1. Теперь можно выполнить вычитание: 1 + 10 – 9 = 2. В старшем разряде вычитаем из оставшейся единицы ноль:

$$\begin{array}{r} {\scriptstyle \bullet\quad\quad} \\ {\scriptstyle \searrow 10} \\ 21 \\ -\ 9 \\ \hline 12 \end{array}$$

Здесь точкой сверху обозначен разряд, из которого берется заем.

Более сложный случай – заем из дальнего (не ближайшего) разряда. Вычтем 9 из 2001. В этом случае занять из ближайшего разряда не удается (там 0), поэтому берем заем из того разряда, где стоит цифра 2. Все промежуточные разряды в результате заполняются цифрой 9, это старшая цифра десятичной системы счисления:

$$\begin{array}{r} {\scriptstyle \bullet\quad\quad\quad} \\ {\scriptstyle 1\,9\,9\,10} \\ 2001 \\ -\ 9 \\ \hline 1992 \end{array}$$

Что изменится в двоичной системе? Когда берется заем, в «рабочий» разряд добавляется уже не 10, а $10_2 = 2$ (основание системы счисления), а все «промежуточные» разряды (между «рабочим» и тем, откуда берется заем) заполняются не девятками, а единицами (старшей цифрой системы счисления). Например,

$$\begin{array}{r} {\scriptstyle \bullet\quad\quad\quad} \\ {\scriptstyle 0\,0\,1\,1\,2} \\ 11000_2 \\ -\ 1_2 \\ \hline 10111_2 \end{array} \qquad \begin{array}{r} {\scriptstyle \bullet\quad\quad\quad\quad\bullet} \\ {\scriptstyle 0\,1\,1\,2\,0\,2} \\ 100001_2 \\ -\ 11011_2 \\ \hline 101010_2 \end{array}$$

Если требуется вычесть большее число из меньшего, вычитают меньшее из большего и ставят у результата знак «минус»:

$$\begin{array}{r} 11011_2 \\ -\ 110101_2 \\ \hline ? \end{array} \rightarrow \begin{array}{r} 110101_2 \\ -\ 11011_2 \\ \hline 11010_2 \end{array} \rightarrow \begin{array}{r} 11011_2 \\ -\ 110101_2 \\ \hline \boxed{-\ 11010_2} \end{array}$$

Умножение и деление столбиком в двоичной системе выполняются практически так же, как и в десятичной системе (но с использованием правил двоичного сложения и вычитания):

$$\begin{array}{r} 10101_2 \\ \times\ 101_2 \\ \hline 10101_2 \\ +\ 10101_2 \\ \hline 1101001_2 \end{array} \qquad \begin{array}{r|l} 10101_2 & 111_2 \\ -\ 111_2 & 11_2 \\ \hline 111_2 & \\ 111_2 & \\ \hline 0 & \end{array}$$

## *2.8.3 Дробные числа*

Для перевода дробного числа в двоичную систему используется общий подход, описанный в параграфе 2.7.3. В данном случае нужно умножать число на 2, запоминать целую часть и отбрасывать ее перед следующим умножением. Например, для числа 0,8125 получаем:

| Вычисления | Целая часть | Дробная часть |
|---|---|---|
| 0,8125 · 2 = 1,625 | ① | 0,625 |
| 0,625 · 2 = 1,25 | ① | 0,25 |
| 0,25 · 2 = 0,5 | ⓪ | 0,5 |
| 0,5 · 2 = 1 | ① | 0 |

Таким образом, $0{,}8125 = 0{,}1101_2$.

Давайте посмотрим, как хранится в памяти число 0,6. Выполняя умножение на 2 и выделение целой части, мы получим периодическую бесконечную дробь:

$$0{,}6 = 0{,}100110011001_2\ldots = 0{,}(1001)_2$$

Это значит, что конечное десятичное число 0,6 требует для хранения в двоичном коде *бесконечное* число разрядов. Поскольку реальный компьютер не может иметь бесконечную память, число 0,6 хранится в памяти с ошибкой (погрешностью).

Большинство дробных чисел хранится в памяти с некоторой ошибкой. При выполнении вычислений с дробными числами ошибки накапливаются и могут существенно влиять на результат.

Отметим, что эта проблема связана не с двоичной системой, а с ограниченным размером ячейки, отведенной на хранение числа. В любой системе счисления существуют бесконечные дроби, которые не могут быть точно представлены конечным числом разрядов.

Обеспечение точности расчетов с дробными (вещественными) числами – это очень важная и актуальная проблема, пока до конца не решенная. Поэтому всегда рекомендуется сначала попытаться решить задачу, используя только операции с целыми числами. Например, пусть требуется проверить, верно ли, что $A < \sqrt{B}$, где $A$ и $B$ – целые числа. При извлечении квадратного корня мы сразу переходим в область вещественных чисел, где могут возникнуть вычислительные ошибки. Вместо этого можно возвести обе части неравенства в квадрат и проверять равносильное условие $A^2 < B$, используя только операции с целыми числами.

Если же все-таки нужно обязательно использовать дробные числа и нельзя жертвовать точностью, приходится хранить их в нестандартном виде, например, в виде отношения целых чисел (например, 0,6 = 6/10) и вычислять отдельно числители и знаменатели простых дробей, переходя к вещественным числам только при выводе конечного результата. Этот подход

применяется, например, в системах символьных вычислений (например, в *Maple*[10] или *Mathematica*[11]). Однако выполнение таких расчетов занимает очень много времени.

## 2.8.4 Выводы

Двоичная система служит основой всех расчетов в современных компьютерах. Она обладает следующими **преимуществами**:

- для того, чтобы построить компьютер, работающий с двоичными данными, достаточно иметь **устройства с двумя состояниями** (включено-выключено); первыми такими устройствами были электромагнитные реле, сейчас применяются микроэлектронные элементы;
- **надежность и защита от помех** при передаче информации (для приема двоичного кода не нужно измерять сигнал, а надо только определять есть он или нет);
- компьютеру **проще выполнять вычисления** с двоичными числами, нежели с десятичными; например, умножение фактически сводится к многократному сложению, а деление – к вычитанию.

Тем не менее, с точки зрения человека у двоичной системы есть **недостатки**:

- двоичная запись чисел получается **длинная**: например, число 1024 записывается в виде $10000000000_2$ – здесь легко перепутать количество идущих подряд нулей;
- запись **однородна**, то есть содержит только нули и единицы; поэтому при работе с двоичными числами легко ошибиться или запутаться.

## ? Контрольные вопросы

1. Как вы думаете, какие дробные числа могут быть точно представлены в памяти компьютера в двоичном коде?
2. Почему всегда рекомендуется выполнять вычисления, используя только операции с целыми числами числах (если есть такая возможность)?
3. Как можно работать с дробными числами, не теряя в точности? В чем недостатки такого подхода?
4. Сравните преимущества и недостатки использования двоичной системы счисления с точки зрения человека и с точки зрения компьютера.

## Задачи

1. Переведите числа 25, 31, 37, 63, 85, 127, 128 в двоичную систему счисления.
2. Переведите числа $100011_2$, $101101_2$, $110111_2$, $1001011_2$, $1011111_2$, $1101001_2$ в десятичную систему счисления.
3. Сколько единиц в двоичной записи чисел 173, 195, 126, 208?
4. Сколько значащих нулей в двоичной записи чисел 48, 73, 96, 254?
5. Как по записи числа в двоичной системе счисления определить, что оно – четное? делится на 4? на 8? на 32?
6. Выполните сложение в двоичной системе:
   а) $1010111_2 + 110101_2$
   б) $1010111_2 + 110101_2$
   в) $101101_2 + 11111_2$
   г) $10111_2 + 101110_2$
   д) $111011_2 + 11011_2$
   е) $111011_2 + 10011_2$

   Для проверки повторите вычисления, переходя к десятичной системе, а потом преобразуя результат обратно в двоичную.
7. Выполните вычитание в двоичной системе:
   а) $101101_2 - 11111_2$
   б) $11011_2 - 110101_2$
   г) $101011_2 - 11011_2$
   д) $1011_2 - 100101_2$

[10] http://www.maplesoft.com
[11] http://www.wolfram.com

в) $10111_2 - 101110_2$ е) $1001_2 - 101101_2$

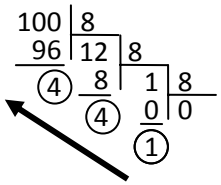
Для проверки повторите вычисления, переходя к десятичной системе, а потом преобразуя результат обратно в двоичную.

8. Переведите в двоичную систему числа 13,125; 23,25; 37,375; 48,625; 78,875.
9. Переведите в двоичную систему числа 11,8; 15,3; 22,7, выделив период в дробной части.
10. Требуется проверить, верно ли, что среднее арифметическое 100 целых чисел превышает 0,2. Как сделать это, не используя операции с дробными числами?

# 2.9 Восьмеричная система счисления

Восьмеричная система (система с основанием 8) использовалась для кодирования команд во многих компьютерах 1950-1980-х годов (например, в американской серии PDP-11, советских компьютерах серий ДВК, СМ ЭВМ, БЭСМ). В ней используются цифры от 0 до 7.

Для перевода десятичного числа в восьмеричную проще всего систему использовать стандартный алгоритм для позиционных систем (деление на 8, выписывание остатков в обратном порядке). Например,




$100 = 144_8$

Для перевода из восьмеричной системы в десятичную значение каждой цифры умножают на 8 в степени, равной разряду этой цифры, и полученные произведения складывают:

*разряды* → 2 1 0

$$144_8 = 1\cdot8^2 + 4\cdot8^1 + 4\cdot8^0 = 64 + 4\cdot8 + 4 = 100.$$

Более интересен перевод из восьмеричной системы в двоичную и обратно. Конечно, можно перевести число сначала в десятичную систему, а потом – в двоичную. Но для этого требуется выполнить две непростых операции, в каждой из них легко ошибиться.

Оказывается, можно сделать перевод из восьмеричной системы в двоичную напрямую, используя тесную связь между этими системами: их основания связаны равенством $2^3 = 8$. Покажем это на примере восьмеричного числа $753_8$. Запишем его в развернутой форме:

$$753_8 = 7\cdot8^2 + 5\cdot8^1 + 3\cdot8^0 = 7\cdot2^6 + 5\cdot2^3 + 3\cdot2^0.$$

Теперь переведем отдельно каждую цифру в двоичную систему:

$$7 = 111_2 = 1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0, \quad 5 = 101_2 = 1\cdot2^2 + 1\cdot2^0, \quad 3 = 11_2 = 1\cdot2^1 + 1\cdot2^0.$$

И подставим эти выражения в предыдущее равенство:

$$753_8 = (1\cdot2^2 + 1\cdot2^1 + 1\cdot2^0)\cdot2^6 + (1\cdot2^2 + 1\cdot2^0)\cdot2^3 + (1\cdot2^1 + 1\cdot2^0)\cdot2^0.$$

Раскрывая скобки, мы получим разложение исходного числа по степеням двойки, то есть его запись в двоичной системе счисления (здесь добавлены нулевые слагаемые для отсутствующих степеней числа 2):

$$753_8 = \textcircled{1}\cdot2^8 + \textcircled{1}\cdot2^7 + \textcircled{1}\cdot2^6 + \textcircled{1}\cdot2^5 + \textcircled{0}\cdot2^4 + \textcircled{1}\cdot2^3 + \textcircled{0}\cdot2^2 + \textcircled{1}\cdot2^1 + \textcircled{1}\cdot2^0.$$

Таким образом, $753_8 = 111\ 101\ 011_2$. Двоичная запись разбита на *триады* (группы из трех цифр), каждая триада – это двоичная запись *одной цифры* исходного восьмеричного числа.

**Алгоритм перевода восьмеричного числа в двоичную систему счисления**.

1. Перевести каждую цифру (отдельно) в двоичную систему. Записать результаты в виде триады, добавив, если нужно, нули в начале. (см. таблицу справа)
2. Соединить триады в одно «длинное» двоичное число.

| | |
|---|---|
| 0 | 000 |
| 1 | 001 |
| 2 | 010 |
| 3 | 011 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |

Например, $35721_8 = 11\ 101\ 111\ 010\ \ 001_2$. В этой записи триады специально отделены друг от друга пробелом. Обратите внимание, что все триады дополнены спереди нулями до трех цифр:

$$2 = 10_2 = 010_2, \quad 1 = 1_2 = 001_2.$$

Для самой первой триады это делать не обязательно, потому что лидирующие нули в записи числа никак его не меняют. Напротив, если «потерять» нули в середине числа, получится неверный результат.

**Алгоритм перевода двоичного числа в восьмеричную систему счисления**.

1. Разбить двоичное число на триады, **начиная справа**. В начало самой первой триады добавить лидирующие нули, если это необходимо.
2. Перевести каждую триаду в восьмеричную (= десятичную) систему.
3. Соединить полученные цифры в одно «длинное» число.

Например, переведем в восьмеричную систему число $1010011100101110111_2$. Разобьем его на триады (начиная справа), к первой триаде нужно добавить два нуля (они подчеркнуты):

$$1010011100101110111_2 = \underline{00}1\ 010\ 011\ 100\ 101\ 110\ 111_2$$

Далее по таблице (см. выше) переводим каждую триаду в восьмеричную систему:

$$1010011100101110111_2 = 1234567_8.$$

Теперь представьте себе объем вычислений, который потребуется для решения этой задачи через десятичную систему.

При вычислениях в восьмеричной системе нужно помнить, что максимальная цифра – это 7. Перенос при сложении возникает тогда, когда сумма в очередном разряде получается больше 7. Заем из старшего разряда равен $10_8 = 8$, а все «промежуточные» разряды заполняются цифрой 7 – старшей цифрой системы счисления. Приведем примеры сложения и вычитания:

$$\begin{array}{r} {\scriptstyle 111} \\ 356_8 \\ +\ 4662_8 \\ \hline 5240_8 \end{array} \quad \begin{array}{l} 6 + 2 = 1\cdot 8 + \textcircled{0} \\ 5 + 6 + 1 = 1\cdot 8 + \textcircled{4} \\ 3 + 6 + 1 = 1\cdot 8 + \textcircled{2} \\ 0 + 4 + 1 = \textcircled{5} \end{array} \qquad \begin{array}{r} {\scriptstyle \bullet\bullet} \\ 456_8 \\ -\ 277_8 \\ \hline 157_8 \end{array} \quad \begin{array}{l} (6 + 8) - 7 = \textcircled{7} \\ (5 - 1 + 8) - 7 = \textcircled{5} \\ (4 - 1) - 2 = \textcircled{1} \end{array}$$

В примере на сложение запись $1\cdot 8 + 2$ означает, что получилась сумма, большая 7, которая не помещается в один разряд. Единица идет в перенос, а двойка остается в этом разряде. При вычитании «– 1» означает, что из этого разряда раньше был заем (его значение уменьшилось на 1), а «+ 8» – заем из следующего разряда.

С помощью восьмеричной системы удобно кратко записывать содержимое областей памяти, содержащих, количество бит, кратное трем. Например, 6-битные данные «упаковываются» в две восьмеричные цифры. Некоторые компьютеры 1960-х годов использовали 24-битные и 36-битные данные, они записывались соответственно с помощью 8 и 12 восьмеричных цифр. Восьмеричная система использовалась даже для компьютеров с 8-битным байтом (PDP-11, ДВК), но позднее была почти вытеснена шестнадцатеричной системой (см. далее).

Сейчас восьмеричная система применяется, например, для установки прав на доступ к файлу в *Linux* (и других *Unix*-системах) с помощью команды **`chmod`**. Режим доступа кодируется тремя битами, которые разрешают чтение (**`r`**, *read*, старший бит), запись (**`w`**, *write*) и выполнение

файла (**x**, *execute*, младший бит). Код 7 = $111_2$ (**rwx**) означает, что все биты установлены (полный доступ), а код 5 = $101_2$ (**r-x**) разрешает чтение и выполнение файла, но запрещает его изменение.

### *Контрольные вопросы*

1. Какие цифры составляют алфавит восьмеричной системы?
2. Сколько существует различных двузначных восьмеричных чисел?

### *Задачи*

1. Переведите числа 49, 53, 64, 150, 266 в восьмеричную и двоичную системы счисления.
2. Переведите числа $123_8$, $234_8$, $345_8$, $456_8$ и $567_8$ в десятичную и двоичную системы счисления.
3. Запишите числа $101111001_2$, $10110100_2$, $1000011_2$, $10101010_2$ в восьмеричной и десятичной системах счисления.
4. Вычислите значения следующих выражений:
   а) $353_8 + 736_8$ в) $1153_8 - 662_8$
   б) $1353_8 + 777_8$ г) $153_8 - 662_8$
5. Вычислите значения следующих выражений, запишите результат в двоичной, восьмеричной и десятичной системах счисления:
   а) $45_8 + 1010110_2$ е) $153_8 - 16 \cdot 101_2$
   б) $271_8 + 11110100_2$ ж) $15_8 \cdot 110_2$
   в) $110111_2 + 135_8$ з) $50_8 \cdot 21_8$
   г) $10 + 10_8 \cdot 10_2$ и) $34_8 : 10101_2$
   д) $123 + 12_8 \cdot 11_2$ к) $214_8 : 1110_2$
6. *Переведите число 12,5 в восьмеричную систему счисления.

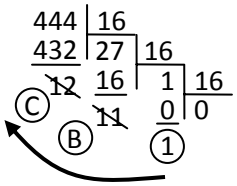
# 2.10 Шестнадцатеричная система счисления

Шестнадцатеричная система (позиционная система с основанием 16) широко используется для записи адресов и содержимого ячеек памяти компьютера. Ее алфавит содержит 16 цифр, вместе с 10 арабскими цифрами (0..9) используются первые буквы латинского алфавита:

A = 10, B = 11, C = 12, D = 13, E = 14 и F = 15.

Таким образом, старшая цифра – F.

Для перевода чисел из десятичной системы в шестнадцатеричную используют алгоритм деления на 16 и взятия остатков. Важно не забыть, что все остатки, большие 9, нужно заменить на буквы:




$444 = 1BC_{16}$

Для обратного перехода значение каждой цифры умножают на 16 в степени, равной ее разряду, и полученные значения складывают:

*разряды* → 2 1 0

$$1BC_{16} = 1 \cdot 16^2 + 11 \cdot 16^1 + 12 \cdot 8^0 = 256 + 176 + 12 = 444.$$

Можно также использовать схему Горнера:

$$1BC_{16} = (1 \cdot 16 + 11) \cdot 16 + 12 = 27 \cdot 16 + 12 = 444.$$

Основания двоичной и шестнадцатеричной систем связаны соотношением $2^4 = 16$, поэтому можно переводить числа из шестнадцатеричной системы в двоичную напрямую: каждая шестнадцатеричная цифра представляется в виде *тетрады* (группы из четырех двоичных цифр). Для этого можно использовать таблицу

| | | | |
|---|---|---|---|
| 0 | 0000 | 8 | 1000 |
| 1 | 0001 | 9 | 1001 |
| 2 | 0010 | A (10) | 1010 |
| 3 | 0011 | B (11) | 1011 |
| 4 | 0100 | C (12) | 1100 |
| 5 | 0101 | D (13) | 1101 |
| 6 | 0110 | E (14) | 1110 |
| 7 | 0111 | F (15) | 1111 |

Например, переведем в двоичную систему число $5E123_{16}$ (здесь показана разбивка на тетрады):

$$5E123_{16} = 101\ 1110\ 0001\ 0010\ 0011_2.$$

Обратите внимание, что для цифр, меньших 8 (кроме первой), результат перевода в двоичную систему нужно дополнить старшими нулями до 4 знаков.

Для перевода из двоичной системы в шестнадцатеричную нужно разбить число на тетрады, **начиная справа**, а затем каждую тетраду отдельно записать в виде одной шестнадцатеричной цифры:

$$1000010000101010111100_2 = 10\ 0001\ 0000\ 1010\ 1011\ 1100_2 = 210ABC_{16}$$

Этот способ оказался очень удобен для записи значений ячеек памяти. Байт в современных компьютерах представляет собой 8 соседних бит, то есть две тетрады. Таким образом, значение байтовой ячейки можно записать как две шестнадцатеричные цифры:

| 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| | | 5 | | | E | | |

Каждый полубайт (4 бита) «упаковывается» в одну шестнадцатеричную цифру. Благодаря этому замечательному свойству, шестнадцатеричная система в сфере компьютерной техники практически полностью вытеснила восьмеричную[12].

Перевод из шестнадцатеричной системы в восьмеричную (и обратно) удобнее выполнять через двоичную систему. Можно, конечно, использовать и десятичную систему, но в этом случае объем вычислений будет значительно больше.

При выполнении сложения нужно помнить, что в системе с основанием 16 перенос появляется тогда, когда сумма в очередном разряде превышает 15. Удобно сначала переписать исходные числа, заменив все буквы на их численные значения:

$$\begin{array}{r} A5B_{16} \\ +\ C7E_{16} \\ \hline 16D9_{16} \end{array} \qquad \begin{array}{r} {\scriptstyle 1}\quad\ {\scriptstyle 1} \\ 10\ 5\ 11 \\ +\ 12\ 7\ 14 \\ \hline 1\ 6\ 13\ 9 \end{array} \qquad \begin{array}{l} 11 + 14 = 1\cdot 16 + \textcircled{9} \\ 5 + 7 + 1 = 13 = \textcircled{D} \\ 10 + 12 = 1\cdot 16 + \textcircled{6} \\ 0 + 0 + 1 = \textcircled{1} \end{array}$$

При вычитании заем из старшего разряда равен $10_{16} = 16$, а все «промежуточные» разряды заполняются цифрой F – старшей цифрой системы счисления.

$$\begin{array}{r} C5B_{16} \\ -\ A7E_{16} \\ \hline 1DD_{16} \end{array} \qquad \begin{array}{r} {\scriptstyle \bullet}\quad\ {\scriptstyle \bullet} \\ 12\ 5\ 11 \\ -\ 10\ 7\ 14 \\ \hline 1\ 13\ 13 \end{array} \qquad \begin{array}{l} (11 + 16) - 14 = 13 = \textcircled{D} \\ (5 - 1 + 16) - 7 = 13 = \textcircled{D} \\ (12 - 1) - 10 = \textcircled{1} \end{array}$$

Если нужно работать с числами, записанными в разных системах счисления, их сначала приводят к какой-нибудь одной системе. Например, требуется сложить $53_8$ и $56_{16}$ и записать результат в двоичной системе счисления. Здесь можно выполнять сложение в двоичной, восьмеричной, десятичной или шестнадцатеричной системах. Переход к десятичной системе, а потом перевод результата в двоичную трудоемок. Практика показывает, что больше всего ошибок

[12] Начиная с 1964 года, когда шестнадцатеричная система стала широко использоваться в документации на новый компьютер IBM/360.

делается при вычислениях в двоичной системе, поэтому лучше выбирать восьмеричную или шестнадцатеричную систему. Например, переведем число $43_8$ в шестнадцатеричную систему через двоичную:

$$53_8 = 101\ 011_2 = 10\ 1011_2 = 2B_{16}.$$

Теперь сложим:

$$2B_{16} + 56_{16} = 81_{16}$$

и переведем результат в двоичную систему:

$$81_{16} = 1000\ 0001_2.$$

### ? *Контрольные вопросы*

1. Какие цифры используются в шестнадцатеричной системе? Сколько их?
2. Почему появилась необходимость использовать латинские буквы?
3. Какое минимальное основание должно быть у системы счисления, чтобы в ней могли быть записаны числа 123, 4AB, 9A3 и 8455? (Ответ: 12)

### *Задачи*

1. Переведите в двоичную и восьмеричную системы числа $7F1A_{16}$, $C73B_{16}$, $2FE1_{16}$, $A112_{16}$.
2. Переведите в двоичную и шестнадцатеричную системы числа $6172_8$, $5341_8$, $7711_8$, $1234_8$.
3. Переведите в восьмеричную и шестнадцатеричную системы числа
   а) $1110111101010_2$ в) $1111001101111101012$
   б) $1010101101010110_2$ г) $1101101101011111102$
4. Переведите числа 29, 43, 54, 120, 206 в шестнадцатеричную, восьмеричную и двоичную системы счисления.
5. Переведите числа $73_8$, $134_8$, $245_8$, $356_8$ и $467_8$ в шестнадцатеричную, десятичную и двоичную системы счисления.
6. Запишите числа $10110101_2$, $1110100_2$, $1000111_2$, $10111110_2$ в шестнадцатеричной, восьмеричной и десятичной системах счисления.
7. Вычислите значения следующих выражений:
   а) $3AF_{16} + 1CBE_{16}$ г) $1CFB_{16} – 22F_{16}$
   б) $1EA_{16} + 7D7_{16}$ д) $22F_{16} – CFB_{16}$
   в) $A81_{16} + 377_{16}$ е) $1AB_{16} – 2CD_{16}$
8. Вычислите значения следующих выражений, запишите результат в двоичной, восьмеричной, десятичной и шестнадцатеричной системах счисления:
   а) $4F_{16} + 111110_2$ г) $110111_2 + 135_8$
   б) $5A_{16} + 1010111_2$ д) $12_{16} + 12_8 \cdot 11_2$
   в) $256_8 + 2C_{16}$ е) $35_8 + 2C_{16} \cdot 101_2$
9. Вычислите значения следующих выражений, запишите результат в двоичной, восьмеричной, десятичной и шестнадцатеричной системах счисления:
   а) $15_{16} \cdot 110_2$ в) $34_{16} : 32_8$
   б) $2A_{16} \cdot 12_8$ г) $740_8 : 18_{16}$
10. *Переведите числа 49,6875 и 52,9 в шестнадцатеричную систему счисления.

# 2.11 Другие системы счисления

## *2.11.1 Троичная уравновешенная система счисления*

В истории компьютерной техники применялись и другие системы счисления. Например, в 1958 г. была создана электронная вычислительная машина (ЭВМ) «Сетунь» (главный

конструктор – Н.П. Брусенцов[13]), которая использовала троичную систему счисления. Всего в 1960-х годах было выпущено более 50 промышленных образцов ЭВМ «Сетунь».

В троичной уравновешенной системе основание равно 3, используются три цифры: $\overline{1}$ («минус 1»), 0 и 1. Один троичный разряд называется *тритом* (в отличие от двоичного бита). Система называется уравновешенной, потому что с помощью любого числа разрядов можно закодировать равное число положительных им отрицательных чисел, и число ноль. Вот, например, все двухразрядные числа

| | | |
|---|---|---|
| –4 | $\overline{1}\,\overline{1}$ | $= (-1)\cdot 3^1 + (-1)\cdot 3^0$ |
| –3 | $\overline{1}\,0$ | $= (-1)\cdot 3^1 + 0\cdot 3^0$ |
| –2 | $\overline{1}\,1$ | $= (-1)\cdot 3^1 + 1\cdot 3^0$ |
| –1 | $0\,\overline{1}$ | $= 0\cdot 3^1 + (-1)\cdot 3^0$ |
| 0 | 0 0 | $= 0\cdot 3^1 + 0\cdot 3^0$ |
| 1 | 0 1 | $= 0\cdot 3^1 + 1\cdot 3^0$ |
| 2 | $1\,\overline{1}$ | $= 1\cdot 3^1 + (-1)\cdot 3^0$ |
| 3 | 1 0 | $= 1\cdot 3^1 + 0\cdot 3^0$ |
| 4 | 1 1 | $= 1\cdot 3^1 + 1\cdot 3^0$ |

В последнем столбце этой таблицы числа записаны в развернутой форме, которую можно использовать для перевода из троичной уравновешенной системы в десятичную.

Заметьте, что положительные и отрицательные числа кодируются с помощью одних и тех же правил. Это большое преимущество в сравнении с двоичным кодированием, при котором для хранения отрицательных чисел пришлось изобретать специальный код.

Троичная уравновешенная система счисления дает ключ к решению *задачи Баше*, которая была известна еще в XIII веке Леонардо Пизанскому (Фибоначчи):

> *Найти такой набор из 4 гирь, чтобы с их помощью на чашечках равноплечных весов можно было взвесить груз массой от 1 до 40 кг включительно. Гири можно располагать на любой чашке весов.*

Каждая гиря может быть в трех состояниях:

1) лежать на той же чашечке весов, что и груз: в этом случае ее вес вычитается из суммы ( $\overline{1}$ );
2) не участвовать во взвешивании (0);
3) лежать на другой чашке: ее вес добавляется к сумме (1).

Поэтому веса гирь нужно выбрать равными степеням числа 3, то есть 1, 3, 9 и 27 кг.

## 2.11.2 Двоично-десятичная система счисления

Существует еще один простой способ записи десятичных чисел с помощью цифр 0 и 1. Этот способ называется двоично-десятичной системой (ДДС), это нечто среднее между двоичной и десятичной системами. На английском языке такое кодирование называется *binary coded decimal* (BCD) – десятичные числа, закодированные двоичными цифрами.

В ДДС каждая *цифра* десятичного числа записывается двоичными знаками. Но среди цифр 0–9 есть такие, которые занимают 1, 2, 3 и 4 двоичных разряда. Чтобы запись числа была однозначной, и не надо было искать границу между цифрами, на любую цифру отводят 4 бита. Таким образом, 0 записывается как 0000, а 9 – как 1001. Например:

$$9024{,}19 = 1001\ 0000\ 0010\ 0100,\ 0001\ 1001_{\text{ДДС}}$$
$$9\quad 0\quad 2\quad 4\quad 1\quad 9$$

При обратном переводе из ДДС в десятичную систему надо учесть, что каждая цифра занимает 4 бита, и добавить недостающие нули:

$$101010011{,}01111_{\text{ДДС}} = 0001\ 0101\ 0011,\ 0111\ 1000_{\text{ДДС}} = 153{,}78$$

Важно помнить, что запись числа в ДДС не совпадает с его записью в двоичной системе:

$$10101{,}1_{\text{ДДС}} = 15{,}8$$

[13] http://www.trinitas.ru/rus/doc/0226/002a/02260054.htm

$10101{,}1_2 = 16 + 4 + 1 + 0{,}5 = 21{,}5$

Использование ДДС дает следующие **преимущества**:

- двоично-десятичный код очень легко переводить в десятичный, например, для вывода результата на экран;
- просто выполняется умножение и деление на 10, а также округление;
- конченые десятичные дроби записываются точно, без ошибки, поэтому вычисления в ДДС (вместо двоичной системы) дадут тот же результат, что и ручные расчеты человека «на бумажке»; поэтому ДДС используется в калькуляторах.

Есть, однако, и **недостатки**:

- хранение чисел в ДДС требует больше памяти, чем стандартный двоичный код;
- усложняются арифметические операции.

### *Контрольные вопросы*

1. Как поменять знак числа, записанного в уравновешенной системе?
2. Сколько положительных и отрицательных чисел можно закодировать с помощью 5 разрядов в троичной уравновешенной системе счисления? (Ответ: 121 положительное и 121 отрицательное)
3. *Попробуйте сформулировать правила сложения чисел в троичной уравновешенной системе.
4. *Сравните троичную уравновешенную систему счисления с двоичной. Как вы думаете, почему разработчики компьютеров все-таки выбрали двоичный код для хранения данных?
5. *Верно ли, что любая последовательность нулей и единиц может быть числом, закодированным в двоично-десятичной системе? Обоснуйте свой ответ.
6. Какие числа записываются одинаково в двоичной и двоично-десятичной системах счисления?
7. *Сравните двоично-десятичную систему с двоичной. В чем ее преимущества и недостатки?

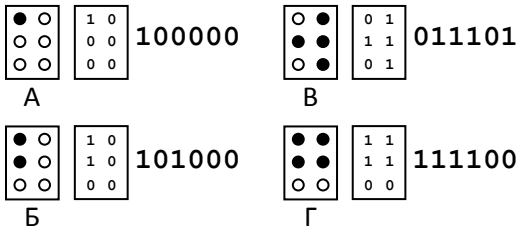
### *Задачи*

1. Запишите числа –15 и 15 в троичной уравновешенной системе. Сколько разрядов вам потребовалось? (Ответ: $\bar{1}110$, $1\bar{1}\bar{1}0$)
2. Найдите минимальный набор гирь, с помощью которых на чашечках равноплечных весов можно было взвесить груз массой от 1 до 13 кг включительно.
3. Закодируйте число 1234 в двоично-десятичной системе счисления. (Ответ: $1001000110100_{ДДС}$)
4. Запишите число $10111100001101001_{ДДС}$ в десятичной системе счисления. (Ответ: 17869)

# 2.12 Кодирование символов

## *2.12.1 Общий подход*

Поскольку в современных компьютерах все виды информации представлены в двоичном коде, нужно разобраться, как закодировать символы в виде цепочек нулей и единиц. Например, можно предложить способ, основанный на системе Брайля для незрячих людей, о которой мы уже упоминали (см. задание 4 к пункту 2.3.2). В нем каждый символ кодируется с помощью 6 точек, расположенных в два столбца. В каждой точке может быть выпуклость, которую чувствует на ощупь. Обозначив выпуклости единицей, а их отсутствие – нулем, можно закодировать первые буквы русского алфавита так:




Здесь двоичный код строится так: строки полученной таблицы, состоящей из цифр 0 и 1, выписываются одна за другой в строчку. Так как используется всего 6 точек, количество символов, которые можно закодировать, равно $2^6$ = 64 (в реальной системе Брайля 63 символа, потому что символ, в коде которого нет ни одной выпуклости, невозможно обнаружить на ощупь.).

Понятно, что совершенно не обязательно использовать код Брайля. Главное – каждому используемому символу как-то сопоставить цепочку нулей и единиц, например, составить таблицу «символ-код». На практике поступают следующим образом:

1) определяют, сколько символов нужно использовать (обозначим это число через $N$);
2) определяют нужно количество $k$ двоичных разрядов так, чтобы с их помощью можно было закодировать не менее $N$ разных последовательностей (то есть $2^k \geq N$);
3) составляют таблицу, в которой каждому символу сопоставляют целое число в интервале от 0 до $2^k-1$ (*код символа*);
4) коды символов переводят в двоичную систему счисления.

В текстовых файлах (которые не содержат оформления, например, в файлах с расширением **`.txt`**) хранятся не изображения символов, а их коды. Откуда же компьютер берет изображения символов, когда выводит текст на экран? Оказывается, при этом с диска загружается шрифтовой файл (он может иметь, например, расширение **`.fon`**, **`.ttf`**, **`.otf`**), в котором хранятся изображения, соответствующие каждому из кодов[14]. Именно эти изображения и выводятся на экран. Это значит, что при изменении шрифта текст, показанный на экране, может выглядеть совсем по-другому. Например, многие шрифты не содержат изображений русских букв. Поэтому, когда вы передаете (или пересылаете) кому-то текстовый файл, нужно убедиться, что у адресата есть использованный вами шрифт. Современные текстовые процессоры умеют *внедрять* шрифты в файл; в этом случае файл содержит не только коды символов, но и шрифтовые файлы. Хотя файл увеличивается в объеме, адресат гарантированно увидит его в таком же виде, что и вы.

## 2.12.2 Кодировка ASCII и ее расширения

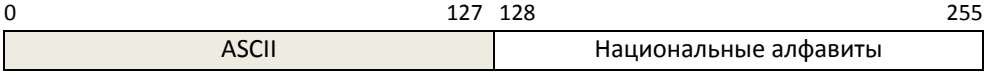
Тем не менее, во многих задачах для передачи и хранения информации необходимо уменьшать ее объем. Если, например, вместе с каждым сообщением в сети передавать шрифтовой файл, скорость обмена информацией будет недопустимо низкой. Поэтому необходим некоторый стандарт, который закреплял бы определенные коды за наиболее часто используемыми символами. Таким международным стандартом является 7-битная кодировка *ASCII* (*(*англ. *American Standard Code for Information Interchange* — американский стандартный код для обмена информацией), в которую входят $2^7$ = 128 символов с кодами от 0 до 127:

- служебные (*управляющие*) символы с кодами от 0 до 31;
- цифры от «0» до «9» с кодами от 48 до 57;
- латинские буквы: заглавные, от «A» до «Z» (с кодами от 65 до 90) и строчные, от «a» до «z» (с кодами от 97 до 122);

[14] Существуют специальные программы, позволяющие создавать и редактировать шрифты, например, *Fontlab Studio* (http://www.fontlab.com/font-editor/fontlab-studio/).

- знаки препинания: **. , : ; ! ?**
- скобки: **[ ] { } ( )**
- математические символы: **+ − * / = < >**
- некоторые другие знаки: **" ' # $ % & ^ | @ \ _ ~**

В современных компьютерах минимальная единица памяти, имеющая собственный адрес – это 8-битный байт. Поэтому для хранения кодов ASCII в памяти можно добавить к ним еще один (старший) нулевой бит, таким образом, получая 8-битную кодировку. Кроме того, дополнительный бит можно использовать: он дает возможность добавить в таблицу еще 128 символов с кодами от 128 до 255. Такое расширение ASCII часто называют *кодовой страницей*. Первую половину кодовой страницы (коды от 0 до 127) занимает стандартная таблица ASCII, а вторую – символы национальных алфавитов (например, русские буквы).




Кодовая страница

Для русского языка существуют несколько кодовых страниц, которые были разработаны для разных операционных систем. Наиболее известны

- кодовая страница *Windows-1251* (CP-1251) – в системе *Windows*;
- кодовая страница *KOI8-R* – в системе *Unix*;
- альтернативная кодировка (CP-1251) – в системе *MS DOS*
- *MacCyrillic* – на компьютерах фирмы *Apple* (*Макинтош* и др.).

Проблема состоит в том, что если набрать русский текст в одной кодировке (например, в *Windows-1251*), а просматривать в другой (например, в *KOI8-R*), текст будет очень сложно прочитать:

| *Windows-1251* | *KOI8-R* |
|---|---|
| Привет, Вася! | оПХБЕР, бЮЯЪ! |
| рТЙЧЕФ, чБУС! | Привет, Вася! |

Для веб-страниц в Интернете чаще всего используют кодировки *Windows-1251* и *KOI8-R*. Браузер после загрузки страницы пытается автоматически определить ее кодировку. Если ему это не удается, вы увидите странный набор букв вместо понятного русского текста. В этом случае нужно сменить кодировку вручную с помощью меню «*Вид*» браузера.

## *2.12.3 Кодировка UNICODE*

Любая 8-битная кодовая страница имеет серьезное ограничение – она может включать только 256 символов. Поэтому не получится набрать в одном документе часть текста на русском языке, а часть – на китайском. Кроме того, существует проблема чтения документов, набранных с использованием другой кодовой страницы. Все это привело к принятию в 1991 году нового стандарта кодирования символов UNICODE, который позволяет одновременно записывать знаки любых существующих и даже некоторых умерших языков, математические и музыкальные символы и др.

Если мы хотим расширить число используемых знаков, необходимо увеличивать место, которое отводится под каждый символ. Вы знаете, что компьютер работает сразу с одним или несколькими байтами, прочитанными из памяти. Поэтому место, отводимое на каждый символ, расширили сразу с одного байта до двух. Это позволило закодировать $2^{16} = 65\,536$ символов в одном наборе. В современной версии UNICODE можно кодировать до $2^{31} = 2\,147\,483\,648$ различных знаков, однако реально используются немногим более 100 000 символов.

В системе *Windows* используется кодировка UNICODE, называемая UTF-16 (от англ. *UNICODE Transformation Format* – формат преобразования UNICODE). В ней на каждый символ отводится 16 бит (2 байта). В *Unix*-подобных системах, например, в *Linux*, чаще применяют кодировку UTF-8. В ней все символы, входящие в таблицу ASCII, кодируются в виде 1 байта, а другие символы могут занимать от 2 до 4 байт. Если значительную часть текста составляют латинские буквы и цифры, такой подход позволяет значительно уменьшить объем файла в сравнении с UTF-16. Текст, состоящий только из символов таблицы ASCII, кодируется точно так же, как и в кодировке ASCII. По данным поисковой системы *Google*[15] на начало 2010 года около 50% сайтов используют кодировку UTF-8.

Достоинства кодировки UNICODE состоят в том, что она позволяет использовать символы разных языков в одном документе и решает проблему правильного отображения текста, вызванную использованием разных кодовых страниц. За это приходится расплачиваться увеличением объема файлов.

## *Контрольные вопросы*

1. Какая информация хранится в текстовом файле?
2. Что такое шрифтовой файл?
3. Вы хотите использовать в тексте придуманный собственный символ, которого нет ни в одном шрифте. Какими путями это можно сделать?
4. Вы сами разработали шрифт и хотите переслать другу документ, в котором этот шрифт используется. Какими способами это можно сделать?
5. В чем недостатки и преимущества внедрения шрифтов в документ?
6. Что такое ASCII? Сколько символов содержится в этой кодировке?
7. Почему в современных компьютерах используются кодировки, в которых каждый символ занимает целое число байт?
8. Что такое кодовая страница?
9. Назовите основные кодовые станицы, содержащие русские буквы.
10. Почему использование кодовых страниц для кодирования текста может привести к проблемам?
11. Что делать, если вы видите непонятный набор символов на веб-странице?
12. В чем состоит ограничение 8-битных кодировок?
13. Что такое UNICODE? В чем достоинства и недостатки использования этого семейства кодировок?
14. Что такое UTF-16 и UTF-8? Чем отличаются эти кодировки?

## *Задачи*

1. Сколько символов можно закодировать с помощью 5 битного кода? 9-битного?
2. Сколько бит нужно выделить на символ для того, чтобы использовать в одном документе 100 разных символов? 200? 500?
3. Какой символ имеет код 100 в кодировке ASCII?
4. Какой код имеет цифра '5' в кодировке ASCII?
5. Определите, чему равен информационный объем следующего высказывания *Рене Декарта*, закодированного с помощью 16-битной кодировки UNICODE:

   **`Я мыслю, следовательно, существую.`**

   (Ответ: 68 байт = 544 бита)

[15] http://googleblog.blogspot.com/2010/01/unicode-nearing-50-of-web.html

6. При перекодировке сообщения на русском языке из 16-битного кода UNICODE в 8-битную кодировку KOI-8 оно уменьшилось на 480 бит. Какова длина сообщения в символах? (Ответ: 60)
7. При перекодировке сообщения из 8-битного кода в 16-битную кодировку UNICODE его объем увеличился на 2048 байт. Каков был информационный объем сообщения до перекодировки? (Ответ: 2048 байт)
8. В таблице ниже представлена часть кодовой таблицы ASCII:

| **Символ** | **1** | **5** | **A** | **B** | **Q** | **a** | **b** |
|---|---|---|---|---|---|---|---|
| Десятичный код | 49 | 53 | 65 | 66 | 81 | 97 | 98 |
| Шестнадцатеричный код | 31 | 35 | 41 | 42 | 51 | 61 | 62 |

Каков шестнадцатеричный код символа «$q$» ? (Ответ: $71_{16}$)

# 2.13 Кодирование графической информации

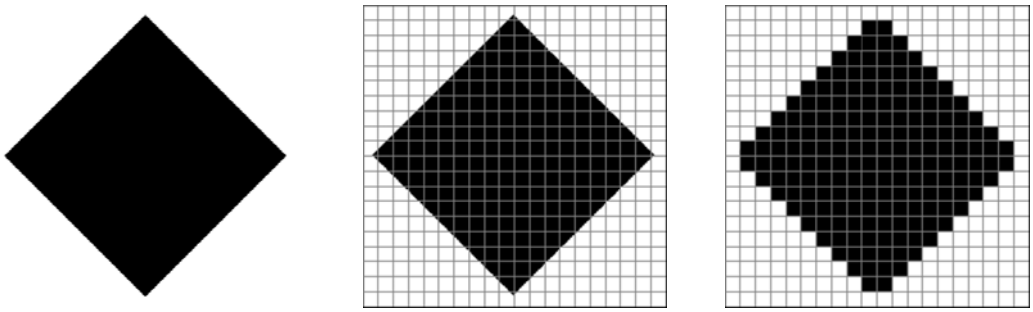
Как и все виды информации, изображения в компьютере закодированы в виде двоичных последовательностей. Используют два принципиально разных метода кодирования, каждый из которых имеет свои достоинства и недостатки.

## *2.13.1 Растровое кодирование*

Рисунок состоит из линий и закрашенных областей. В идеале нам нужно закодировать все особенности этого изображения так, чтобы его можно быть в точности восстановить из кода (например, распечатать на принтере).

И линия, и область состоят из бесконечного числа точек. Цвет каждой из этих точек нам нужно закодировать. Если их бесконечно много, мы сразу приходим к выводу, что для этого нужно бесконечно много памяти. Поэтому «поточечным» способом изображение закодировать не удастся. Однако, эту все-таки идею можно использовать.

Начнем с черно-белого рисунка. Представим себе, что на изображение ромба наложена сетка, которая разбивает его на квадратики. Такая сетка называется *растром*. Теперь для каждого квадратика определим цвет (черный или белый). Для тех квадратиков, в которых часть оказалась закрашена черным цветом, а часть белым, выберем цвет в зависимости от того, какая часть (черная или белая) больше.



У нас получился так называемый *растровый рисунок*, состоящий из квадратиков-пикселей.

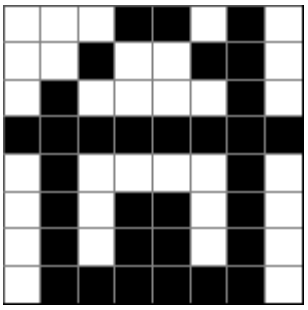
**Пиксель** (англ. *pixel* = *picture element*, элемент рисунка) – это наименьший элемент рисунка, для которого можно задать свой цвет.

Разбив «обычный» рисунок на квадратики, мы выполнили его *дискретизацию* – разбили единый объект на отдельные элементы. Действительно, у нас был единый и неделимый рисунок – изображение ромба. В результаты мы получили *дискретный* объект – набор пикселей.

Двоичный код для черно-белого рисунка, полученного в результате дискретизации можно построить следующим образом:

- заменяем белые пиксели нулями, а черные – единицами;
- выписываем строки полученной таблицы одну за другой.

Покажем это на простом примере:



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

Ширина этого рисунка – 8 пикселей, поэтому каждая строчка таблицы состоит из 8 двоичных разрядов – бит. Чтобы не писать очень длинную цепочку нулей и единиц, удобно использовать шестнадцатеричную систему счисления, закодировав 4 соседних бита (тетраду) одной шестнадцатеричной цифрой. Например, для первой строки получаем код $1A_{16}$:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| | | | 1 | | | A | |

а для всего рисунка: $1A2642FF425A5A7E_{16}$.

Очень важно понять, что мы приобрели и что потеряли в результате дискретизации. Самое важное – мы смогли закодировать рисунок в двоичном коде. Однако при этом рисунок исказился – вместо ромба мы получили набор квадратиков. Причина искажения в том, что в некоторых квадратиках части исходного рисунка были закрашены разными цветами, а в закодированном изображении каждый пиксель обязательно имеет один цвет. Таким образом, часть исходной информации при кодировании была потеряна. Это проявится, например, при увеличении рисунка – квадратики увеличиваются, и рисунок еще больше искажается. Чтобы уменьшить потери информации, нужно уменьшать размер пикселя, то есть увеличивать *разрешение*.

**Разрешение** – это количество пикселей, приходящихся на дюйм размера изображения.

Разрешение обычно измеряется в пикселях на дюйм (используется английское обозначение *ppi = pixels per inch*). Например, разрешение 254 ppi означает, что на дюйм (25,4 мм) приходится 254 пикселя, так что каждый пиксель «содержит» квадрат исходного изображения размером 0,1×0,1 мм. Если провести дискретизацию рисунка размером 10×15 см с разрешением 254 ppi, высота закодированного изображения будет 100/0,1 = 1000 пикселей, а ширина – 1500 пикселей.

Чем больше разрешение, тем точнее кодируется рисунок (меньше информации теряется), однако одновременно растет и объем файла.

## 2.13.2 Кодирование цвета

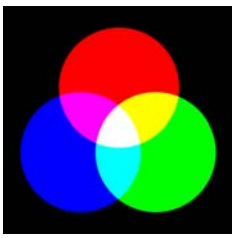
Что делать, если рисунок цветной? В этом случае для кодирования цвета пикселя уже не обойтись одним битом. Например, в показанном на рисунке изображении российского флага 4 цвета: черный, синий, красный и белый. Для кодирования одного из четырех вариантов нужно 2 бита, поэтому код каждого цвета (и код каждого пикселя) будет состоять из двух бит. Пусть 00 обозначает черный цвет, 01 – красный, 10 – синий и 11 – белый. Тогда получаем такую таблицу:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 00 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| 00 | 11 | 11 | 11 | 11 | 11 | 11 | 11 |
| 00 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 00 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 00 | 01 | 01 | 01 | 01 | 01 | 01 | 01 |
| 00 | 01 | 01 | 01 | 01 | 01 | 01 | 01 |

Проблема только в том, что при выводе на экран нужно как-то определить, какой цвет соответствует тому или другому коду. То есть информацию о цвете нужно выразить в виде числа (или набора чисел).

Человек воспринимает свет как множество электромагнитных волн. Определенная длина волны соответствуют некоторому цвету. Например, волны длиной 500-565 нм – это зеленый цвет. Так называемый «белый» свет на самом деле представляет собой смесь волн, длины которых охватывают весь видимый диапазон.

Согласно современному представлению о **цветном зрении** (теории Юнга-Гельмгольца[16]) глаз человека содержит чувствительные элементы трех типов. Каждый из них воспринимает весь поток света, но первые наиболее чувствительны в области красного цвета, вторые – области зеленого, а третьи – в области синего цвета. Цвет – это результат возбуждения всех трех типов рецепторов. Поэтому считается, что любой цвет (то есть ощущения человека, воспринимающего волны определенной длины) можно имитировать, используя только три световых луча (красный, зеленый и синий) разной яркости. Следовательно, любой цвет (в том числе и «белый») приближенно раскладывается на три составляющих – красную, зеленую и синюю. Меняя силу этих составляющих, можно составить любые цвета. Эта модель цвета получила название RGB по начальным буквам английских слов *red* (красный), *green* (зеленый) и *blue* (синий).



В **модели RBG** яркость каждой составляющей (или, как говорят, каждого канала) чаще всего кодируется целым числом от 0 до 255. При этом код цвета – это тройка чисел (R,G,B), яркости отдельных каналов. Цвет (0,0,0) – это черный цвет, а (255,255,255) – белый. Если все составляющие имеют равную яркость, получаются оттенки серого цвета, от черного до белого.

Чтобы сделать светло-красный (розовый) цвет, нужно в красном цвете (255,0,0) одинаково увеличить яркость зеленого и синего каналов, например, цвет (255, 150, 150) – это розовый. Равномерное уменьшение яркости всех каналов делает темный цвет, например, цвет с кодом (100,0,0) – тёмно-красный.

При кодировании цвета на веб-страницах также используется модель RGB, но яркости каналов записываются в шестнадцатеричной системе счисления (от $00_{16}$ до $FF_{16}$), а перед кодом цвета ставится знак #. Например, код красного цвета записывается как #FF0000, а код синего – как #0000FF. Вот коды некоторых цветов:

| Цвет | Код (R,G,B) | Код на веб-странице |
|---|---|---|
| Красный | (255,0,0) | #FF0000 |
| Зеленый | (0,255,0) | #00FF00 |
| Синий | (0,0,255) | #0000FF |
| Белый | (255,255,255) | #FFFFFF |
| Черный | (0,0,0) | #000000 |

[16] http://www.ozrenii.ru/trexkomponentnaya-teoriya-cvetovogo-zreniya-teoriya-yunga-gelmgolca/

| | | |
|---|---|---|
| Серый | (128,128,128) | #808080 |
| Фиолетовый | (255,0,255) | #FF00FF |
| Голубой | (0,255,255) | #00FFFF |
| Желтый | (255,255,0) | #FFFF00 |
| Тёмно-фиолетовый | (128,0,128) | #800080 |
| Светло-желтый | (255,255,128) | #FFFF80 |

Всего есть по 256 вариантов яркости каждого из трех цветов. Это позволяет закодировать $256^3$ = 16 777 216 оттенков, что более чем достаточно для человека. Так как 256 = $2^8$, каждая из трех составляющих занимает в памяти 8 бит или 1 байт, а вся информация о каком-то цвете – 24 бита (или 3 байта). Эта величина называется *глубиной цвета*.

**Глубина цвета** – это количество бит, используемое для кодирования цвета пикселя.

24-битное кодирование цвета часто называют режимом **истинного цвета** (англ. *True Color* – истинный цвет). Для вычисления объема рисунка в байтах при таком кодировании нужно определить общее количество пикселей (перемножить ширину и высоту) и умножить результат на 3, так как цвет каждого пикселя кодируется тремя байтами. Например, рисунок размером 20×30 пикселей, закодированный в режиме истинного цвета, будет занимать 20×30×3 = 1800 байт. Конечно, здесь не учитывается сжатие, которое применяется во всех современных форматах графических файлов. Кроме того, в реальных файлах есть *заголовок*, в котором записана служебная информация (например, размеры рисунка).

Кроме режима истинного цвета используется также 16-битное кодирование (англ. *High Color* – «высокий» цвет), когда на красную и синюю составляющую отводится по 5 бит, а на зеленую, к которой человеческий глаз более чувствителен – 6 бит. В режиме *High Color* можно закодировать $2^{16}$ = 65 536 различных цветов. В мобильных телефонах применяют 12-битное кодирование цвета (4 бита на канал, 4096 цветов).

Как правило, чем меньше цветов используется, тем больше будет искажаться цветное изображение. Таким образом, при кодировании цвета тоже есть неизбежная потеря информации, которая «добавляется» к потерям, вызванным дискретизацией. Однако при увеличении количества используемых цветов одновременно растет объем файла. Например, в режиме истинного цвета файл получится в два раза больше, чем при 12-битном кодировании.

Очень часто (например, в схемах, диаграммах и чертежах) количество цветов в изображении невелико (не более 256). В этом случае применяют **кодирование с палитрой**.

**Цветовая палитра** – это таблица, в которой каждому цвету, заданному в виде составляющих в модели RGB, сопоставляется числовой код.

Кодирование с палитрой выполняется следующим образом:

- выбираем количество цветов $N$ (как правило, не более 256);
- из палитры истинного цвета (16 777 216 цветов) выбираем любые $N$ цветов и для каждого из них находим составляющие в модели RGB;
- каждому из цветов присваиваем номер (код) от 0 до $N$–1;
- составляем палитру, записывая сначала RGB-составляющие цвета, имеющего код 0, затем – составляющие цвета с кодом 1 и т.д.
- цвет каждого пикселя кодируется не в виде значений RGB-составляющих, а как номер цвета в палитре.

Например, при кодировании изображения российского флага (см. выше) были выбраны 4 цвета:

- черный: RGB-код (0,0,0); двоичный код $00_2$;

- красный: RGB-код (255,0,0); двоичный код $01_2$;
- синий: RGB-код (0,0,255); двоичный код $10_2$;
- белый: RGB-код (255,255,255); двоичный код $11_2$;

Поэтому палитра, которая обычно записывается в специальную служебную область в начале файла (ее называют *заголовком файла*), представляет собой четыре трехбайтных блока:

| 0 | 0 | 0 | 255 | 0 | 0 | 0 | 0 | 255 | 255 | 255 | 255 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| цвет $00_2$ | | | цвет $01_2$ | | | цвет $10_2$ | | | цвет $11_2$ | | |

Код каждого пикселя занимает всего два бита.

Чтобы примерно оценить объем рисунка с палитрой, включающей $N$ цветов (без учета сжатия), нужно

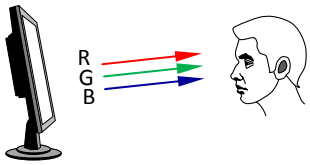
- определить размер палитры, $3\times N$ байт или $24\times N$ бит;
- определить *глубину цвета* (количество бит на пиксель), то есть найти наименьшее натуральное число $k$, такое что $2^k \ge N$;
- вычислить общее количество пикселей $M$, перемножив размеры рисунка;
- определить информационный объем основной части $M\times k$ бит.

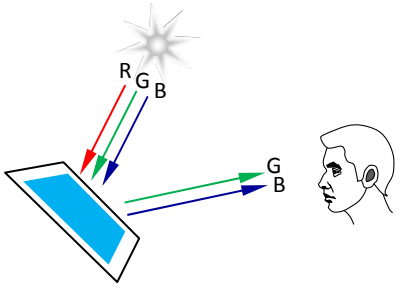
В таблице приведены данные по некоторым вариантам кодирования с палитрой:

| Количество цветов | Размер палитры (байт) | Глубина цвета (бит на пиксель) |
|---|---|---|
| 2 | 6 | 1 |
| 4 | 12 | 2 |
| 16 | 48 | 4 |
| 256 | 768 | 8 |

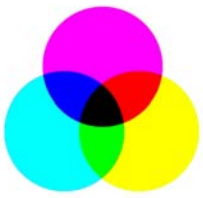
Палитры с количеством цветом более 256 на практике не используются.

RGB-кодирование лучше всего описывает цвет, который *излучается* некоторым устройством, например, монитором или экраном ноутбука. Когда же мы смотрим на изображение, отпечатанное на бумаге, ситуация совершенно другая. Мы видим не прямые лучи источника, попадающие в глаз, а *отраженные* от поверхности. «Белый свет» от какого-то источника (солнца, лампочки), содержащий волны во всем видимом диапазоне, попадает на бумагу, на которой нанесена краска. Краска поглощает часть лучей (их энергия уходит на нагрев), а оставшиеся попадают в глаз, это и есть тот цвет, который мы видим.




Например, если краска поглощает красные лучи, остаются только синие и зеленые – мы видим голубой цвет. В этом смысле красный и голубой цвета дополняют друг друга, так же, как и пары зеленый – фиолетовый и синий – желтый. Действительно, если из белого цвета (его RGB-код #FFFFFF) «вычесть» зеленый, то получится цвет #FF00FF (фиолетовый, пурпурный), а если «вычесть» синий, то получится цвет #FFFF00 (желтый).



На трех дополнительных цветах – голубом, фиолетовом и желтом – строится цветовая модель CMY (англ. ***C**yan* – голубой, ***M**agenta* – фиолетовый, ***Y**ellow* – желтый), которая применяется для вывода на печать. Значения C=M=Y=0 говорят о том,

что на белую бумагу не наносится никакая краска, поэтому все лучи отражаются, это белый цвет. Если добавить голубого цвета, красные лучи поглощаются, остаются только синие и зеленые. Если сверху нанести еще желтую краску, которая поглощает синие лучи, остается только зеленый (см. рисунок).

При наложении голубой, фиолетовой и желтой красок теоретически должен получиться черный цвет, все лучи поглощаются. Однако на практике все не так просто. Краски не идеальны, поэтому вместо черного цвета получается грязно-коричневый. Кроме того, при печати черных областей приходится «выливать» тройную порцию краски в одно место. Нужно также учитывать, что обычно на принтерах часто распечатывают черный текст, а цветные чернила значительно дороже черных.

Чтобы решить эту проблему, в набор красок добавляют черную, это так называемый *ключевой цвет* (англ. ***K**ey color*), поэтому получившуюся модель обозначают CMYK. Изображение, которое печатает большинство принтеров, состоит из точек этих четырех цветов, которые расположены в виде узора очень близко друг к другу. Это создает иллюзию того, что в рисунке есть разные цвета.

Кроме цветовых моделей RGB и CMY (CMYK), существуют и другие. Наиболее интересная из них – модель HSB (англ. ***H**ue* – тон, оттенок; ***S**aturation* – насыщенность, ***B**rightness* – яркость), которая ближе всего к естественному восприятию человека. Тон – это, например, синий, зеленый, желтый. Насыщенность – это чистота тона, при уменьшении насыщенности до нуля получается серый цвет. Яркость определяет, насколько цвет светлый или темный. Любой цвет при снижении яркости до нуля превращается в черный.

Строго говоря, цвет, кодируемый в моделях RGB, CMYK и HSV, зависит от устройства, на котором этот цвет будет изображаться. Для кодирования «абсолютного» цвета применяют модель Lab (англ. *Lightness* – светлота, $a$ и $b$ – параметры, определяющие тон и насыщенность цвета), которая является международным стандартом. Эта модель используется, например, для перевода цвета из RGB в CMYK и обратно.

Обычно изображения, предназначенные для печати, готовятся на компьютере (в режиме RGB), а потом переводятся в цветовую модель CMYK. При этом стоит задача получить при печати такой же цвет, что и на мониторе. И вот тут возникают проблемы. Дело в том, что при выводе пикселей на экран монитор получает некоторые числа (RGB-коды), на основании которых нужно «выкрасить» пиксели тем или иным цветом. Отсюда следует важный вывод.

> Цвет, который мы видим на мониторе, зависит от характеристик и настроек монитора.

Это значит, что, например, красный цвет (R=255, G=B=0) на разных мониторах будет разным. Наверняка вы видели этот эффект в магазине где продают телевизоры и мониторы – одна и та же картинка на каждом из них выглядит по-разному. Что же делать?

Во-первых, выполняется калибровка монитора – настройка яркости, контрастности, белого, черного и серого цветов. Во-вторых, профессионалы, работающие с цветными изображениями, используют *цветовые профили* мониторов, сканеров, принтеров и других устройств. В профилях хранится информация о том, каким реальным цветам соответствуют различные RGB-коды или CMYK-коды. Для создания профиля используют специальные приборы – *калибраторы* (*колориметры*), которые «измеряют» цвет с помощью трех датчиков[17], принимающих лучи в красном, зеленом и синем диапазонах. Современные форматы графических файлов (например,

[17] http://www.ukr-print.net/article/60.htm

формат **.PSD** программы *Adobe Photoshop*) вместе с кодами пикселей содержат и профиль монитора, на котором создавался рисунок.

Для того, чтобы результат печати на принтере был максимально похож на изображение на мониторе, нужно (используя профиль монитора) определить «абсолютный» цвет (например, в модели *Lab*), который видел пользователь, а потом (используя профиль принтера) найти CMYK-код, который даст при печати наиболее близкий цвет.

Проблема состоит в том, что не все цвета RGB-модели могут быть напечатаны. В первую очередь это относится к ярким и насыщенным цветам. Например, ярко-красный цвет (R=255, G=B=0) нельзя напечатать, ближайший к нему цвет в модели CMYK (C=0, M=Y=255, K=0) при обратном переводе в RGB может дать значения[18] в районе R=237, G=28, B=26. Поэтому при преобразовании ярких цветов в модель CMYK (и при печати ярких рисунков) они становятся тусклее. Это обязательно должны учитывать профессиональные дизайнеры.

### *2.13.3 Растровое кодирование: итоги*

Итак, при растровом кодировании рисунок разбивается на пиксели (дискретизируется). Для каждого пикселя определяется единый цвет, который чаще всего кодируется с помощью RGB-кода. На практике эти операции выполняет *сканер* (устройство для ввода изображений) и цифровой фотоаппарат.

Растровое кодирование имеет **достоинства**:

- универсальный метод (можно закодировать любое изображение);
- единственный метод для кодирования и обработки размытых изображений, не имеющих четких границ, например, фотографий;

и **недостатки**:

- при дискретизации всегда есть *потеря информации*;
- при *изменении размеров* изображения искажается цвет и форма объектов на рисунке, поскольку при увеличении размеров надо как-то восстановить недостающие пиксели, а при уменьшении – заменить несколько пикселей одним;
- *размер файла* не зависит от сложности изображения, а определяется только разрешением и глубиной цвета; как правило, растровые рисунки имеют большой объем.

Существует много разных форматов растровых рисунков. Чаще всего встречаются следующие:

- **BMP** (англ. *bitmap* – битовая карта, файлы с расширением **.bmp**) – стандартный формат в операционной системе *Windows*; поддерживает кодирование с палитрой и в режиме истинного цвета;
- **JPEG** (англ. *Joint Photographic Experts Group* – объединенная группа фотографов-экспертов, файлы с расширением **.jpg** или **.jpeg**) – формат, разработанный специально для кодирования фотографий; поддерживает только режим истинного цвета; для уменьшения объема файла используется сильное сжатие, при котором изображение немного искажается, поэтому не рекомендуется использовать его для рисунков с четкими границами;
- **GIF** (англ. *Graphics Interchange Format* – формат для обмена изображениями, файлы с расширением **.gif**) – формат, поддерживающий только кодирование с палитрой (от 2 до 256 цветов); в отличие от предыдущих форматов, части рисунка могут быть прозрачными, то есть на веб-странице через них будет «просвечивать» фон; в

[18] Как вы понимаете, точные цифры зависят от профилей монитора и принтера.

современном варианте формата GIF можно хранить анимированные изображения; используется сжатие без потерь, то есть при сжатии изображение не искажается;

- **PNG** (англ. *Portable Network Graphics* – переносимые сетевые изображения, файлы с расширением **.png**) – формат, поддерживающий как режим истинного цвета, так и кодирование с палитрой; части изображения могут быть прозрачными и даже полупрозрачными (32-битное кодирование RGBA, где четвертый байт задает прозрачность); изображение сжимается без искажения; анимация не поддерживается.

Свойства рассмотренных форматов сведены в таблицу:

| Формат | Истинный цвет | С палитрой | Прозрачность | Анимация |
|---|---|---|---|---|
| **BMP** | да | да | – | – |
| **JPEG** | да | – | – | – |
| **GIF** | – | да | да | да |
| **PNG** | да | да | да | – |

Вы уже знаете, что все виды информации хранятся в памяти компьютера в виде двоичных кодов, то есть цепочек из нулей и единиц. Получив такую цепочку, абсолютно невозможно сказать, что это – текст, рисунок, звук или видео. Например, код $11001000_2$ может обозначать число 200, букву 'И', одну из составляющих цвета пикселя в режиме истинного цвета, номер цвета в палитре для рисунка с палитрой 256 цветов, цвета 8 пикселей черно-белого рисунка и т.п. Как же компьютер разбирается в двоичных данных? В первую очередь нужно ориентироваться на расширение имени файла. Например, чаще всего файлы с расширением **.txt** содержат текст, а файлы с расширениями **.bmp**, **.gif**, **.jpg**, **.png** – рисунки.

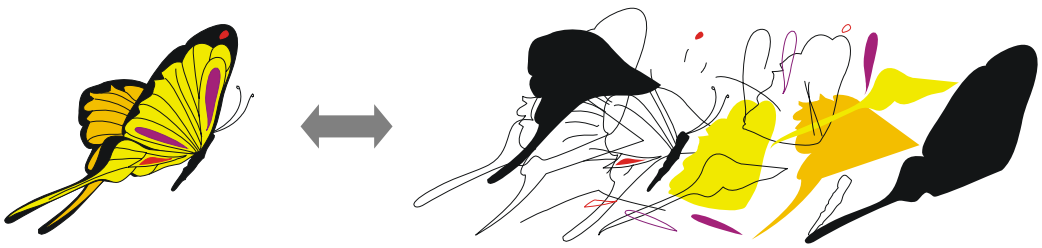
Однако расширение файла можно менять как угодно. Например, можно сделать так, что текстовый файл будет иметь расширение **.bmp**, а рисунок в формате JPEG – расширение **.txt**. Поэтому в начало всех файлов специальных форматов (кроме простого текста, **.txt**) записывается *заголовок*, по которому можно «узнать» тип файла и его характеристики. Например, файлы в формате BMP начинаются с символов «BM», а файлы в формате GIF – с символов «GIF». Кроме того, в заголовке указывается размер рисунка и его характеристики, например, количество цветов в палитре, способ сжатия и т.п. Используя эту информацию, программа «расшифровывает» основную часть файла и выводит его на экран.

## *2.13.4 Векторное кодирование*

Для чертежей, схем, карт применяется другой способ кодирования, который позволяет не терять качество при изменении размеров изображения. Рисунок хранится как набор простейших геометрических фигур (*графических примитивов*): линий, многоугольников, сглаженных кривых, окружностей, эллипсов. Такой рисунок называется *векторным*.

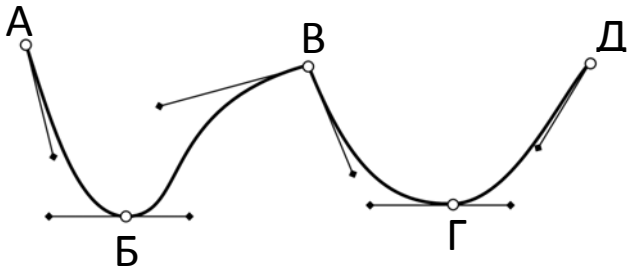
**Векторный рисунок** – это рисунок, который закодирован в виде набора простейших геометрических фигур, параметры которых (размеры, координаты вершин, углы наклона, цвет контура и заливки) хранятся в виде чисел.

Векторный рисунок можно «разобрать» на части, растащив мышкой его элементы, а потом снова собрать полное изображение:



Как вы понимаете, сделать что-то подобное с растровым рисунком не удастся.

При векторном кодировании для отрезка хранятся координаты его концов, для прямоугольников и ломаных – координаты вершин. Окружность и эллипс можно задать координатами прямоугольника, в который вписана фигура. Сложнее обстоит дело со сглаженными кривыми. На рисунке изображена линия с опорными точками А, Б, В, Г и Д.




У каждой из этих точек есть «рукоятки» (*управляющие линии*), перемещая концы этих рукояток можно регулировать наклон касательной и кривизну всех участков кривой. Если обе рукоятки находятся на одной прямой, получается сглаженный узел, если нет – то угловой узел. Таким образом, форма этой кривой полностью задается координатами опорных точек и координатами рукояток. Кривые, заданные таким образом, называют *кривыми Безье* в честь их изобретателя французского инженера Пьера Безье.

Векторный способ кодирования рисунки обладает значительными **преимуществами** в сравнении с растровым тогда, когда изображение может быть полностью разложено на простейшие геометрические фигуры (например, чертеж, схема, карта, диаграмма). В этом случае при кодировании нет потери информации.

Объем файлов напрямую зависит от сложности рисунка – чем меньше элементов, тем меньше места занимает файл. Как правило, векторные рисунки значительно меньше по объему, чем растровые.

При изменении размера векторного рисунка не происходит никакого искажения формы элементов, при увеличении наклонных линий не появляются «ступеньки», как при растровом кодировании:


растровый рисунок


векторный рисунок

Самый главный **недостаток** этого метода – он практически непригоден для кодирования размытых изображений, например, фотографий.

Среди форматов векторных рисунков отметим следующие:

- **WMF** (англ. *Windows Metafile* – метафайл *Windows*, файлы с расширением **`.wmf`** и **`.emf`**) – стандартный формат векторных рисунков в операционной системе *Windows*;

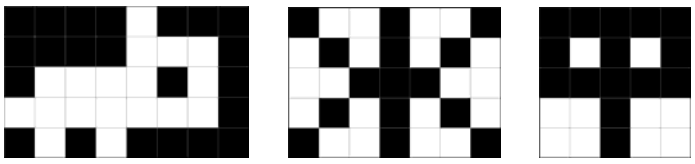
- **CDR** (файлы с расширением **`.cdr`**) – формат векторных рисунков программы *CorelDRAW*;
- **AI** (файлы с расширением **`.ai`**) – формат векторных рисунков программы *Adobe Illustrator*;
- **SVG** (англ. *Scalable Vector Graphics* – масштабируемые векторные изображения, файлы с расширением **`.svg`**) – векторная графика для веб-страниц.

## *Контрольные вопросы*

1. Какие два принципа кодирования рисунков используются в компьютерной технике?
2. Почему не удается придумать единый метод кодирования рисунков, пригодный во всех ситуациях?
3. В чем состоит идея растрового кодирования? Что такое растр?
4. Что такое пиксель?
5. Что такое дискретизация? Почему она необходима?
6. Что теряется при дискретизации? Почему?
7. Как уменьшить потерю информации при дискретизации? Что при этом ухудшается?
8. Что такое разрешение? В чем оно измеряется?
9. Что такое режим истинного цвета?
10. Что такое кодирование с палитрой? В чем его принципиальное отличие от режима истинного цвета?
11. Какие устройства используются для ввода изображений в компьютер?
12. В чем состоят достоинства и недостатки растрового кодирования?
13. Чем отличаются четыре основных современных формата кодирования рисунков?
14. Какие форматы поддерживают рисунки с прозрачными и полупрозрачными областями?
15. В каких форматах целесообразно сохранять фотографии? Рисунки с четкими границами?
16. Как можно уменьшить объем файла, в котором хранится рисунок? Чем при этом придется пожертвовать?
17. Как компьютер определяет, что находится в файле – текст, рисунок, звук или видео?
18. Что такое векторное кодирование? В чем его отличие от растрового, преимущества и недостатки?
19. В каких задачах используют векторное кодирование?
20. Что такое кривые Безье?
21. Почему при увеличении растрового рисунка появляются «ступеньки»?
22. Какие форматы векторных рисунков вы знаете?

## *Задачи*

1. Постройте двоичные коды для черно-белых рисунков и запишите их в шестнадцатеричной системе счисления:



   Какие сложности у вас возникли? Как их можно преодолеть?
2. Постройте черно-белый рисунок шириной 8 пикселей, закодированный шестнадцатеричной последовательностью $2466FF6624_{16}$.
3. Постройте черно-белый рисунок шириной 5 пикселей, закодированный шестнадцатеричной последовательностью $3A53F88_{16}$.
4. Рисунок размером 10×15 см кодируется с разрешением 300 ppi. Оцените количество пикселей в этом рисунке. (Ответ: около 2 мегапикселей)
5. Постройте шестнадцатеричный код для цветов, имеющих RGB-коды (100,200,200), (30,50,200), (60,180, 20), (220, 150, 30). (Ответ: #64C8C8, #1E32C8, #3CB414, #DC961E)

6. Как бы вы назвали цвет, заданный на веб-странице в виде кода: #CCCCCC, #FFCCCC, #CCCCFF, #000066, #FF66FF, #CCFFFF, #992299, #999900, #99FF99? Найдите десятичные значения составляющих RGB-кода. (Ответ: (204,204,204), (255,204,204), (204,204,255), (0,0,102), (255.255,102), (104,255,255), (153,34,153), (153,153,0), (153,255,153))
7. Что такое глубина цвета? Как связаны глубина цвета и объем файла?
8. Какова глубина цвета, если в рисунке используется 65536 цветов? 256 цветов? 16 цветов? (Ответ: 16 бит; 8 бит; 4 бита)
9. Для желтого цвета найдите красную, зеленую и синюю составляющие при 12-битном кодировании. (Ответ: R=G=15, B=0)
10. Сколько места занимает палитра в файле, где используются 64 цвета? 128 цветов?
11. Сколько байт будет занимать код рисунка размером 40×50 пикселей в режиме истинного цвета? при кодировании с палитрой 256 цветов? при кодировании с палитрой 16 цветов? в черно-белом варианте (два цвета)? (Ответ: 6000, 2000, 1000, 250)
12. Сколько байт будет занимать код рисунка размером 80×100 пикселей в кодировании с глубиной цвета 12 бит на пиксель? (Ответ: 12000)
13. Для хранения растрового изображения размером 32×32 пикселя отвели 512 байтов памяти. Каково максимально возможное число цветов в палитре изображения? (Ответ: 16)
14. Для хранения растрового изображения размером 128 x 128 пикселей отвели 4 килобайта памяти. Каково максимально возможное число цветов в палитре изображения? (Ответ: 4)
15. В процессе преобразования растрового графического файла количество цветов уменьшилось с 1024 до 32. Во сколько раз уменьшился информационный объем файла? (Ответ: в 2 раза)
16. В процессе преобразования растрового графического файла количество цветов уменьшилось с 512 до 8. Во сколько раз уменьшился информационный объем файла? (Ответ: в 3 раза)
17. Разрешение экрана монитора – 1024 x 768 точек, глубина цвета – 16 бит. Каков необходимый объем видеопамяти для данного графического режима? (Ответ: 1,5 Мбайт)
18. После преобразования растрового 256-цветного графического файла в черно-белый формат (2 цвета) его размер уменьшился на 70 байт. Каков был размер исходного файла? (Ответ: 80 байт)
19. Сколько памяти нужно для хранения 64-цветного растрового графического изображения размером 32 на 128 точек? (Ответ: 3 Кбайта)
20. Какова ширина (в пикселях) прямоугольного 64-цветного неупакованного растрового изображения, занимающего на диске 1,5 Мбайт, если его высота вдвое меньше ширины? (Ответ: 2048)
21. Какова ширина (в пикселях) прямоугольного 16-цветного неупакованного растрового изображения, занимающего на диске 1 Мбайт, если его высота вдвое больше ширины? (Ответ: 1024)

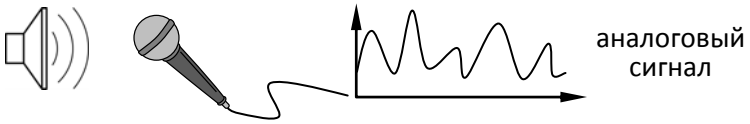
# 2.14 Кодирование звуковой информации

## *2.14.1 Оцифровка звука*

Звук – это колебания среды (воздуха, воды), которые воспринимает человеческое ухо. С помощью микрофона звук преобразуется в так называемый *аналоговый* электрический сигнал.

**Аналоговый сигнал** – это произвольное изменение некоторой величины в заданном диапазоне.

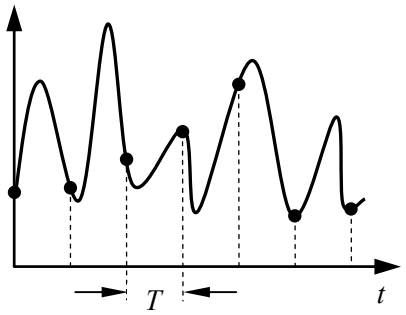
В любой момент времени сигнал на выходе микрофона (ток или напряжение) может принимать любое значение в некотором интервале.




В 1960-х годах были широко распространены *аналоговые компьютеры*, которые работали с аналоговыми сигналами (складывали, умножали и т.п.), однако их точность была невысока и они были постепенно вытеснены цифровыми компьютерами, которые не могут обрабатывать аналоговые сигналы. Таким образом, для хранения и обработки звука с помощью современных компьютеров нужно преобразовать аналоговый сигнал, полученный с микрофона, в двоичный код, то есть в цепочку нулей и единиц. Эта процедура называется *оцифровка.*

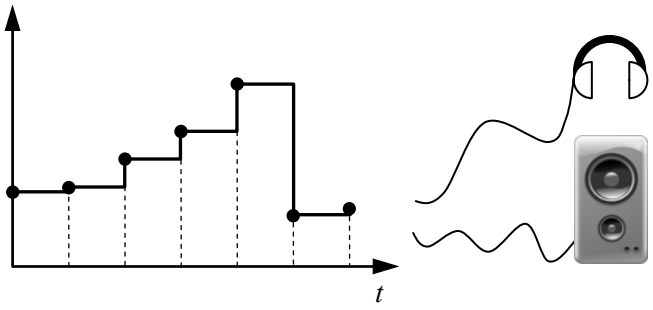
**Оцифровка** – это преобразование аналогового сигнала в цифровой код.

Ситуация напоминает ту, с которой мы столкнулись при кодировании рисунка – любая линия состоит из бесконечного числа точек, поэтому чтобы закодировать «по точкам» нужна бесконечная память. Здесь тоже придется использовать *дискретизацию* – представить аналоговый сигнал в виде набора чисел, то есть записать в память только значения сигнала в отдельных точках, взятых с некоторым шагом $T$ по времени.




Число $T$ называется *интервалом дискретизации*, а обратная ему величина $1/T$ – *частотой дискретизации*. Частота дискретизации обозначается буквой $f$ и измеряется в герцах (Гц) и килогерцах (кГц). Один герц – это 1 раз в секунду, а 1 кГц – 1000 раз в секунду. Чем больше частота дискретизации, тем точнее мы записываем сигнал, тем меньше информации теряем. Однако при этом возрастает количество отсчетов, то есть информационный объем закодированного звука.

Для кодирования звука в компьютерах чаще всего используются частоты дискретизации 8 кГц (плохое качество, но достаточно для распознавания речи), 11 кГц, 22 кГц, 44,1 кГц (звуковые компакт-диски), 48 кГц (фильмы в формате DVD), а также 96 кГц и 192 кГц (высококачественный звук в формате DVD-audio). Выбранная частота влияет на качество цифрового звука. Дело в том, что наушники и звуковые колонки – это аналоговые (не цифровые) устройства, и при проигрывании звука через звуковую карту компьютеру нужно как-то восстановить исходный аналоговый сигнал и передать его на наушники или звуковые колонки. В памяти есть только отсчеты, снятые с интервалом $T$, остальная информация была потеряна при кодировании. В простейшем случае по ним можно восстановить ступенчатый сигнал, который будет существенно отличаться от исходного (до кодирования).

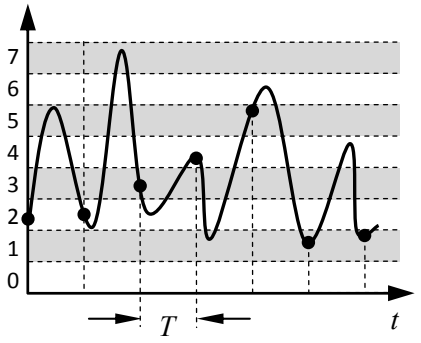



Для повышения качества звука, то есть для большего соответствия между сигналом, принятым микрофоном, и сигналом, выведенным из компьютера на колонки, нужно увеличивать частоту дискретизации, однако при этом, как вы уже знаете, увеличивается и объем файла. Как же выбрать оптимальную частоту при кодировании? Ответ на этот вопрос во многом это зависит от свойств звука, который нужно закодировать.

С точки зрения математики любой сигнал можно представить в виде суммы очень большого числа колебаний разных частот (гармоник). Если выбрать частоту дискретизации больше, чем удвоенная частота самой быстрой гармоники, то теоретически по отдельным отсчетам можно точно восстановить исходный аналоговый сигнал. Это результат известен в радиотехнике как теорема Котельникова-Шеннона.

К сожалению, на практике все несколько сложнее. Дело в том, что в реальных сигналах содержатся гармоники с очень высокими частотами, так что частота дискретизации, полученная с помощью теоремы Котельникова-Шеннона, будет также высока и объем файла недопустимо велик. Однако средний человек слышит только звуки с частотами от 16 Гц до 20 кГц, поэтому все частоты выше 20 кГц можно «потерять» практически без ухудшения качества звука (человек не почувствует разницу!). Удвоив эту частоту (по теореме Котельникова-Шеннона) получаем оптимальную частоту дискретизации около 40 кГц, которая обеспечивает наилучшее качество, различимое на слух. Поэтому при высококачественном цифровом кодировании звука на компакт-дисках и в видеофильмах чаще всего используют частоты 44,1 кГц и 48 кГц. Более низкие частоты применяют тогда, когда важно всячески уменьшать объем данных (например, для трансляции радиопередач через Интернет), даже ценой ухудшения качества.

Кроме того, что при кодировании звука выполняется дискретизация с потерей информации, нужно учитывать, что на хранение одного отсчета в памяти отводится ограниченное место. При этом вносятся дополнительные ошибки.




Представим себе, что на один отсчет выделяется 3 бита. При этом код каждого отсчета – это целое число от 0 до 7. Весь диапазон возможных значений сигнала, от 0 до максимально допустимого, делится на 8 полос, каждой из которых присваивается номер (код). Все отсчеты, попавшие в одну полосу, имеют одинаковый код.

Преобразование измеренного значения сигнала в число называется *дискретизацией по* уровню. Эту операцию выполняет аналого-цифровой преобразователь (АЦП) звуковой карты.

**Разрядность кодирования** – это число бит, используемое для хранения одного отсчета.

Недорогие звуковые карты имеют разрядность 16-18 бит, большинство современных – 24 бита, что позволяет использовать $2^{24}$ = 16 777 216 различных уровней.

Объем информации, полученный после оцифровки звука, зависит от разрядности и частоты дискретизации. Например, если используется 16-разрядное кодирование с частотой 44 кГц, за 1 с выполняется 44000 измерений сигнала, и каждое из измеренных значений занимает 16 бит (2 байта). Поэтому за 1 секунду накапливается 44000 × 2 = 88000 байт информации, а за 1 минуту

$$88000 \times 60 = 5\,280\,000 \text{ байт} \approx 5 \text{ Мбайт}.$$

Если записывается стерео звук (левый и правый каналы), это число нужно удвоить.

С помощью оцифровки можно закодировать любой звук, который принимает микрофон. В частности, это единственный способ кодирования человеческого голоса и различных природных звуков (шум прибоя и т.п.).

Однако у этого метода есть и **недостатки**:

- при оцифровке звука всегда есть потеря информации (из-за дискретизации);
- звуковые файлы имеют, как правило, большой размер, поэтому в большинстве современных форматов используется сжатие.

Среди форматов звуковых файлов наиболее известны

- **WAV** (англ. *Waveform Audio File Format*, файлы с расширением **`.wav`**) – стандартный формат звуковых файлов в операционной системе *Windows*; сжатие данных возможно, но используется редко;
- **MP3** (файлы с расширением **`.mp3`**) – самый популярный формат звуковых файлов, использующий сжатие с потерями: для значительного уменьшения объема файла снижается качество кодирования для тех частот, которые практически неразличимы для человеческого слуха;
- **WMA** (англ. *Windows Media Audio*, файлы с расширением **`.wma`**) – формат звуковых файлов, разработанный фирмой *Microsoft*; чаще всего используется сжатие для уменьшения объема файла;
- **Ogg Vorbis** (файлы с расширением **`.ogg`**) – свободный (не требующий коммерческих лицензий) формат сжатия звука с потерями.

Все эти форматы являются *потоковыми*, то есть можно начинать прослушивание до того момента, как весь файл будет получен (например, из Интернета).

## *2.14.2 Инструментальное кодирование*

Существует еще один, принципиально иной способ кодирования звука, который можно применить только для кодирования инструментальных мелодий. Он основан на стандарте MIDI (англ. *Musical Instrument Digital Interface* — цифровой интерфейс музыкальных инструментов). В отличие от оцифрованного звука, в таком формате хранятся последовательность нот, коды инструментов (можно использовать 128 мелодических и 47 ударных инструментов), громкость, тембр, время затухания каждой ноты и т.д. Фактически это программа, предназначенная для проигрывания звуковой картой, в памяти которой хранятся образцы звуков реальных инструментов (волновые таблицы, англ. *wave tables*).

Современные звуковые карты поддерживают многоканальный звук, то есть в звуковом файле может храниться несколько «дорожек», которые проигрываются одновременно. Таким образом, получается *полифония* – многоголосие, возможность проигрывать одновременно несколько нот. Количество голосов для современных звуковых карт может достигать 1024.

Звук, закодированный с помощью стандарта MIDI, хранится в файлах с расширением **.mid**. Существуют специальные клавиатуры, которые позволяют вводить звук и сразу сохранять его в формате **.mid**. Для проигрывания MIDI-файла используют *синтезаторы* – электронные устройства, имитирующие звук реальных инструментов. Простейшим синтезатором является звуковая карта компьютера.

Главные **достоинства** инструментального кодирования:

- кодирование мелодии (нотной записи) происходит без потери информации;
- файлы имеют значительно меньший объем в сравнении с оцифрованным звуком той же длительности.

Однако произвольный звук (например, человеческий голос) в таком формате закодировать невозможно. Кроме того, производители сами выбирают образцы звуков (так называемые *сэмплы*, от англ. *samples* – образцы), которые записываются в память звуковой карты (нет единого стандарта). Поэтому звучание MIDI-файла может немного отличаться на разной аппаратуре.

## ? *Контрольные вопросы*

1. Что такое аналоговый сигнал?
2. Какие вы знаете аналоговые приборы?
3. Почему аналоговые компьютеры были вытеснены цифровыми?
4. Что такое оцифровка? Если ли потеря информации при оцифровке? Почему?
5. Что такое интервал дискретизации и частота дискретизации?
6. Как связаны частота дискретизации с потерей информации и объемом файла?
7. Какие частоты дискретизации сейчас используются?
8. От чего зависит выбор частоты дискретизации?
9. Почему частоты дискретизации более 48 кГц применяются очень редко?
10. Как происходит вывод закодированного звукового сигнала на колонки или наушники?
11. Что такое дискретизация по уровню?
12. Какие устройство выполняет дискретизацию при записи звука?
13. Что такое разрядность кодирования звука? На что она влияет?
14. В чем достоинства и недостатки оцифровки?
15. Какие форматы файлов для хранения оцифрованного звука вы знаете?
16. Что такое потоковый звук?
17. Что такое инструментальное кодирование?
18. Что такое волновая таблица?
19. Что такое многоканальный звук?
20. В каких файлах хранится звук, закодированный с помощью стандарта MIDI?
21. Что такое синтезаторы?
22. В чем достоинства и недостатки инструментального кодирования звука?
23. Почему MIDI-файлы могут звучать по-разному на разной аппаратуре?

### *Задачи*

1. Заполните пустые клетки таблицы, вычислив объем звукового файла (без сжатия):

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Частота дискретизации, кГц | 8 | 8 | 11 | 11 | 22 | 22 | 44,1 | 44,1 | 48 | 48 |
| Глубина кодирования, бит | 8 | 8 | 16 | 16 | 16 | 8 | 24 | 8 | 8 | 24 |
| Моно/стерео | моно | стерео | моно | стерео | моно | стерео | моно | стерео | стерео | стерео |
| Время звучания, с | 16 | 8 | 64 | 32 | 32 | 32 | 256 | 128 | 4 | 4 |
| Объем файла, Кбайт | | | | | | | | | | |

2. Заполните пустые клетки таблицы, вычислив время звучания записи (объем файла приведен без учета сжатия):

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Частота дискретизации, кГц | 8 | 8 | 11 | 11 | 22 | 22 | 44,1 | 44,1 | 48 | 48 |
| Глубина кодирования, бит | 16 | 24 | 8 | 8 | 8 | 24 | 16 | 24 | 16 | 16 |
| Моно/стерео | моно | стерео | моно | стерео | моно | стерео | моно | стерео | моно | стерео |
| Объем файла, Кбайт | 125 | 375 | 1375 | 1375 | 6875 | 4125 | 11025 | 33075 | 375 | 1875 |
| Время звучания, с | | | | | | | | | | |

3. Заполните пустые клетки таблицы, вычислив глубину кодирования звука (объем файла приведен без учета сжатия):

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Частота дискретизации, кГц | 8 | 8 | 11 | 11 | 22 | 22 | 44,1 | 44,1 | 48 | 48 |
| Моно/стерео | моно | стерео | моно | стерео | моно | стерео | моно | стерео | моно | моно |
| Время звучания, с | 16 | 4 | 128 | 64 | 320 | 16 | 256 | 64 | 40 | 80 |
| Объем файла, Кбайт | 375 | 125 | 4125 | 4125 | 20625 | 1375 | 11025 | 11025 | 1875 | 11250 |
| Глубина кодирования, бит | | | | | | | | | | |

4. Заполните пустые клетки таблицы, вычислив частоту дискретизации звука (объем файла приведен без учета сжатия):

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Глубина кодирования, бит | 16 | 8 | 16 | 24 | 16 | 16 | 8 | 24 | 16 | 24 |
| Моно/стерео | моно | стерео | стерео | стерео | стерео | моно | моно | стерео | моно | стерео |
| Время звучания, с | 64 | 4 | 64 | 64 | 16 | 128 | 320 | 8 | 4 | 4 |
| Объем файла, Кбайт | 1375 | 375 | 11025 | 4125 | 1375 | 11025 | 6875 | 375 | 375 | 1125 |
| Частота дискретизации, кГц | | | | | | | | | | |

## 2.15 Кодирование видеоинформации

Для того чтобы сохранить видео в памяти компьютера, нужно закодировать звук и изменяющееся изображение, причем требуется обеспечить их *синхронность* (одновременность). Для кодирования звука чаще всего используют оцифровку с частотой 48 кГц. Изображение состоит из отдельных растровых рисунков, которые меняются с частотой не менее 25 кадров в секунду, так что глаз человека воспринимает смену кадров как непрерывное движение. Это значит, что для каждой секунды видео нужно хранить в памяти 25 изображений.

Если используется размер 768 на 576 точек (стандарты PAL/SECAM) и глубина цвета 24 бита на пиксель, то закодированная 1 секунда видео будет занимать примерно 32 Мбайта, а 1 минута – около 1,85 Гбайт. Это недопустимо много, поэтому в большинстве форматов видеоизображений используется сжатие с потерями. Это значит, что некоторые незначительные детали теряются, но «обычный» человек (непрофессионал) не почувствует существенного ухудшения качества. Основная идея такого сжатия заключается в том, что за короткое время изображение изменяется

очень мало, поэтому можно запомнить «исходный» кадр, а затем сохранять только изменения. Через 10-15 с изображение изменяется настолько, что необходим новый исходный кадр.

Наиболее известны следующие видеоформаты:

- **AVI** (англ. *Audio Video Interleave* – чередующиеся звук и видео, файлы с расширением **.avi**) – формат видеофайлов, разработанных фирмой *Microsoft* для системы *Windows*; может использовать разные алгоритмы сжатия;
- **WMV** (англ. *Windows Media Video*, файлы с расширением **.wmv**) – система кодирования видео, разработанная фирмой *Microsoft*; может использовать разные алгоритмы сжатия;
- **MPEG** (файлы с расширением **.mpg**, **.mpeg**) – формат звуковых файлов, использующий один из лучших алгоритмов сжатия, который разработала экспертная группа по вопросам движущегося изображения (MPEG = *Motion Picture Experts Group*);
- **MOV** (англ. *Quick Time Movie*, файлы с расширением **.mov**) – формат видеофайлов, разработанный фирмой *Apple*.

### ? *Контрольные вопросы*

1. Что такое синхронность?
2. Какая частота дискретизации звука чаще всего используется при кодировании видеофильмов?
3. Почему при кодировании видео используется частота не менее 25 кадров в секунду?
4. Почему компьютерные фильмы чаще всего хранятся в сжатом виде?
5. Что значит «сжатие с потерями»? В чем состоит его основная идея при кодировании видео?
6. Какие форматы видео вы знаете?

### *Задачи*

1. Кадры видеозаписи закодированы в режиме истинного цвета (24 бита на пиксель) и сменяются с частотой 25 кадров в секунду, запись содержит стереофонический звук. Остальные параметры для разных вариантов заданы в таблице. Оцените объем 1 минуты видеозаписи в Мбайтах (с точностью до десятых). Сколько минут такой записи поместится на стандартный CD-диск объемом 700 Мбайт?

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Ширина кадра, пиксели | 320 | 320 | 640 | 640 | 720 | 720 | 720 | 720 | 1920 | 1920 |
| Высота кадра, пиксели | 240 | 240 | 480 | 480 | 480 | 480 | 576 | 576 | 1080 | 1080 |
| Частота дискретизации, кГц | 11 | 48 | 48 | 48 | 22 | 48 | 22 | 48 | 48 | 48 |
| Глубина кодирования звука, бит | 24 | 16 | 24 | 16 | 16 | 16 | 24 | 24 | 16 | 24 |
| Степень сжатия | 10 | 8 | 6 | 4 | 10 | 12 | 8 | 6 | 8 | 10 |
| Объем файла, Мбайт | | | | | | | | | | |
| Поместится на CD-диск, минут | | | | | | | | | | |

(Ответ: 1,4; 1,7; 9,2; 13,8; 6,2; 5,2; 9,3; 12,4; 46,4; 37,1;
507, 402, 76, 51, 113, 136, 75, 56, 15, 19)

# 2.16 Передача информации

## *2.16.1 Скорость передачи информации*

**Скорость передачи информации** – это количество бит (байт, Кбайт и т.д.), которое передается по каналу связи за единицу времени (например, за 1 с).

Для любого канала связи, в котором есть помехи, пропускная способность ограничена. Это значит, что есть некоторая наибольшая возможная скорость передачи данных, которую принципиально невозможно превысить.

Основная единица измерения скорости – бит в секунду (бит/с, англ. *bps* = *bits per second*). Для характеристики быстродействующих каналов применяют килобиты в секунду (Кбит/с) и мегабиты в секунду (Мбит/с), иногда используют байты в секунду (байт/с) и килобайты в секунду (Кбайт/с).

Объем информации $I$, переданной по каналу за время $t$, вычисляется по формуле $I = v \cdot t$, где $v$ – скорость передачи информации. Например, если скорость передачи данных равна 512 000 бит/с, за 1 минуту можно передать файл объемом

512 000 бит/с × 60 с = 30 720 000 бит = 3 840 000 байт = 3075 Кбайт.

## *2.16.2 Обнаружение ошибок*

В реальных каналах связи всегда присутствуют помехи, искажающие сигнал. В некоторых случаях ошибки допустимы, например, при прослушивании радиопередачи через Интернет небольшое искажение звука не мешает понимать речь. Однако чаще всего требуется обеспечить точную передачу данных. Для этого в первую очередь нужно определить факт возникновения ошибки и, если это произошло, передать блок данных еще раз.

Представьте себе, что получена цепочка нулей и единиц 1010101110, причем все биты независимы. В этом случае нет абсолютно никакой возможности определить, верно ли передана последовательность. Поэтому необходимо вводить *избыточность* в передаваемое сообщение (включать в него «лишние» биты) только для того, чтобы обнаружить ошибку.

Простейший вариант – добавить 1 бит в конце блока данных, который будет равен 1, если в основном сообщении нечетное число единиц, и равен 0 для сообщения с четным числом единиц. Этот дополнительный бит называется **битом четности**. Бит четности используется при передаче данных в сетях, проверка четности часто реализуется аппаратно (с помощью электроники).

Например, требуется передать два бита данных. Возможно всего 4 разных сообщения: 00, 01, 10 и 11. Первое и четвертое из них содержат четное число единиц (0 и 2), значит, бит четности для них равен 0. Во втором и третьем сообщениях нечетное число единиц (1), поэтому бит четности будет равен 1. Таким образом, сообщения с добавленным битом четности будут выглядеть так:

000, 011, 101, 110.

Первые два бита несут полезную информацию, а третий – вспомогательный, он служит только для обнаружения ошибки. Обратим внимание, что каждое из этих сообщений содержит четное число единиц.

Подумаем, сколько ошибок может обнаружить такой метод. Если при передаче неверно передан только один из битов, количество единиц в сообщении стало нечетным, это и служит признаком ошибки при передаче. Однако исправить ошибку нельзя, потому что непонятно, в каком именно разряде она случилась.

Если же изменилось два бита, четность не меняется, и такая ошибка не обнаруживается. В длинной цепочке применение бита четности позволяет обнаруживать нечетное число ошибок (1, 3, 5, …), а ошибки в четном количестве разрядов остаются незамеченными.

Контроль четности хорошо работает тогда, когда отдельные ошибки при передаче независимы одна от другой и встречаются редко. Для обнаружения искажений в передаче блоков данных (например, файлов), когда может сразу возникнуть множество ошибок, используют

другой метод, который называется *методом контрольной суммы* (англ. CRC = *Cyclic Redundancy Check – проверка с помощью циклических сумм*). Если контрольная сумма блока данных, вычисленная приемником по специальному довольно сложному алгоритму, не совпадает с контрольной суммой, записанной передающей стороной, произошла ошибка.

## *2.16.3 Помехоустойчивые коды*

Значительно сложнее исправить ошибку сразу (без повторной передачи), однако в некоторых случаях и эту задачу удается решить. Для этого требуется настолько увеличить избыточность кода (добавить «лишние» биты), что небольшое число ошибок все равно позволяет достаточно уверенно распознать переданное сообщение. Например, несмотря на помехи в телефонной линии, обычно мы легко понимаем собеседника. Это значит, что речь обладает достаточно большой избыточностью, и это позволяет исправлять ошибки «на ходу».

Например, пусть нужно передать один бит, 0 или 1. Утроим его, добавив еще два бита, совпадающих с первым. Таким образом, получается два «правильных» сообщения:

000 и 111.

Теперь посмотрим, что получится, если при передаче одного из битов сообщения 000 произошла ошибка, и приемник получил искаженное сообщение 001. Заметим, что оно отличается одним битом от 000, и двумя битами от второго возможного варианта – 111. Значит, скорее всего, произошла ошибка в последнем бите и сообщение нужно исправить на 000. Если приемник получил 101, можно точно сказать, что произошла ошибка, однако попытка исправить ее приведет к неверному варианту, так как ближайшая «правильная» последовательность – это 111. Таким образом, такой код *обнаруживает* одну или две ошибки. Кроме того, он позволяет *исправить* (!) одну ошибку, то есть является *помехоустойчивым*.

**Помехоустойчивый код** – это код, который позволяет исправлять ошибки, если их количество не превышает некоторого уровня.

Выше мы фактически применили понятие «расстояния» между двумя кодами. В теории передачи информации эта величина называется «расстоянием Хэмминга» в честь американского математика Р. Хэмминга.

**Расстояние Хэмминга** – это количество позиций, в которых отличаются два закодированных сообщения одинаковой длины.

Например, расстояние $d$ между кодами 001 и 100 равно

$$d(\underline{0}0\underline{1}, \underline{1}0\underline{0}) = 2,$$

потому что они отличаются в двух битах (они подчеркнуты). В приведенном выше примере расстояние между «правильными» последовательностями (*словами*) равно $d(000, 111) = 3$. Такой код позволяет обнаружить одну или две ошибки и исправить одну ошибку.

В общем случае, если минимальное расстояние между «правильными» словами равно $d$, можно обнаружить от 1 до $d-1$ ошибок, потому что при этом полученный код будет отличаться от всех допустимых вариантов. Для исправления $r$ ошибок необходимо, чтобы выполнялось условие

$$d \geq 2r+1.$$

Это значит, что слово, в котором сделано $r$ ошибок, должно быть ближе к исходному слову (из которого оно получено искажением), чем к любому другому.

Рассмотрим более сложный пример. Пусть нужно передавать три произвольных бита, обеспечив обнаружение двух любых ошибок и исправление одной ошибки. В этом случае можно использовать, например, такой код с тремя контрольными битами (они подчеркнуты):

| | |
|---|---|
| 000 <u>000</u> | 100 <u>101</u> |
| 001 <u>111</u> | 101 <u>010</u> |
| 010 <u>011</u> | 110 <u>110</u> |
| 011 <u>100</u> | 111 <u>001</u> |

Расстояние Хэмминга между любыми двумя словами в таблице не менее 3, поэтому код обнаруживает две ошибки и позволяет исправить одну. Как же вычислить ошибочный бит?

Предположим, что было получено кодовое слово 011011. Определив расстояние Хэмминга до каждого из «правильных» слов, находим единственное слово 010011, расстояние до которого равно 1 (расстояния до остальных слов больше). Значит, скорее всего, это слово и было передано, но исказилось из-за помех.

На практике используют несколько более сложные коды, которые называются кодами *Хэмминга*. В них информационные и контрольные биты перемешаны, и за счет этого можно сразу, без перебора, определить номер бита, в котором произошла ошибка. Наиболее известен семибитный код, в котором 4 бита – это данные, а 3 бита – контрольные. В нем минимальное расстояние между словами равно 3, поэтому он позволяет обнаружить две ошибки и исправить одну.

## ? *Контрольные вопросы*

1. В чем можно измерять скорость передачи информации?
2. Почему для любого канала связи скорость передачи данных ограничена?
3. Как вычисляется объем информации, который можно передать за некоторое время?
4. В каких случаях при передаче информации допустимы незначительные ошибки?
5. Что такое избыточность сообщения? Для чего ее можно использовать?
6. Как помехи влияют на передачу информации?
7. Что такое бит четности? В каких случаях с помощью бита четности можно обнаружить ошибку, а в каких – нельзя?
8. Можно ли исправить ошибку, обнаружив неверное значение бита четности?
9. Для чего используется метод вычисления контрольной суммы?
10. Какой код называют помехоустойчивым?
11. Каково должно расстояние между двумя кодами, чтобы можно было исправить 2 ошибки?
12. Как исправляется ошибка при использовании помехоустойчивого кода?
13. Каковы возможности семибитного кода Хэмминга?

## *Задачи*

1. Через соединение со скоростью 128000 бит/с передают файл размером 625 Кбайт. Определите время передачи файла в секундах. (Ответ: 40 с)
2. Передача файла через соединение со скоростью 512 000 бит/с заняла 1 минуту. Определить размер файла в килобайтах. (Ответ: 3750 Кбайт)
3. Скорость передачи данных равна 64000 бит/с. Сколько времени займет передача файла объемом 375 Кбайт по этому каналу? (Ответ: 48 с)
4. У Васи есть доступ к Интернет по высокоскоростному одностороннему радиоканалу, обеспечивающему скорость получения им информации 256 Кбит/с. У Пети нет скоростного доступа в Интернет, но есть возможность получать информацию от Васи по низкоскоростному телефонному каналу со средней скоростью 32 Кбит/с. Петя договорился с Васей, что тот будет скачивать для него данные объемом 5 Мбайт по высокоскоростному

каналу и ретранслировать их Пете по низкоскоростному каналу. Компьютер Васи может начать ретрансляцию данных не раньше, чем им будут получены первые 512 Кбайт этих данных. Каков минимально возможный промежуток времени (в секундах), с момента начала скачивания Васей данных, до полного их получения Петей? (Ответ: 1296 с)

5. Определите максимальный размер файла в Кбайтах, который может быть передан за 10 минут со скоростью 32 килобита/с. (Ответ: 2400 байт)
6. Сколько секунд потребуется, чтобы передать 400 страниц текста в 30 строк по 60 символов каждая по линии со скоростью 128000 бит/с, при условии, что каждый символ кодируется 1 байтом?
7. Передача текстового файла по каналу связи со скоростью 128000 бит/с заняла 1,5 мин. Определите, сколько страниц содержал переданный текст, если известно, что он был представлен в 16-битной кодировке UNICODE, а на одной странице – 400 символов. (Ответ: 30 с)
8. Передача текстового файла через соединение со скоростью 64000 бит/с заняла 10 с. Определите, сколько страниц содержал переданный текст, если известно, что он был представлен в 16-битной кодировке UNICODE и на каждой странице – 400 символов. (Ответ: 100)
9. Сколько секунд потребуется, чтобы передать цветное растровое изображение размером 1280×800 пикселей по линии со скоростью 256000 бит/с при условии, что цвет каждого пикселя кодируется 24 битами? (Ответ: 96)
10. Сколько секунд потребуется, чтобы передать цветное растровое изображение размером 1000×800 пикселей по линии со скоростью 128000 бит/с при условии, что цвет каждого пикселя кодируется 24 битами? (Ответ: 150)
11. Через некоторый канал связи за 2 минуты был передан файл, размер которого 3 750 Кбайт. Определите минимальную скорость, при которой такая передача возможна. (Ответ: 256000 бит/с)
12. *Модем, передающий информацию со скоростью 256 000 бит/с, передал файл с несжатой стереофонической музыкой за 2 минуты 45 секунд. Найдите разрядность кодирования этой музыки, если известно, что ее продолжительность составила 1 минуту и оцифровка производилась с частотой 22000 Гц? (Ответ: 16 бит)
13. Найдите расстояние между кодами 11101 и 10110, YUIX и YAIY.
14. Найдите все пятизначные двоичные коды, расстояние от которых до кода 11101 равно 1. Сколько всего может быт таких слов для $n$-битного кода?
15. Для передачи восьмеричных чисел используется помехоустойчивый шестибитный код, приведенный в разд. 2.16.3. Раскодируйте сообщение, исправив ошибки:

    001 101 011 011 011 101 110 101 101 011

    (Ответ: 12345)

# 2.17 Сжатие информации

## *2.17.1 Основные понятия*

Для того чтобы сэкономить место на внешних носителях (винчестерах, флэш-дисках) и ускорить передачу информации по компьютерным сетям, нужно ее сжать – уменьшить информационный объем, сократить длину двоичного кода. Это можно сделать за счет использования *избыточности* информации.

Например, текстовый файл объемом 10 Кбайт содержит всего четырех различных символа: латинские буквы A, B, C и пробел. Вы знаете, что для кодирования одного из четырех возможных вариантов достаточно 2 битов, поэтому использовать для его передачи обычное 8-битное кодирование символов невыгодно. Можно присвоить каждому из четырех символов двухбитные коды, например, так:

A – 00, B – 01, C – 10, пробел – 11.

Тогда последовательность «ABA CABABA», занимающее 10 байт в однобайтной кодировке, может быть представлена как цепочка из 20 бит:

00010011100001000100

Таким образом, нам удалось уменьшить информационный объем текста в 4 раза, и передаваться оно будет в 4 раза быстрее. Однако непонятно, как раскодировать это сообщение, ведь получатель не знает, какой код был использован. Выход состоит в том, чтобы включить в сообщение служебную информацию – заголовок, в котором каждому коду будет сопоставлен ASCII-код символа. Условимся, что первый байт заголовка – это количество используемых символов $N$, а следующие $N$ байт – это ASCII-коды этих символов. В данном случае заголовок занимает 5 байт и выглядит так:

| $00000100_2$ | $01000001_2$ | $01000010_2$ | $01000011_2$ | $00100000_2$ |
|---|---|---|---|---|
| 4 символа | A (код 65) | B (код 66) | C (код 67) | пробел (код 32) |

Файл, занимающий 10 Кбайт в 8-битной кодировке, содержит 10240 символа. В сжатом виде каждый символ кодируется двумя битами, кроме того, есть 5-байтный заголовок. Поэтому сжатый файл будет иметь объем

5 + 1024·2/8 байт = 2565 байт.

**Коэффициент сжатия** – это отношение размеров исходного и сжатого файлов.

В данном случае удалось сжать файл почти в 4 раза, коэффициент сжатия равен

$k = 10240/2565 \approx 4$.

Если принимающая сторона «знает» формат файла (заголовок + закодированные данные), она сможет восстановить в точности его исходный вид. Такое сжатие называют *сжатием без потерь*. Он используется для упаковки текстов, программ, данных, которые ни в коем случае нельзя искажать.

**Сжатие без потерь** – это такое уменьшение объема данных, при котором можно восстановить их исходный вид из кода без искажений.

За счет чего удалось сжать файл? Только за счет того, что в файле была некоторая закономерность, избыточность – использовались только 4 символа вместо полного набора.

## *2.17.2 Алгоритм RLE*

При сжатии данных, в которых есть цепочки одинаковых кодов, можно применять еще один простой алгоритм, который называется *кодированием цепочек одинаковых символов* (англ. *RLE = Run Length Encoding*). Представим себе файл, в котором записаны сначала 100 русских букв А, а потом – 100 букв Б:

| 1 | 100 | 101 | 200 |
|---|---|---|---|
| АА……АА | | ББ……ББ | |

При использовании алгоритма RLE сначала записывается количество повторений первого символа, затем сам первый символ, затем количество повторений второго символа и т.д. В данном случае весь закодированный файл занимает 4 байта:

| $01100100_2$ | $11000000_2$ | $01100100_2$ | $11000001_2$ |
|---|---|---|---|
| 100 | А (код 192) | 100 | Б (код 193) |

Таким образом, мы сжали файл в 50 раз за счет того, что в нем снова была *избыточность* – цепочки одинаковых символов. Это сжатие без потерь, потому что, зная алгоритм упаковки, исходные данные можно восстановить из кода.

Очевидно, что такой подход будет приводить к *увеличению* (в 2 раза) объема данных в том случае, когда в файле нет соседних одинаковых символов. Чтобы улучшить RLE-результаты кодирования даже в этом наихудшем случае, алгоритм модифицировали следующим образом. Упакованная последовательность содержит управляющие байты, за каждым управляющим байтом следует один или несколько байтов данных. Если старший бит управляющего байта равен 1, то следующий за управляющим байт данных при распаковке нужно повторить столько раз, сколько записано в оставшихся 7 битах управляющего байта. Если же старший бит управляющего байта равен 0, то надо взять несколько следующих байтов данных без изменения. Сколько именно – записано в оставшихся 7 битах управляющего байта. Например, управляющий байт $10000111_2$ говорит о том, что следующий за ним байт надо повторить 7 раз, а управляющий байт $00000100_2$ – о том, что следующие за ним 4 байта надо взять без изменений. Например, последовательность

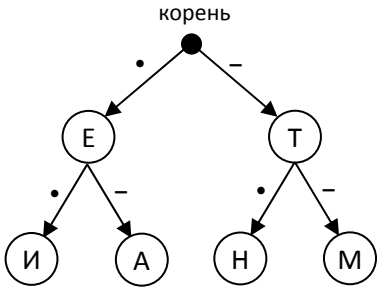
| $10001111_2$ | $11000000_2$ | $00000010_2$ | $11000001_2$ | $11000010_2$ |
|---|---|---|---|---|
| повтор 15 | А (код 192) | 2 | Б (код 193) | В (код 194) |

распаковывается в 17 символов: АААААААААААААААБВ.

Алгоритм RLE успешно применялся для сжатия рисунков, в которых большие области закрашены одним цветом, и некоторых звуковых данных. Сейчас вместо него применяют более совершенные, но более сложные методы. Один из них (алгоритм Хаффмана) мы рассмотрим ниже. Сейчас алгоритм RLE используется, например, на одном из этапов кодирования рисунков в формате JPEG. Возможность использования RLE-сжатия есть также в формате BMP (для рисунков с палитрой 16 или 256 цветов).

## *2.17.3 Префиксные коды*

Вспомните азбуку Морзе (разд. **Ошибка! Источник ссылки не найден.**), в которой для уменьшения длины сообщения используется неравномерный код – часто встречающиеся буквы (А, Е, М, Н, Т) кодируются короткими последовательностями, а редко встречающиеся – более длинными. Такой код можно представить в виде структуры, которая называется *деревом* (вспомните дерево каталогов):




Здесь показано неполное дерево кода Морзе, построенное только для символов, коды которых состоят из одного и двух знаков (точек и тире). Дерево состоит из узлов (черная точка и кружки с символами алфавита) и соединяющих их направленных ребер, стрелки указывают направление движения. Верхний узел (в который не входит ни одна стрелка) называется «корнем» дерева. Из корня и из всех промежуточных узлов (кроме конечных узлов – «листьев») выходят две стрелки,

левая помечена точкой, а правая – знаком «тире». Чтобы найти код символа, нужно пройти по стрелкам от «корня» дерева к нужному «листу», выписывая метки стрелок, по которым мы переходим. В дереве нет циклов (замкнутых путей), поэтому код каждого символа определяется единственным образом. По этому дереву можно построить такие коды:

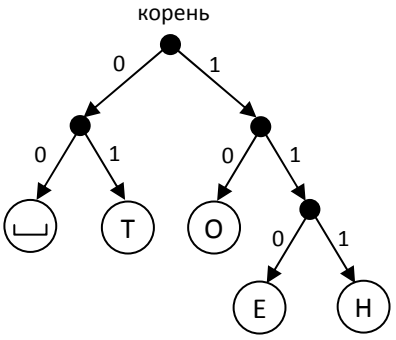
Е • И •• А • – Т – Н – • М – –

Это неравномерный код, в нем символы имеют коды разной длины. При этом всегда возникает проблема разделения последовательности на отдельные кодовые слова. В коде Морзе она решена с помощью символа-разделителя – паузы. Однако, можно не вводить дополнительный символ, если выполняется *условие Фано* (см. разд. 2.4) – ни одно из кодовых слов не является началом другого кодового слова. Это позволяет однозначно раскодировать сообщение в реальном времени, по мере получения очередных символов.

**Префиксный код** – это код, в котором ни одно кодовое слово не является началом другого кодового слова (условие Фано).

Для использования этой идеи в компьютерной обработке информации нужно было разработать алгоритм построения префиксного кода. Впервые эту задачу решили, независимо друг от друга, американские математики и инженеры Клод Шеннон и Роберт Фано (код Шеннона-Фано). Они использовали *избыточность* сообщений, состоящую в том, что символы в тексте имеют разные частоты встречаемости. В этом случае нужно читать данные исходного файла два раза: на первом проходе определяется частота встречаемости каждого символа, затем строится код с учетом этих данных, и на втором проходе символы текста заменяются на их коды.

Пусть, например, текст состоит только из букв О, Е, Н, Т и пробела. Известно, сколько раз они встретились в тексте: пробел – 179, О – 89, Е – 72, Н – 53 и Т – 50 раз. Код Шеннона-Фано, построенный для этого случая, можно нарисовать в виде дерева:




Символ ␣ на этой схеме обозначает пробел. В данном случае получаются такие коды символов:

␣ – 00, Т – 01, О – 10, Е – 110, Н – 111.

Легко проверить, что для этого набора кодов выполняется условие Фано. Это можно сразу определить по построенному дереву. В нем все символы располагаются в листьях, а не в промежуточных узлах. Это значит, что «по пути» от корня дерева до любого символа никаких других символов в промежуточных узлах не встречается (сравните с деревом кода Морзе).

Для раскодирования очередного символа последовательности мы просто спускаемся от корня дерева, выбирая левую ветку, если очередной бит – 0, и правую, если этот бит равен 1. Дойдя до листа дерева, мы определяем символ, а затем снова начинаем с корня дерева, чтобы раскодировать следующий символ, и т.д. Например, пусть получена последовательность

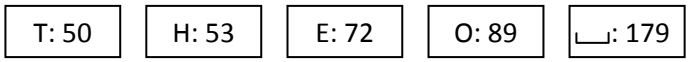
01100110001101111001.

Результат ее раскодирования – «ТОТО ЕНОТ».

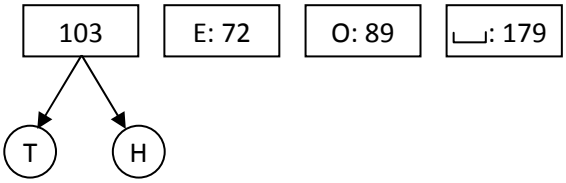
## *2.17.4 Алгоритм Хаффмана*

Однако, было доказано, что в некоторых случаях кодирование Шеннона-Фано дает неоптимальное решение, и можно построить код, который еще больше уменьшит длину кодовой последовательности. Например, в предыдущем примере (код Шеннона-Фано) пробел и буква Т имеют коды одинаковой длины, хотя буква Т встречается в 3,5 раза реже пробела. Через несколько лет Дэвид Хаффман, ученик Фано, разработал новый алгоритм кодирования и доказал его оптимальность.

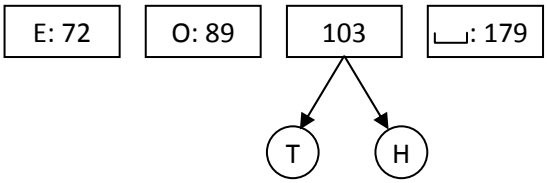
Построим код Хаффмана для того же самого примера. Сначала отсортируем буквы по увеличению частоты встречаемости:



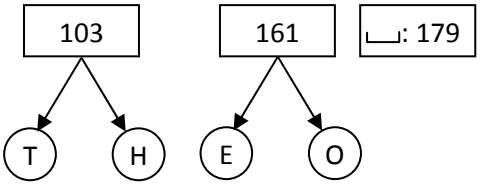

Затем берем две самых первых буквы, они становятся листьями дерева, а в узел, с которым они связаны, записываем сумму их частот:



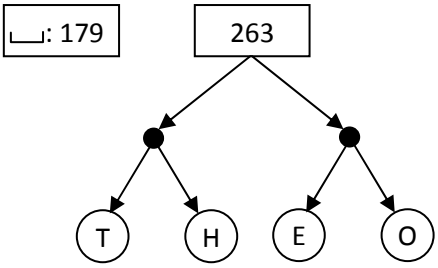

Таким образом, буквы, которые встречаются реже всего, получили самый длинный код. Дальше сортируем по возрастанию частоты в верхней строчке, причем для букв Т и Н используется их суммарная частота:



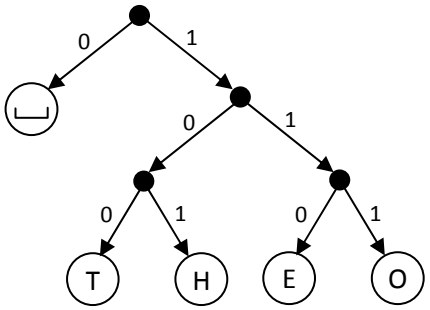

Повторяем ту же процедуру для букв Е и О, частоты которых оказались минимальны, и сортируем по возрастанию частот:




Теперь объединяем уже не отдельные буквы, а пары, и снова сортируем:




На последнем шаге остается объединить символ «пробел» с деревом, которое построено для остальных символов. У каждой стрелки, идущей влево от какого-то узла, ставим код 0, а у каждой стрелки, идущей вправо – код 1:




По этому дереву, спускаясь от корня к листьям-символам, получаем коды Хаффмана, обеспечивающие оптимальное сжатие с учетом частоты встречаемости символов:

␣ – 0, Т – 100, О – 111, Е – 110, Н – 101.

Легко проверить, что этот код также удовлетворяет условию Фано.

Сравним эффективность рассмотренных выше методов. Для алфавита из 5 символов при равномерном кодировании нужно использовать 3 бита на каждый символ, так что общее число бит в сообщении равно (179+89+72+53+50)·3 = 1329 бит. При кодировании методом Шеннона-Фано получаем 179·2+89·2+72·3+53·3+50·2 = 1011 бит, коэффициент сжатия составляет 1329/1011 ≈ 1,31. Использование кода Хаффмана дает последовательность длиной 179·1+89·3+72·3+53·3+50·3 = 971 бит и коэффициент сжатия 1,37. В сравнении со случаем, когда для передачи используется однобайтный код (8 бит на символ), выигрыш получается еще более весомым: кодирование Хаффмана сжимает данные примерно в 3,65 раза. Обратим внимание, что сжать данные удалось за счет *избыточности*: мы использовали тот факт, что некоторые символы встречаются чаще, а некоторые – реже.

Алгоритм кодирования Хаффмана и сейчас широко применяется благодаря своей простоте, высокой скорости кодирования и отсутствию патентных ограничений (он может быть использован свободно). Например, он используется на некоторых этапах при сжатии рисунков (в формате JPEG) и звуковой информации (в формате MP3).

## *2.17.5 Сжатие с потерями*

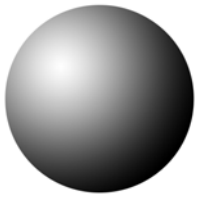
Выше мы рассмотрели методы *сжатия без потерь*, при которых можно в точности восстановить исходную информацию, зная алгоритм упаковки.

В некоторых случаях небольшие неточности в передаче данных несущественно влияют на решение задачи. Например, небольшие помехи в радиопередаче или телефонном разговоре, которая транслируется через Интернет, не мешают пониманию речи. Некоторое дополнительное размытие уже и без этого размытого изображения (фотографии) непрофессионал даже не заметит. Еще в большей степени это относится к видеофильмам. Поэтому иногда можно немного пожертвовать качеством изображения или звука ради того, чтобы значительно сократить объем данных. В этих случаях используется *сжатие с потерями*.

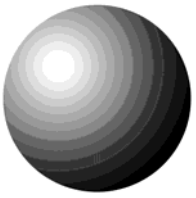
**Сжатие с потерями** – это методы сжатия данных, при которых распакованный файл отличается от оригинала.

Самые известные примеры сжатия с потерями – это алгоритмы JPEG (для изображений), MP3 (для упаковки звука) и все алгоритмы упаковки видеофильмов (MJPEG, MPEG4, DivX, XviD).

Простейшие метод сжатия рисунков – снижение глубины цвета. Ниже показаны три изображения с различной глубиной цвета. Третье из них занимает при кодировании в 4 раза меньше места, чем первое, хотя воспринимается уже с трудом.



8 бит на пиксель
(256 цветов)



4 бита на пиксель
(16 цветов)

2 бита на пиксель
(4 цвета)

Алгоритм JPEG (англ. *Joint Photographic Experts Group* – объединенная группа экспертов по фотографии) – один из наиболее эффективных методов сжатия изображений с потерями. Однако он очень сложен, и мы опишем только основные идеи этого алгоритма.

Из классической RGB-модели (красный, зеленый, синий) цвет переводится в модель YCbCr, где Y – яркость, Cb – «синева», и Cr – «краснота»:

$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B$

$Cb = 128 - 0{,}1687 \cdot R - 0{,}3313 \cdot G + 0{,}5 \cdot B$

$Cr = 128 + 0{,}5 \cdot R - 0{,}4187 \cdot G - 0{,}0813 \cdot B$

В этих формулах[19] R, G и B обозначают красную, зеленую и синюю составляющие цвета в RGB-модели. Черно-белое изображение (точнее, оттенки серого), содержит только Y-составляющую, компоненты цвета Cb и Cr равны 128 и не несут полезной информации.

Основная идея сжатия основана на том, что человеческий глаз более чувствителен к изменению яркости, то есть составляющей Y. Поэтому на синюю и красную составляющие, Cb и Cr, приходится меньше полезной информации, и на этом можно сэкономить. Пусть, например, есть квадрат 2 на 2 пикселя. Можно, например, сделать так: запомнить Y-компоненту для всех пикселей изображения (4 чисел), и по одной компоненте Cb и Cr на все 4 пикселя:

| | |
|---|---|
| $Y_1Cb_1Cr_1$ | $Y_2Cb_2Cr_2$ |
| $Y_3Cb_3Cr_3$ | $Y_4Cb_4Cr_4$ |

$\Rightarrow Y_1, Y_2, Y_3, Y_4, Cb, Cr$

Для определения сохраняемых значений Cb и Cr можно использовать, например, усреднение, например, принять

$$Cb = (Cb_1 + Cb_2 + Cb_3 + Cb_4) / 4, \qquad Cr = (Cr_1 + Cr_2 + Cr_3 + Cr_4) / 4.$$

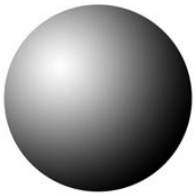
Таким образом, вместо 12 чисел мы должны хранить всего 6, данные сжаты в 2 раза. Однако, мы не сможем восстановить обратно исходный рисунок, потому что информация и том, какие были значения Cb и Cr для каждого пикселя, безвозвратно утеряна, есть только некоторые средние значения. При выводе такого изображения на экран можно, например, считать, что все 4 пикселя имели одинаковые значение Cb и Cr. Более сложные алгоритмы используют информацию о соседних блоках, сглаживая резкие переходы при изменении этих составляющих. Поэтому при кодировании с помощью алгоритма JPEG изображение несколько размывается.

Но это еще не все. После этого к оставшимся коэффициентам применяется сложное *дискретное косинусное преобразование*, с помощью которого удается выделить информацию, которая мало влияет на восприятие рисунка человеком, и отбросить ее. На последнем этапе для сжатия оставшихся данных используются алгоритмы RLE и Хаффмана.

[19] http://www.equasys.de/colorconversion.html

На следующих рисунках вы видите результат сжатия изображения шарика в формате JPEG с разным качеством

качество 100
(8400 байт)



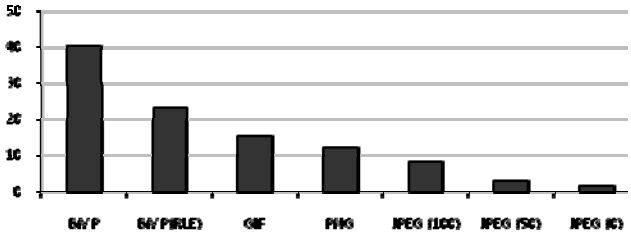
качество 50
(3165 байт)

качество 0
(1757 байт)

фрагмент
(качество 0)

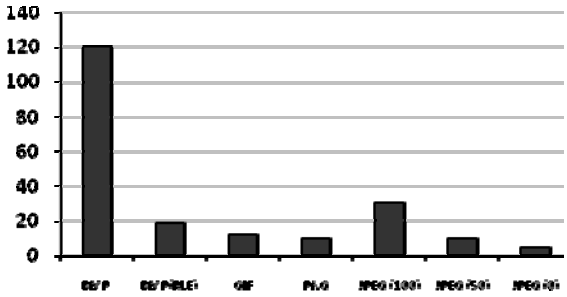
Рисунки, сохраненные с качеством 100 и 50, практически не отличаются внешне, однако второй из них занимает в 2,65 раза меньше места. При снижении качества до минимального явно заметны искажения (*артефакты*), вызванные потерей информации при сжатии (см. фрагмент). Кажется, что рисунок построен из квадратов размером 8 на 8 пикселей (в действительности именно такие квадраты использует алгоритм в качестве базовых ячеек). Кроме того, на краях искажения особенно заметны на границе между двумя цветами.

Тот же самый рисунок был сохранен в формате BMP без сжатия и с RLE-сжатием, а также в форматах GIF (сжатие без потерь с помощью алгоритма LZW) и PNG (сжатие без потерь с помощью алгоритма DEFLATE). Размеры полученных файлов (в Кбайтах) сравниваются на диаграмме.



По диаграмме видно, что наибольшее сжатие (наименьший объем файла) обеспечивается алгоритмом JPEG. Заметим, что особенность этого рисунка – плавные переходы между цветами.

Теперь рассмотрим другой рисунок, в котором у объектов есть большие области одного цвета и четкие границы у объектов. Здесь ситуация несколько меняется:




Форматы GIF и PNG обеспечивают степень сжатия, сравнимую с алгоритмом JPEG среднего качества. Учитывая, что JPEG приводит к искажению изображения, а GIF и PNG – нет, для таких рисунков не рекомендуется использовать формат JPEG.

Другой известный пример сжатия с потерями – алгоритм MPEG-1 Layer 3, который более известен как MP3. В нем отбрасываются некоторые компоненты звука, которые практически неразличимы для большинства людей. Такой подход называется *кодированием восприятия*, потому что он основан на особенностях восприятия звука человеком.

Одна из главных характеристик сжатия звукового файла – *битрейт* (англ. *bitrate* – темп поступления битов). Битрейт – это число бит, используемых для кодирования 1 с звука. Формат MP3 поддерживает разные степени сжатия, с битрейтом от 8 до 320 Кбит/с .

Звук, закодированный с битрейтом 256 Кбит/с и выше, даже профессионал вряд ли отличит от звучания компакт-диска. В разд. 2.14 мы определили, что 1 с звука занимает на компакт-диске (частота дискретизации 44 кГц, глубина кодирования 16 бит, стерео)

$$2 \times 88000 = 176\,000 \text{ байт} = 1\,408000 \text{ бит} = 1375 \text{ Кбит}.$$

Таким образом, формат MP3 обеспечивает степень сжатия $1375/256 \approx 5{,}4$ при сохранении качества звучания для человека (хотя часть информации, строго говоря, теряется).

Для сжатия видео может использоваться алгоритм MJPEG (англ. *Motion JPEG* – JPEG в движении), когда каждый кадр сжимается по алгоритму JPEG. Кроме того, широко применяют стандарт MPEG-4, разработанный специально для сжатия цифрового звука и видео.

## *2.17.6 Выводы*

Мы рассмотрели различные алгоритмы сжатия информации. Все они основаны на том, что в информации есть некоторая *избыточность*, закономерность, которую можно использовать для уменьшения объема данных. **Хорошо сжимается** информация, в которой большая избыточность:

- тексты, в которых повторяются одинаковые слова и символы имеют разные частоты встречаемости (файлы с расширениями **`*.txt`**, **`*.doc`**, **`*.docx`**);
- документы – тексты с оформлением и, возможно, вставленными рисунками, таблицами и т.п.; электронные таблицы (файлы с расширениями **`*.doc`**, **`*.docx`**, **`*.xls`**);
- рисунки, имеющие большие области одного цвета и записанные без сжатия (файлы **`*.bmp`**);
- несжатый звук (файлы **`*.wav`**);
- несжатое видео (файлы **`*.avi`**)[20].

**Плохо сжимаются** данные, где избыточность маленькая или ее совсем нет:

- программы (файлы **`*.exe`**);
- упакованные файлы (**`*.zip`**, **`*.rar`**, **`*.7z`** и др.);
- сжатые рисунки (файлы **`*.gif`**, **`*.jpg`**, **`*.png`**, **`*.tif`** и др.);
- сжатый звук (файлы **`*.mp3`**, **`*.wma`**);
- сжатое видео (файлы **`*.mpg`**, **`*.wmv`**).

Информацию невозможно сжать, если в ней нет никаких закономерностей. Поэтому хуже всего сжимаются случайные числа, например, полученные на компьютере. Современным программам-упаковщикам иногда удается их немного сжать, но не более, чем на 1-2%. Это происходит потому, что в последовательности псевдослучайных чисел, которые выдает компьютерная программа-генератор, все же можно выявить какие-то закономерности.

## ? *Контрольные вопросы*

1. За счет чего удается сжать данные без потерь? Когда это сделать принципиально невозможно?
2. Какие типы файлов сжимаются хорошо, а какие – плохо? Почему?
3. Текстовый файл, записанный в однобайтной кодировке, содержит только 33 заглавные русские буквы, цифры и пробел. Ответьте на следующие вопросы:

[20] Файлы с расширением **`*.avi`** могут хранить как сжатое, так и несжатое видео.

а) какое минимальное число бит нужно выделить на символ при передаче, если каждый символ кодируется одинаковым числом бит?
б) сколько при этом будет занимать заголовок пакета данных?
в) при какой минимальной длине текста коэффициент сжатия будет больше 1?

(Ответ: 6 бит; 45 байт; 23 символа)

4. На чем основан алгоритм сжатия RLE? Когда он работает хорошо? Когда нет смысла его использовать?
5. Что такое префиксный код?
6. В каких случаях допустимо сжатие с потерями?
7. Опишите простейшие методы сжатия рисунков с потерями.
8. На чем основан алгоритм JPEG? Почему это алгоритм сжатия с потерями?
9. Что такое артефакты?
10. Для каких типов изображений эффективно сжатие JPEG? Когда его не стоит применять?
11. На чем основано сжатие звука в алгоритме MP3?
12. Что такое битрейт? Как он связан с качеством звука?
13. Какое качество звука принимается за эталон качества на непрофессиональном уровне?
14. Какие методы используются для сжатия видео?

## *Задачи*

1. С помощью алгоритма RLE закодируйте сообщение «ВАААААВАААРРРРРРРРРР».
2. После кодирования методом RLE получилась следующая последовательность байтов (первый байт – управляющий):

   10000011 10101010 00000010 10101111 11111111 10000101 10101010

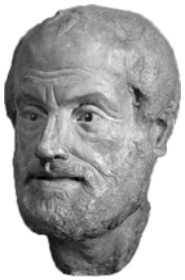
   Сколько байт будет содержать данная последовательность после распаковки? (Ответ: 10 байт)
3. После кодирования методом RLE получилась следующая последовательность байтов (первый байт – управляющий):

   00000011 10101010 00000010 10101111 10001111 11111111

   Сколько байт будет содержать данная последовательность после распаковки? (Ответ: 18 байт)
4. Раскодируйте сообщение, которое закодировано с помощью приведенного в тексте кода Шеннона-Фано : 11111000011011111001001101111001.
5. Постройте дерево, соответствующее коду А – 0, Б – 1, В – 00, Г – 01, Д – 10, Е – 11. Является ли этот код префиксным? Как это определить, посмотрев на дерево?
6. *Постройте дерево Хаффмана для фразы «МАМА МЫЛА ЛАМУ». Найдите коды всех входящих в нее символов и закодируйте сообщение. Чему равен коэффициент сжатия в сравнении с равномерным кодом минимальной длины? с однобайтной кодировкой?
   (Ответ: М – 11, А – 10, Л – 00, пробел – 011, Ы – 0100, У 0101; $k$ = 42/34 ≈ 1,24 ; $k$ = 3,29)

# 3. Логические основы компьютеров

## 3.1 Логика и компьютер


Аристотель
(384-322 до н.э.)

В быту мы часто используем слова «логика», «логично». Логика (от древнегреческого *λογικος* — «наука о рассуждении») – это наука о том, как правильно рассуждать, делать выводы, доказывать утверждения.

В естественном языке рассуждения всегда связаны с конкретными предметами и утверждениями, и поэтому исследовать все это многообразие достаточно сложно. Древнегреческий философ Аристотель стал основоположником *формальной логики*, которая отвлекается от конкретного содержания и изучает общие правила построения правильных выводов из известной информации, которая считается истинной. Формальная логика изучает *высказывания*.

**Высказывание** – это повествовательное предложение, про которое можно однозначно сказать, что оно истинно или ложно.

Используя это определение, проверим, можно ли считать высказываниями следующие предложения:

1) *Сейчас идет дождь.*
2) *Вчера жирафы улетели на север.*
3) *Красиво!*
4) *Который час?*
5) *В городе N живут более 2 миллионов человек.*
6) *Посмотрите на улицу.*
7) *У квадрата 10 сторон, и все разные.*
8) *История – интересный предмет.*

Здесь высказываниями являются только предложения 1, 2 и 7, остальные не подходят под определение. Утверждения 3 и 4 – это не повествовательные предложения. Предложение 5 станет высказыванием только в том случае, если «*N*» заменить на название конкретного города. Предложение 6 – это призыв к действию, а не утверждение. Утверждение 8 кто-то считает истинным, а кто-то ложным (нет однозначности), его можно более строго сформулировать в виде «*По мнению N, история – интересный предмет*». Для того, чтобы оно стало высказыванием, нужно заменить «*N*» на имя человека.

Какая же связь между логикой и компьютерами? В классической формальной логике высказывание может быть истинно или ложно, третий вариант исключается[1]. Если обозначить истинное значение единицей, а ложное – нулем, то получится, что формальная логика представляет собой правила выполнения операций с нулями и единицами, то есть с двоичными кодами. Как вы помните, именно такой способ используется в компьютерах для кодирования всех видов информации. Поэтому обработку информации оказалось возможным свести к выполнению логических операций. Важный шаг в этом направлении сделал английский математик Джордж Буль. Он предложил применить для исследования логических высказываний математические методы. Позже этот раздел математики получил название *алгебра логики* или *булева алгебра*.


Дж. Буль
(1815-1864)

**Алгебра логики** — это математический аппарат, с помощью которого записывают, вычисляют, упрощают и преобразовывают логические высказывания.

Алгебра логики определяет правила выполнения операций с *логическими* величинами, которые могут быть равны только 0 или 1, то есть с двоичными данными. Используя эти правила, можно

[1] Существуют неклассические логические системы, например, трехзначная логика, где кроме «истинно» и «ложно» есть еще состояние «не определено».

строить элементы памяти и выполнять арифметические действия. О том, как это сделать, вы узнаете в этой главе.

### ? *Контрольные вопросы*

1. Объясните значения слов «логика», «формальная логика», «алгебра логики».
2. Чем отличается формальная логика от «обычной», бытовой?
3. Что такое высказывание?
4. Можно ли считать высказываниями эти предложения:
   а) Не плачь, девчонка!
   б) Почему я водовоз?
   в) Купите слоника!
   г) Клубника очень вкусная.
   д) Сумма X и Y равна 36.
5. Как вы думаете, зачем в курсе информатики изучается логика?

## 3.2 Логические операции

Высказывания бывают простые и сложные. Простые высказывания нельзя разделить на более мелкие высказывания, например: «Сейчас идет дождь» или «Форточка открыта». Сложные (составные) высказывания строятся из простых с помощью логических связок (операций) «И», «ИЛИ», «НЕ», «если…, то», «тогда и только тогда».

В булевой алгебре высказывания обычно обозначаются латинскими буквами. Таким образом, мы уходим от конкретного содержания высказываний, нас интересует только их истинность или ложность. Например, можно обозначить буквой **A** высказывание «Сейчас идет дождь», а буквой **B** – высказывание «Форточка открыта». Их них строятся сложные высказывания:

не **A**: *«Сейчас нет дождя».*
не **B**: *«Форточка закрыта».*
**A** и **B**: *«Сейчас идет дождь и открыта форточка».*
**A** или **B**: *«Сейчас идет дождь или открыта форточка».*
если **A**, то **B**: *«Если сейчас идет дождь, то форточка открыта».*

Кроме этих есть еще и другие высказывания, которые можно получить из двух исходных. С некоторыми из них мы также познакомимся.

Операции «НЕ», «И» и «ИЛИ» используются чаще других. Оказывается, с их помощью можно выразить любую логическую операцию, поэтому эти три операции можно считать основными, базовыми.

### *3.2.1 Операция «НЕ»*

Операция «НЕ» часто называется отрицанием или *инверсией*. В алгебре логики всего два знака, 0 и 1, поэтому логические отрицание – это переход от одного значения к другому, от 1 к 0 или наоборот. Если высказывание **A** истинно, то «не **A**» ложно, и наоборот.

Для обозначения операции «НЕ» используется несколько способов. Выражение «не A» в алгебре логики записывается как $\overline{A}$ или **¬A**, в языках программирования Паскаль и Бейсик – как **not A**, в языке Си – как **!A**.

Операцию «НЕ» можно задать в виде таблицы:

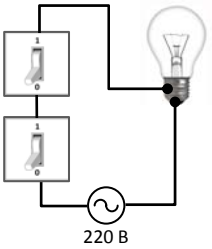
| A | $\overline{A}$ |
|---|---|
| 0 | 1 |
| 1 | 0 |

Эта таблица состоит из двух частей: слева перечисляются все возможные значения исходного сигнала (их всего два – 0 и 1), а в последнем столбце записывают результат выполнения логической операции для каждого из этих вариантов. Такая таблица называется **таблицей истинности** логической операции.

Таблица истинности задает **логическую функцию**, то есть правила преобразования входных логических значений в выходные.

## 3.2.2 Операция «И»

Пусть есть два высказывания, **A** – «Сейчас идет дождь», **B** – «Форточка открыта». Сложное высказывание «**A** и **B**» выглядит так: «Сейчас идет дождь и форточка открыта». Оно будет истинным (верным) в том и только в том случае, когда оба высказывания, **A** и **B**, истинны одновременно.




Для понимания операции «И» можно представить себе простую схему, в которой для включения лампочки используются два выключателя, соединенных последовательно (см. рисунок). Чтобы лампочка загорелась, нужно обязательно включить оба выключателя. С другой стороны, чтобы выключить лампочку, достаточно выключить любой из них.

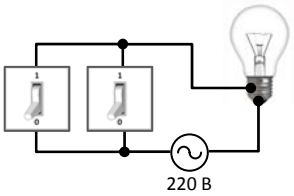
Операция «И» (в отличие от «НЕ») выполняется с двумя логическими значениями, которые мы обозначим как **A** и **B**. Результат этой операции в алгебре логики записывают как **A·B**, **A** ∧ **B** или **A & B**. В языках программирования используют обозначения «**A and B**» (Паскаль, Бейсик) или «**A && B**» (Си).

| | A | B | A·B |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 2 | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 |

В таблице истинности будет уже не один столбец с исходными данными, а два. Число строк также выросло с 2 до 4, поскольку для 2 бит мы получаем 4 разных комбинации: 00, 01, 10 и 11. Эти строчки расположены в определенном порядке: двоичные числа, полученные соединением битов **A** и **B**, идут в порядке возрастания (слева от таблицы они переведены в десятичную систему). Как следует из определения, в последнем столбце будет всего одна единица, для варианта **A = B = 1**.

Легко проверить, что этот результат можно получить «обычным» умножением **A** на **B**, поэтому операцию «И» называют *логическим умножением*. Существует и другое название этой операции – *конъюнкция* (от латинского *conjunctio* – союз, связь).

## 3.2.3 Операция «ИЛИ»




Высказывание «Сейчас идет дождь или форточка открыта» истинно тогда, когда истинно хотя бы одно из входящих в него высказываний, или оба одновременно. В алгебре логики операция «ИЛИ» обозначается как **A+B** или **A** ∨ **B**, в языках программирования – «**A or B**» (Паскаль, Бейсик) или «**A || B**» (Си).

| A | B | A+B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

Можно представить себе схему с двумя выключателями, соединенными параллельно (см. рисунок). Чтобы лампочка загорелась, достаточно включить хотя бы один из выключателей. Чтобы выключить лампочку, необходимо обязательно выключить оба. В таблице истинности будет только один ноль, для варианта **A = B = 0**.

Операцию «ИЛИ» называют *логическим сложением*, потому что она похоже на обычное математическое сложение. Единственное отличие – в последней строке таблицы истинности: в математике 1+1 равно 2, а в алгебре логики – 1. Другое название операции «ИЛИ» – *дизъюнкция* (от латинского *disjunctio* – разделение).

В учебнике для обозначения операций «И» и «ИЛИ» мы будем использовать знаки умножения и сложения (например, **A·B** и **A+B**). Это очень удобно потому, что они привычны для нас и позволяют легко увидеть аналогию с обычной математикой.

Доказано, что операций «НЕ», «И» и «ИЛИ» достаточно для того, чтобы записать с их помощью любую логическую операцию, которую только можно придумать. Например, для двух переменных существует всего $2^4 = 16$ логических операций: их таблицы истинности отличаются только

последним столбцом, в котором 4 двоичных значения (4 бита). Далее мы рассмотрим еще три известных операции и покажем, как их можно представить через операции «НЕ», «И» и «ИЛИ».

## *3.2.4 Операция «исключающее ИЛИ»*

| A | B | A ⊕ B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Операция «исключающее ИЛИ» отличается от обычного «ИЛИ» только тем, что результат равен 0, если оба значения равны 1 (последняя строчка в таблице истинности). То есть ее результат – истина в том и только в том случае, когда два значения *не равны*.

«Исключающее ИЛИ» в алгебре логики обозначается знаком $\oplus$, в языке Паскаль как **xor** (например, «**A xor B**»), а в языке Си – знаком **^** («**A ^ B**»). Эту операцию можно представить через базовые операции («НЕ», «И», «ИЛИ») следующим образом:

$$\mathbf{A} \oplus \mathbf{B} = \overline{\mathbf{A}} \cdot \mathbf{B} + \mathbf{A} \cdot \overline{\mathbf{B}}.$$

Пока мы не можем вывести эту формулу, но можем доказать ее (или опровергнуть – доказать, что она неправильная). Для этого достаточно для всех возможных комбинация **A** и **B** вычислить значения выражения, стоящего в правой части равенства и сравнить его со значением $\mathbf{A} \oplus \mathbf{B}$ для тех же исходных данных. Поскольку провести такие вычисления в уме достаточно сложно, сначала вычислим значения $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$, $\overline{\mathbf{A}} \cdot \mathbf{B}$ и $\mathbf{A} \cdot \overline{\mathbf{B}}$, а потом уже $\overline{\mathbf{A}} \cdot \mathbf{B} + \mathbf{A} \cdot \overline{\mathbf{B}}$. В таблице истинности появятся дополнительные столбцы для промежуточных результатов:

| A | B | $\overline{A}$ | $\overline{B}$ | $\overline{A} \cdot B$ | $A \cdot \overline{B}$ | $\overline{A} \cdot B + A \cdot \overline{B}$ | A ⊕ B |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

Легко видеть, что выражение $\overline{\mathbf{A}} \cdot \mathbf{B} + \mathbf{A} \cdot \overline{\mathbf{B}}$ совпадает с $\mathbf{A} \oplus \mathbf{B}$ для всех возможных вариантов. Это значит, что формула доказана.

Операция «исключающее ИЛИ» иначе называется *разделительной дизъюнкцией* (это значит «один или другой, но не оба вместе») или *сложением по модулю два*. Второе название связано с тем, что ее результат равен остатку от деления «обычной» суммы **A** + **B** на 2:

$$\mathbf{A} \oplus \mathbf{B} = \mathbf{(A + B)} \bmod 2.$$

Здесь **mod** обозначает операцию взятия остатка от деления.

Операция «исключающее ИЛИ» обладает интересными свойствами. По таблице истинности несложно проверить, что

$$\mathbf{A} \oplus \mathbf{0} = \mathbf{A}, \quad \mathbf{A} \oplus \mathbf{1} = \overline{\mathbf{A}}, \quad \mathbf{A} \oplus \mathbf{A} = \mathbf{0}.$$

Для доказательства этих равенств можно просто подставить в них **A = 0** и **A = 1**. Теперь докажем, что

$$\mathbf{(A \oplus B) \oplus B} = \mathbf{A}. \qquad (*)$$

Подставляя в левую часть **B = 0**, получим $\mathbf{(A \oplus 0) \oplus 0} = \mathbf{A \oplus 0} = \mathbf{A}$. Аналогично для **B = 1** имеем $\mathbf{(A \oplus 1) \oplus 1} = \overline{\mathbf{A}} \oplus \mathbf{1} = \mathbf{A}$. Это означает, что формула (*) справедлива для любых значений **B**. Отсюда следует важный вывод: если два раза применить операцию «исключающее ИЛИ» с одним и тем же **B**, мы восстановим исходное значение. В этом смысле «исключающее ИЛИ» – *обратимая* операция (кроме нее обратима также операция «НЕ» – если применить ее дважды, мы вернемся к исходному значению).

Формула (*) верна не только для высказываний, но и для чисел, состоящих из нескольких двоичных разрядов. Чтобы зашифровать данные, надо применить операцию «исключающее ИЛИ» с некоторым числом (кодом) отдельно для каждого разряда. Для расшифровки еще раз применяется «исключающее ИЛИ» с тем же ключом. Нужно отметить, что такой метод шифрования очень нестойкий: для больших текстов его легко раскрыть частотным анализом.

## 3.2.5 Импликация

| A | B | $A \to B$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Мы часто используем логическую связку «если…, то», например: «Если пойдет дождь, то я надену плащ» или «Если все стороны прямоугольника равны, то это квадрат». В логике эта связка называется *импликацией*[2] (следованием) и обозначается стрелкой: $\mathbf{A} \to \mathbf{B}$ («если A, то B», «из A следует B»).

Разобраться с импликацией будет легче, если мы рассмотрим конкретное высказывание, например, такое: «Если хорошо работаешь, то получаешь большую зарплату». Обозначим буквами два простых высказывания: **A** – «хорошо работаешь» и **B** – «получаешь большую зарплату». Понятно, что если высказывание $\mathbf{A} \to \mathbf{B}$ истинно, то все, кто хорошо работают ($\mathbf{A} = 1$) должны получать большую зарплату ($\mathbf{B} = 1$). Если же кто-то работает хорошо ($\mathbf{A} = 1$), а получает мало ($\mathbf{B} = 0$), то высказывание $\mathbf{A} \to \mathbf{B}$ ложно.

Лодыри и бездельники ($\mathbf{A} = 0$) могут получать как маленькую ($\mathbf{B} = 0$), так и большую зарплату ($\mathbf{B} = 1$), это не нарушает справедливость высказывания $\mathbf{A} \to \mathbf{B}$. Иногда, определяя импликацию, говорят так: из истины следует истина, а из лжи – что угодно. Это значит, что при ложном высказывании **A** высказывание **B** может быть как ложно, так и истинно.

Нужно обратить внимание на разницу между высказываниями вида «если **A**, то **B**» в обычной жизни и в алгебре логики. В быту мы чаще всего имеем в виду, что существует причинно-следственная связь между **A** и **B**, то есть именно **A** вызывает **B**. Алгебра логики не устанавливает взаимосвязь явлений; истинность высказывания $\mathbf{A} \to \mathbf{B}$ говорит только о *возможности* такой связи. Например, с точки зрения алгебры логики может быть истинным высказывание «если Вася – студент, то Петя – лыжник».

Импликация чаще всего используется при решении логических задач. Например, условие «если **A**, то **B**» можно записать в виде $\mathbf{A} \to \mathbf{B} = 1$.

Для импликации (в отличие от других изученных операций с двумя переменными) не действует переместительный закон: если в записи $\mathbf{A} \to \mathbf{B}$ поменять местами **A** и **B**, то результат изменится: $\mathbf{A} \to \mathbf{B} \neq \mathbf{B} \to \mathbf{A}$. Внешне это видно по стрелке, которая указывает «направление».

Импликацию можно заменить на выражение, использующее только базовые операции (здесь – только «НЕ» и «ИЛИ»):

$$\mathbf{A} \to \mathbf{B} = \overline{\mathbf{A}} + \mathbf{B}.$$

Доказать это равенство вы уже можете самостоятельно.

## 3.2.6 Эквивалентность

| A | B | $A \leftrightarrow B$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Эквивалентность (также эквиваленция, равносильность) – это логическая операция, которая соответствует связке «тогда и только тогда». Высказывание $\mathbf{A} \leftrightarrow \mathbf{B}$ истинно в том и только в том случае, когда $\mathbf{A} = \mathbf{B}$ (см. таблицу истинности).

Возможно, вы заметили, что эквивалентность – это обратная операция для «исключающего ИЛИ» (проверьте по таблицам истинности), то есть

$$\mathbf{A} \leftrightarrow \mathbf{B} = \overline{\mathbf{A} \oplus \mathbf{B}}.$$

Здесь черта сверху, охватывающая все выражение в правой части равенства, означает отрицание (инверсию), которое применяется к результату вычисления выражения $\mathbf{A} \oplus \mathbf{B}$, а не к отдельным высказываниям.

Можно заменить эквивалентность выражением, которое включает только базовые логические операции:

[2] От латинского *implicatio* – сплетение, тесная связь.

$$A \leftrightarrow B = \overline{A} \cdot \overline{B} + A \cdot B .$$

Эту формулу вы можете доказать (или опровергнуть) самостоятельно.

## 3.2.7 Другие логические операции

Мы уже говорили, что существуют и другие логические операции. Таблицы истинности операций с двумя переменными содержат 4 строки и отличаются только значением последнего столбца. Поэтому любая новая комбинация нулей и единиц в этом столбце дает новую логическую операцию (логическую функцию). Всего их, очевидно, столько, сколько существует четырехразрядных двоичных чисел, то есть $16 = 2^4$. Из тех, что мы еще не рассматривали, наиболее интересны две – *штрих Шеффера* («И-НЕ», англ. **nand** = «*not and*»)

$$A \mid B = \overline{A \cdot B}$$

и стрелка Пирса («ИЛИ-НЕ», англ. **nor** = «*not or*»).

$$A \downarrow B = \overline{A + B} .$$

Штрих Шеффера

| A | B | A \| B |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Стрелка Пирса

| A | B | $A \downarrow B$ |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Особенность этих операций состоит в том, что с помощью любой одной их них можно записать *произвольную* логическую операцию. Например, операции «НЕ», «И» и «ИЛИ» (базовый набор) выражаются через штрих Шеффера так:

$$\overline{A} = A \mid A , \quad A \cdot B = \overline{A \mid B} = (A \mid B) \mid (A \mid B), \quad A + B = \overline{A} \mid \overline{B} = (A \mid A) \mid (B \mid B) .$$

Эти формулы можно доказать через таблицы истинности.

## 3.2.8 Логические выражения

Обозначив простые высказывания буквами (переменными) и используя логические операции, можно записать любое высказывание в виде логического выражения. Например, пусть система сигнализации должна дать аварийный сигнал, если вышли из строя два из трех двигателей самолета. Обозначим высказывания:

**A** — «Первый двигатель вышел из строя».
**B** — «Второй двигатель вышел из строя».
**C** — «Третий двигатель вышел из строя».
**X** — «Аварийная ситуация».

Тогда логическое высказывание **X** можно записать в виде формулы

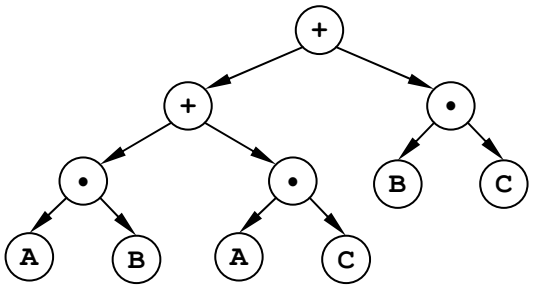
$$X = (A \cdot B) + (A \cdot C) + (B \cdot C). \qquad (*)$$

Таким образом, мы выполнили *формализацию*.

**Формализация** — это переход от конкретного содержания к формальной записи с помощью некоторого языка.

В логических выражениях операции выполняются в следующем порядке

1) действия в скобках;
2) отрицание («НЕ»);
3) логическое умножение («И»);
4) логическое сложение («ИЛИ») и «исключающее ИЛИ»;
5) импликация;
6) эквивалентность.




Такой порядок означает, что все скобки в выражении (*) для **X** можно убрать. Порядок вычисления выражения можно, так же, как и для арифметических выражений, определить с помощью дерева (см. рисунок). Вычисление начинается с листьев, корень – это самая последняя операция.

Здесь каждая операция выполняется с двумя значениями. Такие операции называются *бинарными* (лат. *bis* – дважды) или *двуместными*.

Операции, которые выполняются над одной величиной, называют *унарными* (лат. *uno* – один) или *одноместными*. Пример унарной логической операции – это отрицание (операция «НЕ»).

Любую формулу можно задать с помощью *таблицы истинности*, которая показывает, чему равно значение логического выражения при всех возможных комбинациях значений исходных переменных. Сложные выражения удобно разбить на несколько более простых, сначала вычислить значения этих промежуточных величин, а затем – окончательный результат.

| A | B | C | A·B | A·C | B·C | X |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Рассмотрим формулу (*). Выражение в правой части зависит от трех переменных, поэтому существует $2^3 = 8$ комбинаций их значений. Таблица истинности выглядит так, как показано на рисунке. По ней видно, что при некоторых значениях переменных значение **X** истинно, а при некоторых – ложно. Такие выражения называют *вычислимыми*.

Высказывание «Вася – школьник, или он не учится в школе» всегда истинно (для любого Васи). Оно может быть записано в виде логического выражения $\mathbf{A} + \overline{\mathbf{A}}$. Выражение, истинное при любых значениях переменных, называется *тождественно истинным* или *тавтологией*.

Высказывание «сегодня безветрие, и дует сильный ветер» никогда не может быть истинным. Соответствующее логическое выражение $\mathbf{A} \cdot \overline{\mathbf{A}}$ всегда ложно, оно называется *тождественно ложным* или *противоречием*.

Если два выражения принимают одинаковые значения при всех значениях переменных, они называются *равносильными* или *тождественно равными*. Например, равносильны выражения $\mathbf{A} \to \mathbf{B}$ и $\overline{\mathbf{A}} + \mathbf{B}$. Равносильные выражения определяют одну и ту же логическую функцию, то есть, при одинаковых исходных данных приводят к одинаковым результатам.

## 3.2.9 Некоторые задачи

Рассмотрим ряд задач, в которых требуется исследовать логическое выражение.

***Задача 1**. Каково наибольшее целое число **X**, при котором истинно высказывание*

$$\mathbf{A} = (90 < \mathbf{X}^2) \to (80 > (\mathbf{X}+2)^2)$$

Сначала удобно заменить импликацию по формуле $\mathbf{A} \to \mathbf{B} = \overline{\mathbf{A}} + \mathbf{B}$. Отрицание для высказывания $90 < \mathbf{X}^2$ запишется как $90 \ge \mathbf{X}^2$, поэтому

$$\mathbf{A} = (90 \ge \mathbf{X}^2) \text{ или } (80 > (\mathbf{X}+2)^2).$$

В этой задаче нас интересуют только целые числа. Поэтому условие $90 \ge \mathbf{X}^2$ можно заменить на $|\mathbf{X}| \le 9$ или $-9 \le \mathbf{X} \le 9$, а условие $80 > (\mathbf{X}+2)^2$ – на $|\mathbf{X}+2| \le 8$ или $-10 \le \mathbf{X} \le 6$. Таким образом, требуется выбрать наибольшее целое число, которое входит в один или в другой промежуток:


Это число – 9.

***Задача 2**. **A**, **B** и **C** – целые числа, для которых истинно высказывание*

$$\mathbf{X} = \overline{(\mathbf{A} = \mathbf{B})} \cdot ((\mathbf{A} > \mathbf{B}) \to (\mathbf{B} > \mathbf{C})) \cdot ((\mathbf{B} > \mathbf{A}) \to (\mathbf{C} > \mathbf{B}))$$

*Чему равно **B**, если **A = 27** и **C = 25**?.*

Это сложное высказывание состоит из трех простых:

$$\overline{(\mathbf{A} = \mathbf{B})},\ (\mathbf{A} > \mathbf{B}) \to (\mathbf{B} > \mathbf{C}),\ (\mathbf{B} > \mathbf{A}) \to (\mathbf{C} > \mathbf{B})$$

Они связаны операцией «И», то есть должны выполняться одновременно.

Из $\overline{(A = B)} = 1$ сразу следует, что $A \neq B$. Предположим, что $A > B$, тогда из второго условия получаем $1 \to (B > C)$ это выражение может быть истинно тогда и только тогда, когда $(B > C) = 1$, поэтому имеем $A > B > C$ этому условию соответствует только число 26.

На всякий случай проверим и вариант $A < B$, тогда из второго условия получаем $0 \to (B > C)$, это выражение истинно при любом B. Теперь проверяем третье условие: получаем $1 \to (C > B) = 1$; это выражение может быть истинно тогда и только тогда, когда $C > B$, и тут мы получили противоречие, потому что нет такого числа B, для которого $C > B > A$. Таким образом, правильный ответ – 26.

## ? *Контрольные вопросы*

1. Даны два высказывания: A – «В Африке водятся жирафы» и B – «В Мурманске идет снег». Постройте из них различные сложные высказывания.
2. Дано высказывание «*Винни-Пух любит мед, и дверь в дом открыта*». Как бы вы сформулировали отрицание этого высказывания?
3. Что такое таблица истинности?
4. Почему в таблице истинности для операции «НЕ» две строки, а для других изученных операций – четыре? Сколько строчек в таблице истинности выражения с тремя переменными? с четырьмя? с пятью?
5. В каком порядке обычно записываются значения переменных в таблице истинности?
6. Когда истинно высказывание «A и B»? «A или B»?
7. Какие электрические схемы можно использовать для иллюстрации операций «И» и «ИЛИ»?
8. Какие знаки применяют для обозначения операций «НЕ», «И», «ИЛИ»?
9. Почему операция «И» называется логическим умножением, а «ИЛИ» – логическим сложением?
10. В чем отличие «обычного» и логического сложения?
11. Сколько существует различных логических операций с двумя переменными? С тремя переменными?
12. Чем отличается операция «исключающее ИЛИ» от «ИЛИ»?
13. Почему операция «исключающее ИЛИ» называется сложением по модулю 2?
14. Как записать выражение $A \oplus B$ с помощью базового набора операций («НЕ», «И», «ИЛИ»)?
15. Как можно доказать или опровергнуть логическую формулу?
16. Какими интересными свойствами обладает операция «исключающее ИЛИ»?
17. Что значит выражение «обратимая операция»? Какие изученные логические операции являются обратимыми?
18. Какое свойство операции «исключающее ИЛИ» позволяет использовать ее для простейшего шифрования?
19. Чем отличается смысл высказывания «если A, то B» в обычной речи и в математической логике?
20. Запишите в виде формулы высказывание «если утюг горячий, то лоб холодный».
21. Запишите в виде формулы высказывание «неверно, что если утюг горячий, то лоб холодный». Можно ли в этом случае сразу сказать, какой утюг и какой лоб?
22. Как выразить импликацию через операции «НЕ» и «ИЛИ»? Докажите эту формулу.
23. Как выразить эквивалентность через операции «НЕ», «И» и «ИЛИ»? Докажите эту формулу.
24. Чем интересны операции «штрих Шеффера» и «стрелка Пирса»?
25. Докажите формулы, позволяющие представить базовые логические операции через штрих Шеффера. Попробуйте построить и доказать аналогичные формулы для операции «стрелка Пирса».
26. Что такое формализация?
27. В каком порядке выполняются действия в логических выражениях?
28. Что можно сделать для того, чтобы изменить «естественный» порядок действия?

29. Какие операции называются бинарными и унарными? Приведите примеры унарных и бинарных операций в математике.
30. Поясните разницу между терминами «логическое выражение» и «логическая функция».
31. Можно ли сказать, что таблица истинности однозначно определяет
    а) логическое выражение;
    б) логическую функцию.
32. Что такое вычислимое логическое выражение?
33. Что тавтология? противоречие? Приведите примеры.
34. Что такое равносильные выражения?

## *Задачи*

1. Составьте деревья для вычисления логических выражений и их таблицы истинности:

    а) $\overline{\mathbf{A}\cdot\mathbf{B}}+\mathbf{A}\cdot\mathbf{B}$
    б) $\mathbf{A}\cdot\mathbf{B}+\overline{\mathbf{A}}\cdot\overline{\mathbf{B}}+\mathbf{A}\cdot\overline{\mathbf{B}}$
    в) $(\mathbf{A}+\mathbf{B})\cdot(\overline{\mathbf{A}}+\overline{\mathbf{B}})\cdot(\mathbf{A}+\overline{\mathbf{B}})$
    г) $\mathbf{A}\cdot\overline{\mathbf{B}}+\mathbf{B}\cdot\overline{\mathbf{C}}+\mathbf{C}\cdot\overline{\mathbf{A}}$
    д) $\mathbf{A}\cdot\overline{\mathbf{B}}\cdot\mathbf{C}+\overline{\mathbf{A}}\cdot\mathbf{B}\cdot\overline{\mathbf{C}}+\mathbf{B}\cdot\mathbf{C}$
    е) $\mathbf{A}\cdot(\overline{\mathbf{B}}\cdot\mathbf{C}+\overline{\mathbf{A}})\cdot(\overline{\mathbf{C}}+\mathbf{B})$
    ж) $\overline{\mathbf{A}\cdot\mathbf{C}}+\overline{\mathbf{B}\cdot\overline{\mathbf{C}}}$
    з) $\overline{(\mathbf{A}+\mathbf{C})}+\overline{(\mathbf{B}+\overline{\mathbf{C}})}$
    и) $\overline{\overline{(\overline{\mathbf{A}}\cdot\mathbf{C})}\cdot\overline{(\overline{\mathbf{B}}\cdot\mathbf{C})}}$
    к) $\mathbf{A}\cdot(\mathbf{C}+\mathbf{B}\cdot\overline{\mathbf{C}})+\mathbf{C}\cdot\overline{(\mathbf{A}+\mathbf{B})}$
    л) $\mathbf{A}\cdot(\mathbf{C}+\overline{(\overline{\mathbf{B}}+\mathbf{C})})+\mathbf{B}\cdot\overline{(\mathbf{A}\cdot\mathbf{C})}$

2. Составьте деревья для вычисления логических выражений и их таблицы истинности:

    а) $(\mathbf{A}\to\mathbf{B})+(\overline{\mathbf{A}}\to\overline{\mathbf{B}})$
    б) $(\overline{\mathbf{A}}\to\mathbf{B})\cdot(\mathbf{A}\to\overline{\mathbf{B}})$
    в) $(\mathbf{A}\cdot\mathbf{B})\to(\overline{\mathbf{A}}+\overline{\mathbf{B}})$
    г) $(\mathbf{A}+\overline{\mathbf{B}})\to(\mathbf{A}\cdot\mathbf{B})$
    д) $(\mathbf{A}\to\overline{\mathbf{B}})\cdot(\mathbf{A}+\mathbf{C})\cdot(\overline{\mathbf{A}}\to\mathbf{C})$
    е) $(\overline{\mathbf{A}}\to\overline{\mathbf{B}})\to(\mathbf{A}\to\overline{\mathbf{C}})$
    ж) $\mathbf{A}\cdot\mathbf{B}\to(\mathbf{B}+\overline{\mathbf{C}})$
    з) $(\overline{\mathbf{A}}\to\mathbf{B})\to\overline{(\overline{\mathbf{A}}\to\overline{\mathbf{C}})}$
    и) $(\mathbf{A}\leftrightarrow\mathbf{B})+(\overline{\mathbf{A}}\leftrightarrow\mathbf{B})$
    к) $(\mathbf{A}\leftrightarrow\overline{\mathbf{B}})+(\mathbf{A}\leftrightarrow\mathbf{C})+(\overline{\mathbf{B}}\leftrightarrow\mathbf{C})$

3. Символом **F** обозначено одно из указанных ниже логических выражений от трех аргументов: **X**, **Y**, **Z**. Дан фрагмент таблицы истинности выражения **F**. Какие из этих выражений могут соответствовать **F**? (Ответ: а, б)

| X | Y | Z | F |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |

    м) $\mathbf{X}+\overline{\mathbf{Y}}+\overline{\mathbf{Z}}$
    н) $\mathbf{X}+\mathbf{Y}+\mathbf{Z}$
    о) $\overline{\mathbf{X}}+\mathbf{Y}+\mathbf{Z}$
    п) $\overline{\mathbf{X}}+\overline{\mathbf{Y}}+\overline{\mathbf{Z}}$

4. Для предыдущего задания определите, сколько различных логических функций соответствует заданной частичной таблице истинности? (Ответ: 32)
5. Задано 5 строк таблицы истинности некоторого логического выражения с тремя переменными. Сколько различных логических функций ей соответствуют? (Ответ: 8)
6. Символом **F** обозначено одно из указанных ниже логических выражений от трех аргументов: **X**, **Y**, **Z**. Дан фрагмент таблицы истинности выражения **F**. Какие из этих выражений могут соответствовать **F**? (Ответ: б, г)

| X | Y | Z | F |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |

    а) $\overline{\mathbf{X}}+\mathbf{Y}+\overline{\mathbf{Z}}$
    б) $\mathbf{X}\cdot\mathbf{Y}\cdot\overline{\mathbf{Z}}$
    в) $\overline{\mathbf{X}}\cdot\overline{\mathbf{Y}}\cdot\mathbf{Z}$
    г) $\mathbf{X}+\overline{\mathbf{Y}}\cdot\mathbf{Z}$

7. Символом **F** обозначено одно из указанных ниже логических выражений от трех аргументов: **X**, **Y**, **Z**. Дан фрагмент таблицы истинности выражения **F**. Какие из этих выражений могут соответствовать **F**? (Ответ: а, в)

| X | Y | Z | F |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 |

    а) $\mathbf{X}\to(\overline{\mathbf{Y}}+\overline{\mathbf{Z}})$
    б) $\overline{\mathbf{X}}\cdot\overline{\mathbf{Y}}\cdot\overline{\mathbf{Z}}$
    в) $\overline{\mathbf{X}}+\overline{\mathbf{Y}}+\overline{\mathbf{Z}}$

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 0 |

г) $X+Y+Z$

8. Символом **F** обозначено одно из указанных ниже логических выражений от трех аргументов: **X**, **Y**, **Z**. Дан фрагмент таблицы истинности выражения **F**. Какие из этих выражений могут соответствовать **F**? (Ответ: а)

| X | Y | Z | F |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 |

а) $X \cdot \overline{Y} \cdot \overline{Z}$
б) $X \to (\overline{Y} + \overline{Z})$
в) $X+Y+Z$
г) $Y \to (X \cdot Z)$

9. Символом **F** обозначено одно из указанных ниже логических выражений от трех аргументов: **X**, **Y**, **Z**. Дан фрагмент таблицы истинности выражения **F**. Какие из этих выражений могут соответствовать **F**? (Ответ: б, г)

| X | Y | Z | F |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |

а) $(X+\overline{Y}) \to Z$
б) $(\overline{X}+Y) \to Z$
в) $X+(Y \to Z)$
г) $X+Y \cdot Z$

10. Определите значение логического выражения $(X>2) \to (X>3)$ для $X=$ 1, 2, 3, 4. (Ответ: 1, 1, 0, 1)
11. Определите значение логического выражения
$$((X<5) \to (X<3)) \cdot ((X<2) \to (X<1))$$ для $X=$ 1, 2, 3, 4.
(Ответ: 0, 1, 0, 0)
12. Определите значение логического выражения
$$((X>3)+(X<3)) \to (X<1)$$ для $X=$ 1, 2, 3, 4.
(Ответ: 0, 0, 1, 0)
13. Определите значение логического выражения
$$((X<4) \to (X<3)) \cdot ((X<3) \to (X<1))$$ для $X=$ 1, 2, 3, 4.
(Ответ: 0, 0, 0, 1)
14. Определите значение логического выражения
$$(X \cdot (X-8) > 2 \cdot X - 25) \to (X>7)$$ для $X=$ 4, 5, 6, 7.
(Ответ: 0, 1, 0, 0)
15. Найдите все целые значения X, при которых логическое выражение
$$(X>2) \to (X>5)$$ ложно.
(Ответ: 3, 4, 5)
16. Найдите все целые значения X, при которых логического выражение
$$((X>0)+(X>4)) \to (X>4)$$ ложно.
(Ответ: 1, 2, 3, 4)
17. Автопилот может работать, если исправен главный бортовой компьютер или два вспомогательных. Выполните формализацию и запишите логические формулы для высказываний «автопилот работоспособен» и «автопилот неработоспособен».
18. Каково наибольшее целое положительное число **X**, при котором истинно высказывание:
$$(X(X+3) > X^2+9) \to (X(X+2) \le X^2+11)$$ ?
(Ответ: 5)
19. Каково наибольшее целое положительное число X, при котором истинно высказывание:
$$(121 < X^2) \to (X < X+5)$$?
(Ответ: 11)
20. Каково наибольшее целое положительное число X, при котором ложно высказывание:
$$(X(X+6)+9>0) \to (X^2>45)$$?
(Ответ: 6)
21. Каково наибольшее целое положительное число X, при котором истинно высказывание:
$$(X^2-1>100) \to (X(X-1)<100)$$?
(Ответ: 10)
22. Каково наибольшее целое положительное число X, при котором ложно высказывание:

$$(7X-3<75)\rightarrow(X(X-1)>65)?$$

(Ответ: 8)

23. Известно, что для чисел A, B и C истинно высказывание

$$((C<A)+(C<B))\cdot\overline{((C+1)<A)}\cdot\overline{((C+1)<B)}.$$

а) Чему равно C, если $A=25$ и $B=48$?

б) Чему равно C, если $A=45$ и $B=18$?

(Ответ: 47, 44)

24. Известно, что для чисел A, B и C истинно высказывание

$$\overline{(A=B)}\cdot((B<A)\rightarrow(2C>A))\cdot((A<B)\rightarrow(A>2C)).$$

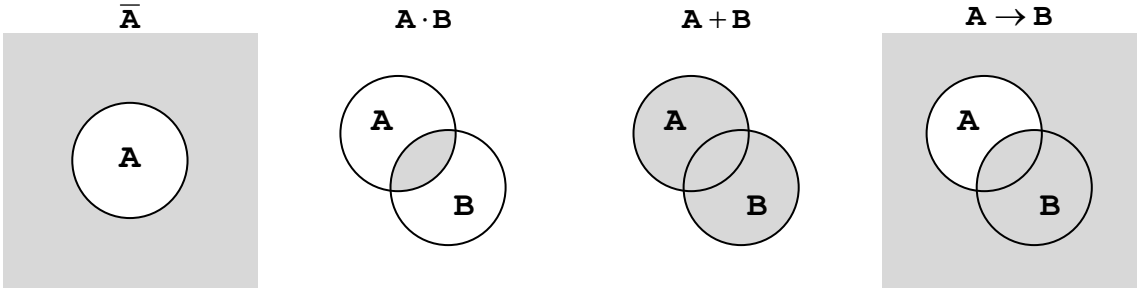
Чему равно A, если $C=10$ и $B=22$?

(Ответ: 21)

## 3.3 Диаграммы

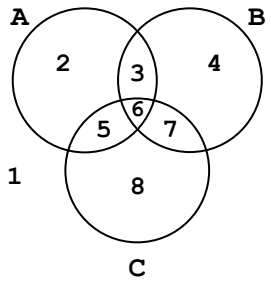
Выражения, зависящие от небольшого количества переменных (обычно не более 4-х), удобно изображать в виде диаграмм, которые называют *диаграммами Венна* или *кругами Эйлера*.

На такой диаграмме каждой переменной соответствует круг, внутри которого она равна единице, а вне его – нулю. Круги пересекаются, каждый с каждым. Области, в которых рассматриваемое логическое выражение истинно, закрашиваются каким–либо цветом. Ниже приведены диаграммы для простейших операций с одной и двумя переменными. Серым цветом залиты области, где рассматриваемое выражение равно единице.




Такие диаграммы часто используются при работе с множествами: операция «И» соответствует пересечению двух множеств, а «ИЛИ» – объединению.

Для трех переменных диаграмма будет немного сложнее. Для каждой из областей показанной ниже диаграммы запишем логические выражения:




1: $\overline{A}\cdot\overline{B}\cdot\overline{C}$
2: $A\cdot\overline{B}\cdot\overline{C}$
3: $A\cdot B\cdot\overline{C}$
4: $\overline{A}\cdot B\cdot\overline{C}$
5: $A\cdot\overline{B}\cdot C$
6: $A\cdot B\cdot C$
7: $\overline{A}\cdot B\cdot C$
8: $\overline{A}\cdot\overline{B}\cdot C$

Для того, чтобы найти выражение для объединения двух или нескольких областей, надо сложить (используя логическое сложение – операцию «ИЛИ») выражения для всех составляющих. Например, выражение для объединения областей 3 и 4 имеет вид

3 + 4: $A\cdot B\cdot\overline{C}+\overline{A}\cdot B\cdot\overline{C}$.

С другой стороны, можно заметить, что справедлива формула

3 + 4: $B\cdot\overline{C}$.

Это означает, что логические выражения в некоторых случаях можно упростить. Как это делается, вы узнаете в следующем параграфе.

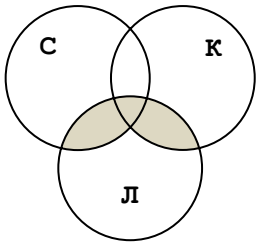
Диаграммы удобно применять для решения задач, в которых используются множества, например, множества ссылок, полученных от поисковой системы в ответ на какой-то запрос. Рассмотрим такую задачу:

*Известно количество ссылок, которые находит поисковый сервер по следующим запросам (здесь символ «&» обозначает операцию «И», а «|» – операцию «ИЛИ»):*

| | |
|---|---|
| **собаки** | **200** |
| **кошки** | **250** |
| **лемуры** | **450** |
| **кошки \| собаки** | **450** |
| **кошки & лемуры** | **40** |
| **собаки & лемуры** | **50** |

*Сколько ссылок найдет этот сервер по запросу*

**(кошки | собаки) & лемуры?**




Обозначим буквами **С**, **К** и **Л** высказывания «ключевое слово – собаки», «ключевое слово – кошки» и «ключевое слово – лемуры». Построим диаграмму с тремя переменными и выделим интересующую область, которая соответствует запросу

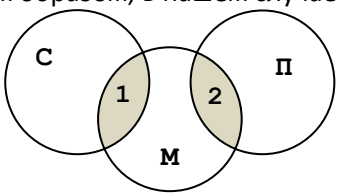
**(кошки | собаки) & лемуры**

На рисунке эта область закрашена серым цветом.

В общем виде задача очень сложна. Попробуем найти какое-нибудь упрощающее условие. Например, выделим три условия

| | |
|---|---|
| **собаки** | **200** |
| **кошки** | **250** |
| **кошки \| собаки** | **450** |

Это означает, что область «кошки ИЛИ собаки» равна сумме областей «кошки» и «собаки», то есть эти области *не пересекаются*! Таким образом, в нашем случае диаграмма выглядит так:




Области 1 (**собаки & лемуры**) и 2 (**кошки & лемуры**) нам известны, они составляют соответственно 40 и 50 ссылок, поэтому по запросу **(кошки | собаки) & лемуры** поисковый сервер выдаст 40 + 50 = 90 ссылок.

## *Задачи*

1. Используя диаграмму с тремя переменными, запишите логические выражения для объединения областей 2 + 5, 3 + 6, 4 + 7, 6 + 7, 5 + 6, 5 + 8, 7 + 8. Для каждой сложной области найдите два эквивалентных выражения.
2. Известно количество ссылок, которые находит поисковый сервер по следующим запросам:

   | | |
   |---|---|
   | **собаки** | **200** |
   | **кошки** | **250** |
   | **лемуры** | **450** |
   | **кошки \| собаки** | **450** |
   | **кошки & лемуры** | **40** |
   | **собаки & лемуры** | **50** |

   Сколько ссылок найдет этот сервер по запросу

   **кошки | собаки | лемуры?**

   (Ответ: 810)
3. Известно количество ссылок, которые находит поисковый сервер по следующим запросам:

   | | |
   |---|---|
   | **собаки** | **250** |
   | **кошки** | **200** |
   | **лемуры** | **500** |
   | **собаки & лемуры** | **0** |

| | |
|---|---|
| **собаки & кошки** | **20** |
| **кошки & лемуры** | **10** |

Сколько ссылок найдет этот сервер по запросу

**кошки | собаки | лемуры?**

(Ответ: 920)

4. Известно количество ссылок, которые находит поисковый сервер по следующим запросам:

| | |
|---|---|
| **собаки** | **120** |
| **кошки** | **270** |
| **лемуры** | **100** |
| **кошки \| собаки** | **390** |
| **кошки & лемуры** | **20** |
| **собаки & лемуры** | **10** |

Сколько ссылок найдет этот сервер по запросу

**кошки | собаки | лемуры?**

(Ответ: 460)

# 3.4 Упрощение логических выражений

## *3.4.1 Законы алгебры логики*

Для упрощения логических выражений используют законы алгебры логики. Они формулируются для базовых логических операций – «НЕ», «И» и «ИЛИ».

| Закон | для «И» | для «ИЛИ» |
|---|---|---|
| двойного отрицания | $\overline{\overline{A}} = A$ | |
| исключения третьего | $A \cdot \overline{A} = 0$ | $A + \overline{A} = 1$ |
| операции с константами | $A \cdot 1 = A$, $A \cdot 0 = 0$ | $A + 1 = 1$, $A + 0 = A$ |
| повторения | $A \cdot A = A$ | $A + A = A$ |
| переместительный | $A \cdot B = B \cdot A$ | $A + B = B + A$ |
| сочетательный | $A \cdot (B \cdot C) = (A \cdot B) \cdot C$ | $A + (B + C) = (A + B) + C$ |
| распределительный | $A + B \cdot C = (A + B) \cdot (A + C)$ | $A \cdot (B + C) = A \cdot B + A \cdot C$ |
| поглощения | $A + A \cdot B = A$ | $A \cdot (A + B) = A$ |
| законы де Моргана | $\overline{A \cdot B} = \overline{A} + \overline{B}$ | $\overline{A \cdot B} = \overline{A} + \overline{B}$ |

Закон двойного отрицания означает, что операция «НЕ» обратима: если применить ее два раза, логическое значение не изменится. Закон исключения третьего основан на том, что любое логическое выражение либо истинно, либо ложно («третьего не дано»). Поэтому если $A = 1$, то $\overline{A} = 0$ (и наоборот), так что произведение этих величин всегда равно нулю, а сумма – единице.

Операции с константами и закон повторения легко проверяются по таблицам истинности операций «И» и «ИЛИ». Переместительный и сочетательный законы выглядят вполне привычно, так же, как и в математике. Почти везде «работает» аналогия с обычной алгеброй, нужно только помнить, что в логике 1 + 1 = 1, а не 2.

Распределительный закон для «ИЛИ» – это обычное раскрытие скобок. А вот для операции «И» мы видим незнакомое выражение, в математике это равенство неверно. Доказательство можно начать с правой части, раскрыв скобки:

$$(A + B) \cdot (A + C) = A \cdot A + A \cdot C + B \cdot A + B \cdot C$$

Дальше используем закон повторения ( $A \cdot A = A$ ) и заметим, что

$$A + A \cdot C = A \cdot (1 + C) = A \cdot 1 = A .$$

Аналогично доказываем, что $A + B \cdot A = A \cdot (1 + B) = A$, таким образом,

$$(A + B) \cdot (A + C) = A + B \cdot C .$$

Равенство доказано. Попутно мы доказали также и закон поглощения для операции «И» (для «ИЛИ» вы можете сделать это самостоятельно). Отметим, что из распределительного закона следует полезная формула

$$\mathbf{A}+\overline{\mathbf{A}}\cdot\mathbf{B}=(\mathbf{A}+\overline{\mathbf{A}})\cdot(\mathbf{A}+\mathbf{B})=\mathbf{A}+\mathbf{B}.$$


А. де Морган
(1806-1871)

Правила, позволяющие раскрывать отрицание сложных выражений, названы в честь шотландского математика и логика де Моргана. Обратите внимание, что при этом не просто «общее» отрицание переходит на отдельные выражения, но и операция «И» заменяется на «ИЛИ» (и наоборот). Доказать законы де Моргана можно с помощью таблиц истинности.

Теперь с помощью приведенных законов алгебры логики упростим полученное ранее логическое выражение для объединения областей 3 и 4 на диаграмме с тремя переменными:

$$\mathbf{A}\cdot\mathbf{B}\cdot\overline{\mathbf{C}}+\overline{\mathbf{A}}\cdot\mathbf{B}\cdot\overline{\mathbf{C}}=(\mathbf{A}+\overline{\mathbf{A}})\cdot\mathbf{B}\cdot\overline{\mathbf{C}}=\mathbf{B}\cdot\overline{\mathbf{C}}.$$

Здесь мы сначала вынесли общий множитель двух слагаемых за скобки, а затем применили закон исключения третьего.

В общем случае можно рекомендовать такую последовательность действий:

1. Заменить все «небазовые» операции (исключающее ИЛИ, импликацию, эквивалентность и др.) на их выражения через базовые операции «НЕ», «И» и «ИЛИ».
2. Раскрыть отрицания сложных выражений по законам де Моргана так, чтобы операции отрицания остались только у отдельных переменных.
3. Используя вынесение общих множителей за скобки, раскрытие скобок и другие законы алгебры логики, упростить выражение.

**Пример**.

$$(\mathbf{A}+\overline{\mathbf{B}})\cdot\overline{(\mathbf{A}+\mathbf{B})}\cdot(\overline{\mathbf{A}}+\mathbf{C})=(\mathbf{A}+\overline{\mathbf{B}})\cdot\overline{\mathbf{A}}\cdot\overline{\mathbf{B}}\cdot(\overline{\mathbf{A}}+\mathbf{C})=(\mathbf{A}\cdot\overline{\mathbf{A}}+\overline{\mathbf{B}}\cdot\overline{\mathbf{A}})\cdot\overline{\mathbf{B}}\cdot(\overline{\mathbf{A}}+\mathbf{C})=$$

$$=\overline{\mathbf{B}}\cdot\overline{\mathbf{A}}\cdot\overline{\mathbf{B}}\cdot(\overline{\mathbf{A}}+\mathbf{C})=\overline{\mathbf{A}}\cdot\overline{\mathbf{B}}\cdot\overline{\mathbf{B}}\cdot(\overline{\mathbf{A}}+\mathbf{C})=\overline{\mathbf{A}}\cdot\overline{\mathbf{B}}\cdot(\overline{\mathbf{A}}+\mathbf{C})=\overline{\mathbf{B}}\cdot\overline{\mathbf{A}}\cdot(\overline{\mathbf{A}}+\mathbf{C})=\overline{\mathbf{B}}\cdot\overline{\mathbf{A}}.$$

Здесь последовательно использованы закон де Моргана, распределительный закон, закон исключения третьего, переместительный закон, закон повторения, снова переместительный закон и закон поглощения.

## *3.4.2 Логические уравнения*

Если приравнять два логических выражения, мы получим уравнение. Его решением будут значения переменных, при которых уравнение превращается в тождество, то есть значения левой и правой частей совпадают. Например, уравнение $\mathbf{A}\cdot\mathbf{B}=\mathbf{1}$ имеет единственное решение: $\mathbf{A}=\mathbf{B}=\mathbf{1}$, для остальных комбинаций значений переменных левая часть равна нулю. В то же время уравнение $\mathbf{A}+\mathbf{B}=\mathbf{1}$ имеет три решения: ($\mathbf{A}=\mathbf{0}$, $\mathbf{B}=\mathbf{1}$), ($\mathbf{A}=\mathbf{1}$, $\mathbf{B}=\mathbf{0}$) и $\mathbf{A}=\mathbf{B}=\mathbf{1}$.

***Пример 1.*** Требуется найти все решения уравнения

$$\overline{(\mathbf{B}+\mathbf{C})}\cdot\mathbf{A})\rightarrow(\overline{\mathbf{A}}\cdot\overline{\mathbf{C}}+\mathbf{D})=\mathbf{0}.$$

Вспоминаем, что импликация равна нулю только тогда, когда первое выражение равно 1, а второе – 0. Поэтому исходное уравнение сразу разбивается на два

$$\overline{(\mathbf{B}+\mathbf{C})}\cdot\mathbf{A}=\mathbf{1},\quad\overline{\mathbf{A}}\cdot\overline{\mathbf{C}}+\mathbf{D}=\mathbf{0}.$$

Первое уравнение с помощью закона де Моргана можно преобразовать к виду $\overline{\mathbf{B}}\cdot\overline{\mathbf{C}}\cdot\mathbf{A}=\mathbf{1}$, откуда сразу следует, что все три сомножителя должны быть равны 1. Это значит, что $\mathbf{A}=\mathbf{1}$, $\mathbf{B}=\mathbf{0}$ и $\mathbf{C}=\mathbf{0}$. Кроме того, из второго уравнения следует, что $\mathbf{D}=\mathbf{0}$. Решение найдено, причем оно единственное.

Возможен другой вариант – упростить выражение. Заменяя импликацию по формуле $\mathbf{A}\rightarrow\mathbf{B}=\overline{\mathbf{A}}+\mathbf{B}$, получаем

$$\overline{\overline{(\mathbf{B}+\mathbf{C})}\cdot\mathbf{A})}+\overline{\mathbf{A}}\cdot\overline{\mathbf{C}}+\mathbf{D}=\mathbf{0}.$$

Используем закон де Моргана:

$$\mathbf{B}+\mathbf{C}+\overline{\mathbf{A}}+\overline{\mathbf{A}}\cdot\overline{\mathbf{C}}+\mathbf{D}=\mathbf{0}$$

и закон поглощения

$$\mathbf{B}+\mathbf{C}+\overline{\mathbf{A}}+\mathbf{D}=\mathbf{0}.$$

Для того, чтобы логическая сумма была равна нулю, каждое слагаемое должно быть равно нулю, поэтому $\mathbf{A}=\mathbf{1}$, $\mathbf{B}=\mathbf{C}=\mathbf{D}=\mathbf{0}$.

Есть и третий вариант – построить таблицу истинности выражения в левой части и найти все варианты, при которых оно равно 0. Однако таблица истинности выражения с четырьмя переменными содержит $2^4$ = 16 строк, поэтому такой подход достаточно трудоемок.

***Пример 2.*** Требуется найти все решения уравнения

$$(\mathbf{A}+\overline{\mathbf{B}})\to(\mathbf{B}\cdot\mathbf{C}\cdot\mathbf{D})=\mathbf{1}.$$

Преобразуем выражение, раскрыв импликацию через «НЕ» и «ИЛИ» и применив закон де Моргана:

$$\overline{\mathbf{A}+\overline{\mathbf{B}}}+\mathbf{B}\cdot\mathbf{C}\cdot\mathbf{D}=\overline{\mathbf{A}}\cdot\mathbf{B}+\mathbf{B}\cdot\mathbf{C}\cdot\mathbf{D}=\mathbf{1}.$$

Если логическая сумма равна 1, то хотя бы одно слагаемое равно 1 (или оба одновременно).

Равенство $\overline{\mathbf{A}}\cdot\mathbf{B}=\mathbf{1}$ верно при $\mathbf{A}=\mathbf{0}$, $\mathbf{B}=\mathbf{1}$ и любых **C** и **D**. Поскольку есть всего 4 комбинации значений **C** и **D**, уравнение $\overline{\mathbf{A}}\cdot\mathbf{B}=\mathbf{1}$ имеет 4 решения:

| A | B | C | D |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 |

Второе уравнение, $\mathbf{B}\cdot\mathbf{C}\cdot\mathbf{D}=\mathbf{1}$, дает $\mathbf{B}=\mathbf{C}=\mathbf{D}=\mathbf{1}$ при любом **A**, то есть оно имеет два решения:

| A | B | C | D | |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | повторяется |
| 1 | 1 | 1 | 1 | |

Видим, что первое из этих решений уже было получено раньше, поэтому уравнение имеет всего пять разных решений. Заметим, что определить все повторяющиеся решения можно из уравнения $(\overline{\mathbf{A}}\cdot\mathbf{B})\cdot(\mathbf{B}\cdot\mathbf{C}\cdot\mathbf{D})=\mathbf{1}$, которое имеет единственное решение $\mathbf{A}=\mathbf{0}$, $\mathbf{B}=\mathbf{C}=\mathbf{D}=\mathbf{1}$.

***Пример 3.*** Требуется найти число решений уравнения

$$\mathbf{A}\cdot\mathbf{B}\cdot\mathbf{C}+\overline{\mathbf{B}}\cdot\overline{\mathbf{C}}\cdot\mathbf{D}=\mathbf{0}.$$

Здесь, в отличие от предыдущих задач, не нужно находить сами решения, интересует только их количество. Уравнение распадается на два

$$\mathbf{A}\cdot\mathbf{B}\cdot\mathbf{C}=\mathbf{0} \text{ и } \overline{\mathbf{B}}\cdot\overline{\mathbf{C}}\cdot\mathbf{D}=\mathbf{0}.$$

Каждое из них имеет достаточно много решений. Можно поступить следующим образом: сначала найти количество решений «обратного» уравнения, с единицей в правой части:

$$\mathbf{A}\cdot\mathbf{B}\cdot\mathbf{C}+\overline{\mathbf{B}}\cdot\overline{\mathbf{C}}\cdot\mathbf{D}=\mathbf{1},$$

и затем вычесть его из 16 (общего количества комбинаций четырех переменных). Уравнение $\mathbf{A}\cdot\mathbf{B}\cdot\mathbf{C}=\mathbf{1}$ имеет два решения: $\mathbf{A}=\mathbf{B}=\mathbf{C}=\mathbf{1}$ и любое **D** (0 или 1). Второе уравнение, $\overline{\mathbf{B}}\cdot\overline{\mathbf{C}}\cdot\mathbf{D}=\mathbf{1}$, тоже имеет два решения: **A** – любое, $\mathbf{B}=\mathbf{C}=\mathbf{0}$, $\mathbf{D}=\mathbf{1}$. Среди этих четырех решений нет повторяющихся, поэтому исходное уравнение имеет 16 – 4 = 12 решений.

Обратите внимание, что число решений логических уравнений, в отличие от «обычных уравнений», всегда конечно. Это связано с тем, что каждая переменная может принимать только два значения (0 и 1), и число разных комбинаций значений переменных конечно, оно равно $2^n$, где **n** – это количество переменных. Поэтому уравнение с **n** переменными имеет не более $2^n$ решений.

## *Задачи*

1. Упростите логические выражения:

   а) $\mathbf{A}\cdot\mathbf{B}\cdot\overline{\mathbf{A}}\cdot\mathbf{B}+\mathbf{B}$  г) $\mathbf{A}+\overline{\mathbf{A}}\cdot\mathbf{B}+\overline{\mathbf{A}}\cdot\mathbf{C}$  ж) $(\overline{\mathbf{A}}+\mathbf{B})\cdot\overline{\mathbf{C}}\cdot(\mathbf{C}+\mathbf{A}\cdot\overline{\mathbf{B}})$

   б) $(\mathbf{A}+\mathbf{B})\cdot(\overline{\mathbf{A}}+\overline{\mathbf{B}})$  д) $\mathbf{A}\cdot(\mathbf{A}+\mathbf{B}+\mathbf{C})$  з) $\overline{\mathbf{A}}\cdot\overline{\mathbf{C}}+\mathbf{A}\cdot\mathbf{B}+\overline{\mathbf{A}}\cdot\mathbf{C}+\mathbf{A}\cdot\overline{\mathbf{B}}$

в) $A + A \cdot B + A \cdot C$ е) $A \cdot B + \overline{B} + \overline{A} \cdot B$ и) $A \cdot (\overline{B} \cdot \overline{C} + B \cdot C) + A \cdot (B \cdot \overline{C} + \overline{B} \cdot C)$

(Ответ: а – B, б – $A \cdot \overline{B} + B \cdot \overline{A}$, в – A, г – $A + B + C$, д – A, е – 1, ж – 0, з – 1, и – A)

2. Упростите логические выражения:

а) $A \cdot \overline{(\overline{B} + C)}$ г) $\overline{(A + \overline{B} + \overline{C})}$ ж) $(A + B + C) \cdot \overline{(\overline{A} \cdot \overline{B})} + C$

б) $\overline{(A + \overline{B})} + \overline{(A + B)} + A \cdot B$ д) $\overline{(A + B)} \cdot A \cdot \overline{B}$ з) $A \cdot \overline{(\overline{C} + \overline{B})} + \overline{(\overline{A} + B)} \cdot C + A \cdot C$

в) $A + \overline{(A + B)} + \overline{A} \cdot B$ е) $A + B \cdot \overline{C} + \overline{(\overline{A} + B + \overline{C})}$ и) $(A + B) \cdot (\overline{A} + B) \cdot (\overline{A} + \overline{B})$

(Ответ: а – $A \cdot B \cdot \overline{C}$, б – $\overline{A} + B$, в – 1, г – $\overline{A} \cdot B \cdot C$, д – 0, е – $A + \overline{B} + C$, ж – $A + B + C$, з – $A \cdot C$, и – $\overline{A} \cdot B$)

3. Упростите логические выражения:

а) $\overline{(A \to C)} \cdot C$ в) $A + \overline{(\overline{A} \to B)} + \overline{(A + \overline{B})}$ д) $\overline{(\overline{A} \to B)} \cdot \overline{(A \to B)}$

б) $\overline{(\overline{A} \to \overline{B})} + \overline{(\overline{A} \to B)} + A \cdot B$ г) $\overline{(\overline{A} \to (B \to \overline{C})}$ е) $A + \overline{B \cdot \overline{C}} + \overline{(A \to \overline{\overline{B} \cdot C})}$

(Ответ: а – 0, б – $\overline{A} + B$, в – 1, г – $\overline{A} \cdot B \cdot C$, д – 0, е – $A + \overline{B} + C$)

4. Решите уравнения

а) $A + \overline{B} + (B \to (C + D)) = 0$

б) $(A \to C) + B \cdot A + \overline{D} = 0$

в) $(\overline{A} + C) \to (\overline{B} + C + D) = 0$

г) $(A \to \overline{C}) + \overline{B} \cdot C \cdot A + D = 0$

д) $(\overline{(B + C)} \cdot A) \to (\overline{(A + C)} + D) = 0$

е) $(A \to C) \cdot (A \to \overline{C}) \cdot (\overline{A} \to (C \cdot \overline{B} \cdot D)) = 1$

(Ответ: а – 0100, б – 1001, в – 0100, г – 1110, д – 1100, е – 0011)

5. Сколько различных решений имеют уравнения

а) $A \cdot B + C \cdot D = 1$

б) $(A + B) \cdot (C + D) = 1$

в) $(A + B) \to (B \cdot C \cdot D) = 0$

г) $A \cdot \overline{B} \cdot C \cdot \overline{D} \cdot (E + \overline{E}) = 0$

д) $(A + B + C) \cdot \overline{B} \cdot \overline{C} \cdot D = 1$

е) $(A \cdot B \cdot C) \to (\overline{C} \cdot D) = 1$

ж) $(A \to B) \cdot C + \overline{C} \cdot D \cdot C = 1$

з) $(\overline{A} + \overline{B} + \overline{C}) \cdot (B + \overline{C} + \overline{D}) = 0$

(Ответ: а – 3, б – 7, в – 10, г – 30, д – 1, е – 14, ж – 6, з – 4)

## 3.5 Синтез логических выражений

До этого момента мы считали, что логическое выражение уже задано, и нам надо что–то с ним сделать (построить таблицу истинности, упростить и т.п.). Такие задачи называются задачами *анализа* (от греческого αναλυσις – разложение) – мы исследуем имеющееся выражение. При проектировании различных логических устройств, в том числе и узлов компьютеров, приходится решать обратную задачу – строить логическое выражение по готовой таблице истинности, которая описывает нужное правило обработки данных. Эта задача называется задачей *синтеза* (от греческого συνθεσις – совмещение).

В качестве простейшего примера построим логическое выражение для операции импликации $X = A \to B$.

| A | B | X | |
|---|---|---|---|
| 0 | 0 | 1 | • $\overline{A} \cdot \overline{B}$ |
| 0 | 1 | 1 | • $\overline{A} \cdot B$ |
| 1 | 0 | 0 | |
| 1 | 1 | 1 | • $A \cdot B$ |

*Способ 1*. В таблице истинности мы выделяем все строки, где логическое выражение равно единице. Тогда выражение может быть записано как логическая сумма выражений, каждое из которых истинно только в одном случае.

Например, выражение $\overline{A} \cdot \overline{B}$ истинно только при $A = 0$ и $B = 0$, то есть только в первой строке таблицы. Выражение $\overline{A} \cdot B$ истинно только во второй строке, а $A \cdot B$ – только в последней. Су-

ществует простое правило: если в этой строке переменная равна нулю, она входит в произведение с отрицанием, а если равна 1, то без отрицания.

Складывая выражения для всех отмеченных строк (кроме третьей, где функция равна нулю), получаем $\mathbf{X = \overline{A}\cdot\overline{B} + \overline{A}\cdot B + A\cdot B}$. Упрощаем это выражение:

$$\mathbf{X = \overline{A}\cdot(\overline{B}+B) + A\cdot B = \overline{A} + A\cdot B = (\overline{A}+A)\cdot(\overline{A}+B) = \overline{A}+B}.$$

Таким образом, мы вывели формулу, которая позволяет заменить импликацию через операции «НЕ» и «ИЛИ».

*Способ 2*. Если в таблице истинности нулей меньше, чем единиц, удобнее сначала найти формулу для обратного выражения, $\overline{\mathbf{X}}$, а потом применить операцию «НЕ». В данном случае выражение равно нулю в единственной строчке, при $\mathbf{A = 1}$ и $\mathbf{B = 0}$, то есть $\overline{\mathbf{X}} = \mathbf{A}\cdot\overline{\mathbf{B}}$. Теперь остается применить операцию «НЕ» и закон де Моргана:

$$\mathbf{X} = \overline{\mathbf{A}\cdot\overline{\mathbf{B}}} = \overline{\mathbf{A}} + \mathbf{B}.$$

Рассмотрим более сложный пример, когда выражение зависит от трех переменных. В этом случае в таблице истинности будет 8 строк.

| A | B | C | X | |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | $\bullet\ \overline{A}\cdot\overline{B}\cdot\overline{C}$ |
| 0 | 0 | 1 | 1 | $\bullet\ \overline{A}\cdot\overline{B}\cdot C$ |
| 0 | 1 | 0 | 1 | $\bullet\ \overline{A}\cdot B\cdot\overline{C}$ |
| 0 | 1 | 1 | 1 | $\bullet\ \overline{A}\cdot B\cdot C$ |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | $\bullet\ A\cdot\overline{B}\cdot C$ |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 1 | $\bullet\ A\cdot B\cdot C$ |

Отметим все строки, где $\mathbf{X = 1}$, и для каждой из них построим выражение, истинное только для этой комбинации переменных (см. таблицу). Теперь выполним логическое сложение:

$$\mathbf{X} = \overline{\mathbf{A}}\cdot\overline{\mathbf{B}}\cdot\overline{\mathbf{C}} + \overline{\mathbf{A}}\cdot\overline{\mathbf{B}}\cdot\mathbf{C} + \overline{\mathbf{A}}\cdot\mathbf{B}\cdot\overline{\mathbf{C}} + \overline{\mathbf{A}}\cdot\mathbf{B}\cdot\mathbf{C} + \mathbf{A}\cdot\overline{\mathbf{B}}\cdot\mathbf{C} + \mathbf{A}\cdot\mathbf{B}\cdot\mathbf{C}$$

Упрощение этого выражения дает

$$\mathbf{X} = \overline{\mathbf{A}}\cdot\overline{\mathbf{B}}\cdot(\overline{\mathbf{C}}+\mathbf{C}) + \overline{\mathbf{A}}\cdot\mathbf{B}\cdot(\overline{\mathbf{C}}+\mathbf{C}) + \mathbf{A}\cdot\mathbf{C}\cdot(\overline{\mathbf{B}}+\mathbf{B})$$

$$= \overline{\mathbf{A}}\cdot\overline{\mathbf{B}} + \overline{\mathbf{A}}\cdot\mathbf{B} + \mathbf{A}\cdot\mathbf{C} = \overline{\mathbf{A}}\cdot(\overline{\mathbf{B}}+\mathbf{B}) + \mathbf{A}\cdot\mathbf{C} = \overline{\mathbf{A}} + \mathbf{A}\cdot\mathbf{C} = (\overline{\mathbf{A}}+\mathbf{A})\cdot(\overline{\mathbf{A}}+\mathbf{C}) = \overline{\mathbf{A}}+\mathbf{C}.$$

Используя второй способ, получаем

$$\overline{\mathbf{X}} = \mathbf{A}\cdot\overline{\mathbf{B}}\cdot\overline{\mathbf{C}} + \mathbf{A}\cdot\mathbf{B}\cdot\overline{\mathbf{C}} = \mathbf{A}\cdot\overline{\mathbf{C}}\cdot(\overline{\mathbf{B}}+\mathbf{B}) = \mathbf{A}\cdot\overline{\mathbf{C}},$$

Тогда $\overline{\mathbf{X}} = \overline{\mathbf{A}\cdot\overline{\mathbf{C}}} = \overline{\mathbf{A}} + \mathbf{C}$. В данном случае второй способ оказался проще, потому что в таблице истинности меньше нулей, чем единиц.

*Способ 3.* При небольшом количестве нулей можно использовать еще один метод. Попробуем применить операцию «НЕ» к исходному выражению для $\overline{\mathbf{X}}$, без предварительного упрощения:

$$\mathbf{X} = \overline{\mathbf{A}\cdot\overline{\mathbf{B}}\cdot\overline{\mathbf{C}} + \mathbf{A}\cdot\mathbf{B}\cdot\overline{\mathbf{C}}}.$$

Применяя закон де Моргана, получим

$$\mathbf{X} = \overline{(\mathbf{A}\cdot\overline{\mathbf{B}}\cdot\overline{\mathbf{C}})}\cdot\overline{(\mathbf{A}\cdot\mathbf{B}\cdot\overline{\mathbf{C}})}.$$

Используя закон де Моргана еще два раза (для обеих скобок), находим

$$\mathbf{X} = (\overline{\mathbf{A}}+\mathbf{B}+\mathbf{C})\cdot(\overline{\mathbf{A}}+\overline{\mathbf{B}}+\mathbf{C}).$$

Заметим, что выражение в каждой скобке *ложно* только для одной комбинации исходных данных, при которых $\mathbf{X = 0}$. Таким образом, для каждой строчки в таблице истинности, где выражение равно 0, нужно построить логическую сумму, в которую переменные, равные в этой строчке единице, входят с инверсией, а равные нулю – без инверсии. Выражение для $\mathbf{X}$ – это произведение полученных сумм.

В нашем примере выражение упрощается с помощью распределительного закона для «И» и закона исключения третьего:

$$\mathbf{X} = (\overline{\mathbf{A}} + \mathbf{B} + \mathbf{C}) \cdot (\overline{\mathbf{A}} + \overline{\mathbf{B}} + \mathbf{C}) = (\overline{\mathbf{A}} + \mathbf{C}) + \mathbf{B} \cdot \overline{\mathbf{B}} = \overline{\mathbf{A}} + \mathbf{C}.$$

Неудивительно, что мы получили тот же ответ, что и раньше.

### *Задачи*

1. Постройте выражения для логических функций, заданных таблицами истинности. Используйте разные методы и сравните их.

а)

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

б)

| A | B | X |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

в)

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

г)

| A | B | X |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

(Ответ: а – $\mathbf{B}$, б – $\mathbf{A} + \overline{\mathbf{B}}$, в – $\overline{\mathbf{A}} \cdot \mathbf{B}$, г – $\mathbf{A} \cdot \overline{\mathbf{B}}$)

2. Постройте выражения для логических функций, заданных таблицами истинности. Используйте разные методы и сравните их.

а)

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

б)

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

в)

| A | B | C | X |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

(Ответ: а – $\overline{\mathbf{A}} \cdot \mathbf{C} + \mathbf{B} \cdot \overline{\mathbf{C}}$, б – $\mathbf{A} \cdot \mathbf{B} + \overline{\mathbf{B}} \cdot \overline{\mathbf{C}}$, в – $\overline{\mathbf{A}} \cdot \overline{\mathbf{B}} + \mathbf{B} \cdot \overline{\mathbf{C}}$)

## 3.6 Предикаты и кванторы

В предыдущих разделах мы видели, как алгебра логики позволяет нам записывать высказывания в виде формул и делать выводы. Однако с помощью алгебры логики невозможно доказать некоторые довольно простые утверждения. Рассмотрим такие высказывания:

**«Все люди смертны».**
**«Сократ – человек».**

Каждый из нас понимает, что если оба эти высказывания истинны, то Сократ тоже смертен. Однако алгебра высказываний не позволяет это доказать. В таких случаях приходится использовать более сложный математический аппарат, с которым мы познакомимся в этом параграфе.

В начале этой главы было сказано, что утверждение «*В городе N живут более 2 миллионов человек*» нельзя считать логическим высказыванием, поскольку непонятно, о каком городе идет речь. В этом предложении содержится некоторое утверждение, зависящее от *N*; если вместо *N* подставить название города, можно будет определить, истинно оно или ложно. Такое утверждение, зависящее от переменной, называют логической функцией или *предикатом*.

**Предикат** (от лат. *praedicatum* — заявленное, упомянутое, сказанное) — это утверждение, содержащее переменные.

Предикаты часто обозначаются буквой **P**, например,

**P(N) = «В городе N живут более 2 миллионов человек».**

Если мы задаем конкретные значения переменных, предикат превращается в логическое высказывание. Например, для предиката **P(N)** полученное высказывание будет истинно для **N** = «Москва» и ложно для **N** = «Якутск».

Предикат, зависящий от одной переменной – это свойство. Например, только что рассмотренный предикат **P(N)** характеризует свойство города. Вот еще примеры предикатов-свойств:

**Простое(x) = «x – простое число»**
**Студент(x) = «x – студент»**
**Спит(x) = «x всегда спит на уроке»**

Предикаты могут зависеть от нескольких переменных, например

**Больше(x,y) = «x больше y»**
**Живет(x,y) = «x живет в городе y»**
**Любит(x,y) = «x любит y»**

Это предикаты-отношения, они определяют связь между двумя объектами.

Предикаты нередко используются для того, чтобы задать множество, не перечисляя все его элементы. Так множество положительных чисел может быть задано предикатом, который принимает истинное значение для положительных чисел и ложное для остальных: $\mathbf{P(x) = (x > 0)}$. Множество пар чисел, сумма которых равна 1, задается предикатом $\mathbf{P(x,y) = (x + y = 1)}$, который зависит от двух переменных.

Существуют предикаты, которые справедливы для всех допустимых значений переменных. Например, $\mathbf{P(x) = (x^2 \geq 0)}$, определенный на множестве всех вещественных чисел. В таком случае используют запись $\forall \mathbf{x}\ \mathbf{P(x)}$, это означает «при любом **x** предикат **P(x)** справедлив». Знак $\forall$ – это буква «А», развернутая вверх ногами (от англ. *all* – все); он обозначает «любой», «всякий», «для любого», «для всех». Символ $\forall$ называют *квантором всеобщности*.

**Квантор** (от лат. *quantum* — сколько) — это знак или выражение, обозначающее количество.

Выражения «любой», «для всех» и т.п. также можно считать кванторами, они равносильны знаку $\forall$.

Кванторы широко применяются в математике. Например, для натуральных **n**

$$\forall \mathbf{n}: \quad 1 + 2 + \ldots + \mathbf{n} = \frac{\mathbf{n(n+1)}}{2}.$$

Часто используют еще один квантор – *квантор существования* $\exists$ (зеркальное отражение буквы Е, от англ. *exist* – существовать). Знак $\exists$ означает «существует», «хотя бы один». Например, если $\mathbf{P(x) = (x - 5 > 0)}$, то можно записать $\exists \mathbf{x}\ \mathbf{P(x)}$, что означает «существует **x**, такой что $\mathbf{x - 5 > 0}$». Это уже высказывание, а не предикат, потому что можно сразу установить его истинность. Запись $\forall \mathbf{x}\ \mathbf{P(x)}$ – это тоже высказывание, но оно ложно, потому что неравенство $\mathbf{x - 5 > 0}$ верно не для всех **x**.

Логическое выражение может включать несколько кванторов. Например, фразу «для любого **x** существует **y**, такой что **x + y = 0**» можно записать как $\forall \mathbf{x}\ \exists \mathbf{y}\ \mathbf{(x + y = 0)}$. Это утверждение истинно (на множестве чисел), потому что для любого **x** существует **(–x)**, число с обратным знаком. Переставлять местами кванторы нельзя, это меняет смысл выражения. Например, высказывание $\exists \mathbf{y}\ \forall \mathbf{x}\ \mathbf{(x + y = 0)}$ означает «существует такое значение **y**, что для любого **x** выполняется равенство $\mathbf{x + y = 0}$», это ложное высказывание.

Теперь давайте вернемся к Сократу, точнее, к двум высказываниям, приведенным в начале параграфа. Как записать утверждение «Все люди смертны»? Можно сказать иначе: «если **x** – человек, то **x** смертен», причем это верно для любого **x**. Вспоминаем, что связка «если…, то» записывается как импликация, а выражение «для любого **x**» – в виде квантора $\forall \mathbf{x}$. Поэтому получаем

$$\forall \mathbf{x}\ \mathbf{(P(x) \rightarrow Q(x))},$$

где **P(x) = «x – человек»**, **Q(x) = «x – смертен»**. Так как утверждение $\mathbf{P(x) \rightarrow Q(x)}$ верно для любого **x**, оно также верно при подстановке **x** = **Сократ**:

$$\mathbf{P(Сократ) \rightarrow Q(Сократ) = 1}.$$

Поскольку Сократ – человек, **P(Сократ) = 1**. Поэтому с помощью таблицы истинности для импликации мы находим, что **Q(Сократ) = 1**, то есть «Сократ смертен».

Если построить отрицание для высказывания с квантором $\forall$ или $\exists$, мы увидим, что один квантор заменяет другой. Например, отрицание высказывания $\forall x\, P(x)$ («неверно, что для любого **x** выполняется **P(x)**») звучит «существует такой **x**, для которого не выполняется **P(x)**» и может быть записано в виде $\exists x\, \overline{P(x)}$. Здесь, как и раньше, черта сверху обозначает отрицание. Таким образом,

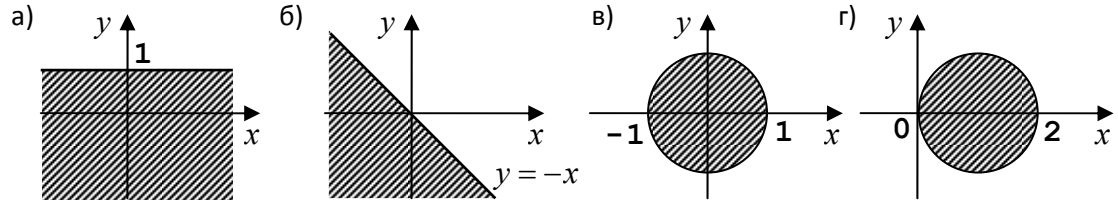
$$\overline{\forall x\, P(x)} = \exists x\, \overline{P(x)}.$$

Аналогично можно показать, что $\overline{\exists x\, P(x)} = \forall x\, \overline{P(x)}$.

Где можно использовать язык предикатов? Самая подходящая для этого область информатики – системы искусственного интеллекта, в которых моделируется человеческое мышление. Многие из них строятся на языке логического программирования Пролог, в котором программа представляет собой набор данных и правила вывода новых результатов из этих данных.

## *Задачи*

1. Какие из следующих предложений являются предикатами (здесь **x** и **y** – вещественные числа)?
   а) $x + y = 5$
   б) $\exists x\ (x + y = 5)$
   в) $\forall y\ \exists x\ (x + y = 5)$
   г) $\sin^2 x + \cos^2 x = 1$
   д) $x^2 + y^2 < 0$
   е) «x работает в ВУЗе»
   ж) $\forall x$ («x - студент»)
   з) $\exists x$ («x – учитель y»)
2. Задайте с помощью предикатов множества точек, соответствующие заштрихованным областям на плоскости:




3. Поставьте в начале каждого предложения одно из слов: «все» или «не все»:
   а) «... окуни – рыбы».
   б) «... рыбы умеют плавать».
   в) «... реки впадают в моря».
   г) «... моря соленые».
   д) «... числа четные».
   е) «... ломаные состоят из отрезков».
   ж) «... прямоугольники – квадраты».
   з) «... кошки – млекопитающие».
4. Запишите с помощью кванторов следующие утверждения:
   а) «Существует **x**, такой что $x > y$».
   б) «Не существует **x**, такой что $x > y$».
   в) «Для любого **x** имеем $x^2 > 1$».
   г) «Любая река впадает в Каспийское море».
   д) «Существует река, которая впадает в Каспийское море».
   е) «Для любой реки существует море, в которое она впадает».
   ж) «Для любого моря существует река, которая в него впадает».
   з) «Существует река, которая впадает во все моря».
   и) «Существует море, в которое впадают все реки».
5. *Запишите с помощью кванторов следующие утверждения:
   а) «Некоторые школьники ходят в театр».
   б) «Все кошки серые».
   в) «Встречаются злые собаки».
   г) «Все люди разные».
   д) «Люди ошибаются».

е) «Никто не обращает на него внимания».
ж) «Ни одна фирма не обанкротилась».
з) «Все лебеди – белые или черные».

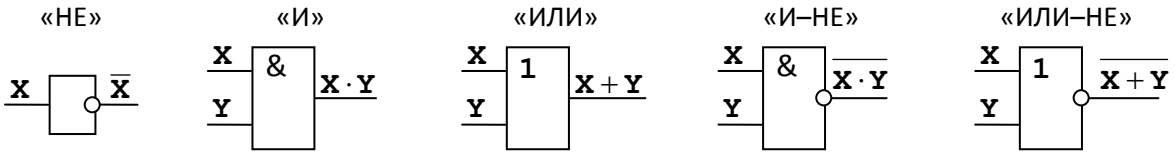
6. Запишите отрицание для следующих утверждений:

а) $\exists x\ (x^2 = 5)$
б) $\exists x\ (x + y = 5)$
в) $\forall y\ (x + y = 5)$
г) $\forall y\ \exists x\ (x + y = 5)$
г) «x работает в ВУЗе»
д) $\forall x$ («x - студент»)
е) $\exists x$ («x – учитель y»)
ж) $\exists x\ \forall y$ («x – учитель y»)

# 3.7 Логические элементы компьютера

## *3.7.1 Простейшие элементы*

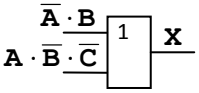
В компьютерах все вычисления выполняются с помощью *логических элементов* – электронных схем, выполняющих логические операции. Обозначения простейших элементов приводятся в таблице (ГОСТ 2.743-91). Заметьте, что небольшой кружок на выходе (или на входе) обозначает операцию «НЕ» (отрицание, инверсию).



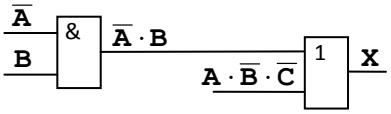

Может показаться, что для реализации сложных логических функций нужно много разных логических элементов. Однако, как мы видели в разд. 3.5, любую логическую функцию можно представить с помощью операций «НЕ», «И» и «ИЛИ» (такой набор элементов называется *полным*). Именно эта классическая «тройка» используется в книгах по логике, а также во всех языках программирования. Тем не менее, инженеры часто предпочитают строить логические схемы на основе элементов «ИЛИ-НЕ». Как показано в разд. 3.2.7, эта функция (штрих Шеффера) позволяет реализовать «НЕ», «И» и «ИЛИ», а значит и любую другую операцию.

Если нужно составить схему по известному логическому выражению, ее начинают строить с конца. Находят операцию, которая будет выполняться последней, и ставят на выходе соответствующий логический элемент. Затем повторяют то же самое для сигналов, поступающих на вход этого элемента. В конце концов, должны остаться только исходные сигналы – переменные в логическом выражении.

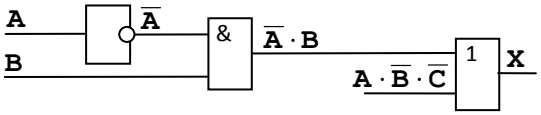
Составим схему, соответствующую выражению $X = \overline{A} \cdot B + A \cdot \overline{B} \cdot \overline{C}$. Последняя операция – это логическое сложение, поэтому на выходе схемы будет стоять элемент «ИЛИ»:




Для того, чтобы получить на первом входе $\overline{A} \cdot B$, нужно умножить $\overline{A}$ на B, поэтому добавляем элемент «И»:




Чтобы получить $\overline{A}$, ставим элемент «НЕ»:




Аналогично разбираем вторую ветку, которая поступает на второй вход элемента «ИЛИ»:

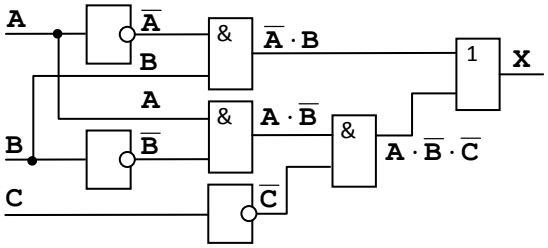


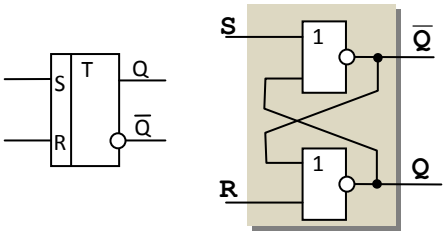

Схема составлена, ее входами являются исходные сигналы **A**, **B** и **C**, а выходом – **X**.

## *3.7.2 Триггер*

Слово *триггер* происходит от английского слова *trigger* – «защёлка» или спусковой крючок[3]. Так называют электронную схему, которая может находиться только в двух состояниях (их можно обозначить как 0 и 1), и способна почти мгновенно переходить из одного состояния в другое. Триггер изобрели независимо друг от друга М.А. Бонч-Бруевич и англичане У. Икклз и Ф. Джордан в 1918 году.

В компьютерах триггер используются для запоминания одного бита информации. Соответственно, для того, чтобы запомнить 1 байт информации требуется 8 триггеров, а для хранения 1 Кб – $8 \cdot 1024 = 8192$ триггера.

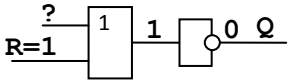
Триггеры бывают разных типов. Самый распространенный – это **RS–триггер**. Он имеет два входа, которые обозначаются как **S** (англ. *set* – установить) и **R** (англ. *reset* — сброс), и два выхода – **Q** и $\overline{\mathbf{Q}}$, причем выходной сигнал $\overline{\mathbf{Q}}$ является логическим отрицанием сигнала **Q** (если $\mathbf{Q = 1}$, то $\overline{\mathbf{Q}} = \mathbf{0}$, и наоборот). RS–триггер можно построить на двух элементах «И–НЕ» или на двух элементах «ИЛИ-НЕ». На следующем рисунке показано условное обозначение RS-триггера, внутреннее устройство триггера на элементах «ИЛИ–НЕ» и его таблица истинности.



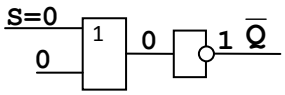

| S | R | Q | $\overline{\mathbf{Q}}$ | |
|---|---|---|---|---|
| 0 | 0 | Q | $\overline{\mathbf{Q}}$ | хранение бита |
| 0 | 1 | 0 | 1 | сброс в 0 |
| 1 | 0 | 1 | 0 | установка в 1 |
| 1 | 1 | 0 | 0 | запрещено |

Триггер использует так называемые *обратные связи* – сигналы с выходов схем «ИЛИ–НЕ» поступают на вход соседней схемы. Именно это позволяет хранить информацию.

Рассмотрим таблицу истинности триггера. Начнем с варианта, когда $\mathbf{S = 0}$ и $\mathbf{R = 1}$. Элемент «ИЛИ-НЕ» в нижней части схемы можно заменить на последовательное соединение элементов «ИЛИ» и «НЕ». Тогда, независимо от второго входа, на выходе «ИЛИ» будет 1, а на выходе «НЕ» - нуль. Это значит, что $\mathbf{Q = 0}$.



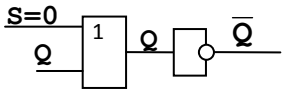

Тогда на входе другого элемента «ИЛИ-НЕ» будут два нуля, а на выходе $\overline{\mathbf{Q}}$ – единица.




Поскольку основным выходом считается **Q**, мы записали в триггер значение 0. Схема симметрична, поэтому легко догадаться, что при $\mathbf{S = 1}$ и $\mathbf{R = 0}$ мы запишем в триггер 1 ($\mathbf{Q = 1}$).

Теперь рассмотрим случай, когда $\mathbf{S = 0}$ и $\mathbf{R = 0}$. На выходе первого элемента «ИЛИ» будет сигнал $\mathbf{Q + 0 = Q}$, поэтому на выходе $\overline{\mathbf{Q}}$ останется его предыдущее значение:

[3] В английском языке триггер называется *flip-flop.*




Аналогично легко показать, что на выходе **Q** тоже остается его предыдущее значение. Это *режим хранения бита*. Для случая **S = 1** и **R =** 1 мы увидим, что оба выхода становятся равны нулю – в этом нет смысла, поэтому такой вариант запрещен.

Для хранения многоразрядных данных триггеры объединяются в единый блок, который называется *регистром*. Регистры (от 8 до 64 бит) используются во всех процессорах для временного хранения промежуточных результатов.

Над регистром, как над единым целым, можно производить ряд стандартных операций: сбрасывать (обнулять), заносить в него код и т.д. Часто регистры способны не просто хранить информацию, но и обрабатывать ее. Например, существуют регистры-счетчики, которые подсчитывают количество импульсов, поступающих на вход.

Триггеры применяются также в микросхемах быстродействующей оперативной памяти.

## *3.7.3 Сумматор*

Как следует из названия, сумматор предназначен для сложения (суммирования) двоичных чисел. Сначала рассмотрим более простой элемент, который называют *полусумматором*. Он выполняет сложение двух битов с учетом того, что в результате может получиться двухразрядное число (с переносом в следующий разряд).

Обозначим через **A** и **B** входы полусумматора, а через **P** и **S** – выходы (перенос в следующий разряд и бит, остающийся в текущем разряде). Таблица истинности этого устройства показана на рисунке. Легко увидеть, что столбец **P** – это результат операции «И», а столбец **S** – результат «исключающего ИЛИ»:

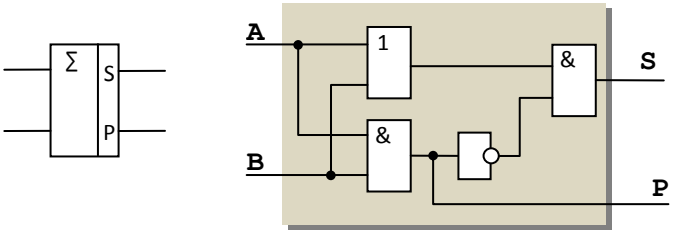
| A | B | P | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$$\mathbf{P} = \mathbf{A} \cdot \mathbf{B}, \quad \mathbf{S} = \mathbf{A} \oplus \mathbf{B} = \overline{\mathbf{A}} \cdot \mathbf{B} + \mathbf{A} \cdot \overline{\mathbf{B}}.$$

Формулу для **S** можно также записать в таком виде

$$\mathbf{S} = \overline{\mathbf{A}} \cdot \mathbf{B} + \mathbf{A} \cdot \overline{\mathbf{B}} = (\mathbf{A} + \mathbf{B}) \cdot (\overline{\mathbf{A}} + \overline{\mathbf{B}}) = (\mathbf{A} + \mathbf{B}) \cdot \overline{(\mathbf{A} \cdot \mathbf{B})} = (\mathbf{A} + \mathbf{B}) \cdot \mathbf{P},$$

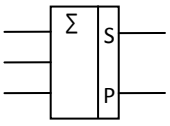
что позволяет построить полусумматор, используя всего 4 простейших элемента:




Слева показано условное обозначение полусумматора, греческая буква ∑ здесь (и в математике) обозначает сумму.

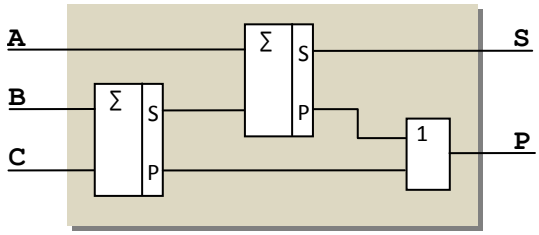
Полный **одноразрядный сумматор** учитывает также и третий бит – перенос из предыдущего разряда **C**. Сумматор имеет три входа и два выхода. Таблица истинности и обозначение сумматора показаны на рисунках.

| A | B | C | P | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |



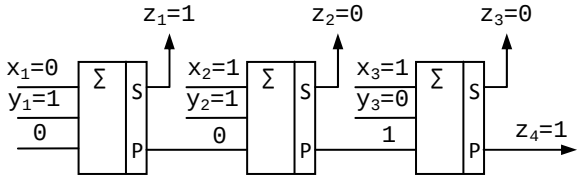

Логические функции для выходов сумматора вы можете найти самостоятельно.

Сумматор можно построить с помощью двух полусумматоров и одного элемента «ИЛИ»:




Сначала складываются биты B и C, а затем к результату добавляется бит A. Перенос на выходе сумматора появляется тогда, когда любое из двух промежуточных сложений дает перенос.

Для сложения многоразрядных чисел сумматоры объединяют в цепочку. При этом выход **P** одного сумматора (перенос в следующий разряд) соединяется с входом **C** следующего. На рисунке показано, как складываются два трехразрядных разрядных числа: $\mathbf{X} = 110_2$ и $\mathbf{Y} = 011_2$. Сумма $\mathbf{Z} = 1001_2$ состоит их четырех бит, поэтому на выходе последнего сумматора бит переноса будет равен 1.




Сложение начинается с самого младшего разряда. На вход первого сумматора подается младшие биты исходных чисел, $\mathbf{x}_1$ и $\mathbf{y}_1$ (см. рисунок), а на третий вход – ноль (нет переноса из предыдущего разряда). Выход **S** первого сумматора – это младший бит результата, $\mathbf{z}_1$, а его выход **P** (перенос) передается на вход второго сумматора и т.д. Выход **P** последнего из сумматоров представляет собой дополнительный разряд суммы, то есть $\mathbf{z}_4$.

Сумматор играет важную роль не только при сложении чисел, но при выполнении других арифметических действий. Фактически является основой арифметического устройства современного компьютера.

## ? *Контрольные вопросы*

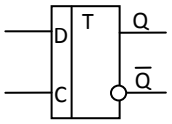
1. Что такое триггер? Объясните его принцип действия.
2. Почему для RS-триггера комбинация входов **S** = 1 и **R** = 1 запрещена?
3. Чем отличается одноразрядный сумматор от полусумматора?
4. Как можно построить сумматор с помощью двух полусумматоров?
5. Постройте логические выражения для выходов сумматора и нарисуйте соответствующие им схемы.
6. Объясните, как работает многоразрядный сумматор.
7. Что такое перенос? Как он используется в многоразрядном сумматоре?

## *Задачи*

1. Используя логические элементы, постройте схемы, соответствующие логическим выражениям $\mathbf{X}_1 = \overline{\mathbf{A}} \cdot \mathbf{C} + \mathbf{B} \cdot \overline{\mathbf{C}}$, $\mathbf{X}_2 = \mathbf{A} \cdot \mathbf{B} + \overline{\mathbf{B}} \cdot \overline{\mathbf{C}}$, $\mathbf{X}_3 = \overline{\mathbf{A}} \cdot \overline{\mathbf{B}} + \mathbf{B} \cdot \overline{\mathbf{C}}$.
2. Соревнования по поднятию тяжестей судит бригада из трех человек, один из них старший. Лампочка «Вес взят» должна зажигаться, если проголосовали, по крайней мере, два судьи, причем один из них – старший. Предложите логическую схему, которая решала бы эту задачу.
3. В двухэтажном коттедже есть два выключателя, которые управляют освещением лестницы, один из них – на первом этаже, а второй – на втором. Каждый выключатель имеет два состояния, при нажатии на кнопку состояние изменяется. В исходный момент оба выключателя выключены. Когда человек заходит в неосвещенное здание, он нажимает кнопку выключателя на первом этаже, при этом должна загореться лампочка, освещающая всю лестницу. Поднявшись на второй этаж, он нажимает на кнопку второго выключателя, и

лампочка должна погаснуть. Когда следом идет другой человек, он действует так же (хотя оба выключателя находятся в другом положении). Предложите логическую схему, которая решала бы эту задачу.

4. В самолете есть три бака с горючим. Бортовой компьютер получает сигналы от датчиков уровня в каждом баке: если горючего в баке достаточно, то сигнал равен 0, если горючее кончилось – 1. Когда горючее заканчивается, по крайней мере, в двух баках, должна загореться лампочка «Тревога». Предложите логическую схему, которая решала бы эту задачу.
5. В парламенте некоторой страны выбирают спикера из трех кандидатов. Каждый парламентарий должен нажать одну и только одну из трех кнопок. Если он проголосовал правильно (нажал ровно одну кнопку), на пульте должна загореться зеленая лампочка. Предложите логическую схему, которая решала бы эту задачу.
6. Постройте RS-триггер на элементах «И-НЕ» и составьте его таблицу истинности.
7. *Постройте таблицу истинности и логическую схему D-триггера, который запоминает сигнал на входе D (англ. *data* – данные) при подаче логической единицы на вход C (англ. *clock* – синхронизация). Для этого можно, например, немного изменить входную часть RS-триггера.




# 3.8 Логические задачи

## *3.8.1 Метод рассуждений*

***Задача 1.*** *Среди трех приятелей (их зовут Сеня, Вася и Миша) один всегда говорит правду, второй говорит правду через раз, а третий все время обманывает. Как-то раз они впервые прогуляли урок информатики. Директор школы вызвал их в свой кабинет для разговора. Сеня сказал: «Я всегда прогуливаю информатику. Не верьте тому, что скажет Вася». Вася сказал: «Я раньше не прогуливал этот предмет». Миша сказал: «Все, что говорит Сеня, – правда». Директору стало все понятно. Кто из них правдив, кто лгун, а кто говорит правду через раз?*

Для решения используем *метод рассуждений*. Во-первых, есть «точная» информация, которая не подвергается сомнению: все трое прогуляли урок информатики в первый раз.

Запишем высказывания мальчиков:

*Сеня*: 1. Я всегда прогуливаю информатику.
2. Вася сейчас соврет.

*Вася*: 1. Я раньше не прогуливал информатику.

*Миша*: 1. Сеня говорит правду.

Известно, что один из них говорит правду всегда, второй ¬– через раз, а третий все время лжет. Отметим, что если у нас есть только одно высказывание «полу-лжеца», оно может быть как истинным, так и ложным.

Сопоставив первое высказывание Сени и высказывание Васи с «точной» информацией, сразу определяем, что тут Сеня соврал, а Вася сказал правду. Это значит, что второе высказывание Сени тоже неверно, поэтому мальчик Сеня всегда лжет.

Тогда один из оставшихся, Вася или Миша, правдив, а второй говорит правду через раз. Мишино высказывание неверно, поскольку мы уже определили, что Сеня лжет; это значит, что Миша не всегда говорит правду, он – «полу-лжец». Тогда получается, что Вася правдив.

## *3.8.2 Табличный метод*

***Задача 2.*** *Перед началом турнира по шахматам болельщики высказали следующие предположения по поводу результатов:*

*А) Максим победит, Борис – второй;*
*Б) Борис – третий, Коля – первый;*
*В) Максим – последний, а первый – Дима.*

*Когда соревнования закончились, оказалось, что каждый из болельщиков был прав только в одном из своих прогнозов. Как распределились призовые места?*

Запишем высказывания трех болельщиков в форме таблицы (заголовок строки обозначает место в турнирной таблице). Будем считать, что каждое место занял ровно один участник.

| | А | Б | В |
|---|---|---|---|
| 1 | Максим? | Коля? | Дима? |
| 2 | Борис? | | |
| 3 | | Борис? | |
| 4 | | | Максим? |

Начнем «раскручивать» эту таблицу с той строчки, где больше всего информации, в данном случае – с первой. Предположим, что Максим действительно занял первое место, как и сказал болельщик «А». В этом случае «В» ошибся, поставив на первое место Диму. Тогда получается, что второй прогноз болельщика «В» верен, и Максим – последний.

Так как мы предполагали, что Максим занял первое место, получается противоречие. Следовательно, первый прогноз «А» не сбылся. Но тогда должен быть верен его второй прогноз, и Борис занял второе место. При этом он не мог занять еще и третье место, поэтому первый прогноз болельщика «Б» неверный, а верен его второй прогноз: Коля – первый.

| | А | Б | В |
|---|---|---|---|
| 1 | ~~Максим?~~ | Коля | ~~Дима?~~ |
| 2 | Борис | | |
| 3 | | ~~Борис?~~ | |
| 4 | | | Максим |

В этом случае Дима не может быть первым, поэтому верен первый прогноз «В»: Максим – последний. Диме осталось единственное свободное третье место. В результате места распределились так: I – Коля, II – Борис, III – Дима и IV – Максим.

***Задача 3.*** *На одной улице стоят в ряд 4 дома, в каждом из них живет по одному человеку. Их зовут Василий, Семен, Геннадий и Иван. Известно, что все они имеют разные профессии: скрипач, столяр, охотник и врач. Известно, что*

*(1) Столяр живет правее охотника.*
*(2) Врач живет левее охотника.*
*(3) Скрипач живет с краю.*
*(4) Скрипач живет рядом с врачом.*
*(5) Семен не скрипач и не живет рядом со скрипачом.*
*(6) Иван живет рядом с охотником.*
*(7) Василий живет правее врача.*
*(8) Василий живет через дом от Ивана.*
*Определите, кто где живет.*

Из условий (1) и (2) следует, что охотник живет не с краю, потому что справа от него живет столяр, а слева – врач. Скрипач по условию (3) живет с краю, он может жить как слева, так и справа от остальных:

| | | | | |
|---|---|---|---|---|
| скрипач? | врач | охотник | столяр | скрипач? |

Согласно условию (4), скрипач живет рядом с врачом, поэтому он занимает крайний дом слева:

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| скрипач | врач | охотник | столяр |

Профессии жильцов определили, остается разобраться с именами. Из условия (5) «*Семен не скрипач и не живет рядом со скрипачом*» следует, что Семен – охотник или столяр:

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| скрипач | врач | охотник | столяр |
| | | Семен? | Семен? |

Из условия (6) «*Иван живет рядом с охотником*» следует, что он – врач или столяр:

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| скрипач | врач | охотник | столяр |
| | | Семен? | Семен? |
| | Иван? | | Иван? |

Из условия (7) *«Василий живет правее врача»* определяем, что Василий – охотник или столяр:

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| скрипач | врач | охотник | столяр |
| | | Семен? | Семен? |
| | Иван? | | Иван? |
| | | Василий? | Василий? |

Согласно условию (8), *«Василий живет через дом от Ивана»*, поэтому Иван – врач, а Василий – столяр:

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| скрипач | врач | охотник | столяр |
| | Иван | Семен? | Василий |

Тогда сразу получается, что Семен – охотник, а Геннадий должен занять оставшееся свободное место, он – скрипач:

| 1 | 2 | 3 | 4 |
|---|---|---|---|
| скрипач | врач | охотник | столяр |
| Геннадий | Иван | Семен | Василий |

***Задача 3.*** *Шесть приятелей, Саша, Петя, Витя, Дима, Миша и Кирилл, встретившись через 10 лет после окончания школы, выяснили, что двое из них живут в Москве, двое – в Санкт-Петербурге, а двое – в Перми. Известно, что*

*(1) Витя ездит в гости к родственникам в Москву и Санкт-Петербург.*

*(2) Петя старше Саши.*

*(3) Дима и Миша летом были в Перми в командировке.*

*(4) Кирилл и Саша закончили университет в Санкт-Петербурге и уехали в другие города.*

*(5) Самый молодой из них живет в Москве.*

*(6) Кирилл редко приезжает в Москву.*

*(7) Витя и Дима часто бывают в Санкт-Петербурге по работе.*

*Определите, кто где живет.*

Составим таблицу, где каждая строка соответствует городу, а столбец – человеку:

| | Саша | Петя | Витя | Дима | Миша | Кирилл |
|---|---|---|---|---|---|---|
| Москва | | | | | | |
| Санкт-Петербург | | | | | | |
| Пермь | | | | | | |

Единица в таблице будет обозначать, что человек живет в данном городе, а ноль – что точно не живет. По условию в каждом городе живут ровно 2 человека, каждый живет только в одном городе. Поэтому в каждой строке должно быть две единицы, а в каждом столбце – одна.

Из условия (1) следует, что Витя живет в Перми:

| | Саша | Петя | Витя | Дима | Миша | Кирилл |
|---|---|---|---|---|---|---|
| Москва | | | 0 | | | |
| Санкт-Петербург | | | 0 | | | |
| Пермь | | | 1 | | | |

Из (2) и (5) находим, что Петя живет не в Москве. Кроме того, как следует из (6), Кирилл – тоже не москвич.

| | Саша | Петя | Витя | Дима | Миша | Кирилл |
|---|---|---|---|---|---|---|
| Москва | | 0 | 0 | | | 0 |
| Санкт-Петербург | | | 0 | | | |
| Пермь | | | 1 | | | |

Согласно (3), Дима и Миша живут не в Перми:

| | Саша | Петя | Витя | Дима | Миша | Кирилл |
|---|---|---|---|---|---|---|
| Москва | | 0 | 0 | | | 0 |
| Санкт-Петербург | | | 0 | | | |
| Пермь | | | 1 | 0 | 0 | |

Из условия (4) делаем вывод, что Кирилл и Саша живут не в Санкт-Петербурге, отсюда сразу следует, что Кирилл живет в Перми. Двух пермяков мы уже определили, поэтому Саша и Петя живут не в Перми:

| | Саша | Петя | Витя | Дима | Миша | Кирилл |
|---|---|---|---|---|---|---|
| Москва | | 0 | 0 | | | 0 |
| Санкт-Петербург | 0 | | 0 | | | 0 |
| Пермь | 0 | 0 | 1 | 0 | 0 | 1 |

Далее находим, что Саша – москвич, а Петя живет в Санкт-Петербурге.

| | Саша | Петя | Витя | Дима | Миша | Кирилл |
|---|---|---|---|---|---|---|
| Москва | 1 | 0 | 0 | | | 0 |
| Санкт-Петербург | 0 | 1 | 0 | | | 0 |
| Пермь | 0 | 0 | 1 | 0 | 0 | 1 |

По условию (7) Витя и Дима – не петербуржцы, поэтому в Петербурге живет Миша, а Дима – в Москве:

| | Саша | Петя | Витя | Дима | Миша | Кирилл |
|---|---|---|---|---|---|---|
| Москва | 1 | 0 | 0 | 1 | 0 | 0 |
| Санкт-Петербург | 0 | 1 | 0 | 0 | 1 | 0 |
| Пермь | 0 | 0 | 1 | 0 | 0 | 1 |

Таким образом, Саша и Дима живут в Москве, Петя и Миша – в Санкт-Петербурге, а Витя и Кирилл – в Перми.

## *3.8.3 Использование алгебры логики*

Когда в условии задачи встречаются сложные логические высказывания, удобно использовать методы алгебры логики. Покажем этот подход на примерах.

***Задача 4.*** *Следующие два высказывания истинны:*

*1. Неверно, что если корабль А вышел в море, то корабль С – нет.*

*2. В море вышел корабль В или корабль С, но не оба вместе.*

*Определить, какие корабли вышли в море.*

Введем три высказывания: $\mathbf{A}$ – корабль А вышел в море; $\mathbf{B}$ – корабль В вышел в море; $\mathbf{C}$ – корабль С вышел в море. Вспомним, что связка «если…, то» в логических выражениях заменяется импликацией, поэтому фразу «*если корабль А вышел в море, то корабль С – нет*» можно записать как $\mathbf{A} \to \overline{\mathbf{C}} = \mathbf{1}$. Но в условии сказано, что это утверждение неверно, поэтому

$$\mathbf{A} \to \overline{\mathbf{C}} = \mathbf{0} \text{ или } \overline{\mathbf{A} \to \overline{\mathbf{C}}} = \mathbf{1}.$$

Второе условие – это «исключающее ИЛИ», то есть $\mathbf{B} \oplus \mathbf{C} = \mathbf{1}$. Оба условия истинны одновременно, то есть их логическое произведение («И») тоже истинно:

$$\overline{(\mathbf{A} \to \overline{\mathbf{C}})} \cdot (\mathbf{B} \oplus \mathbf{C}) = \mathbf{1}.$$

Нам нужно решить это уравнение и найти неизвестные **A**, **B** и **C**. Для этого выразим импликацию и «исключающее ИЛИ» через базовый набор логических операций («НЕ», «И», «ИЛИ»), а затем раскроем инверсию сложного выражения с помощью закона де Моргана:

$$\overline{(A \to \overline{C})} \cdot (B \oplus C) = (\overline{\overline{A} + \overline{C}}) \cdot (B \cdot \overline{C} + \overline{B} \cdot C) = A \cdot C \cdot (B \cdot \overline{C} + \overline{B} \cdot C) = 1 .$$

В последнем выражении раскроем скобки и учтем, что $C \cdot \overline{C} = 0$ и $C \cdot C = C$. Получим

$$A \cdot \overline{B} \cdot C = 1 .$$

Это уравнение имеет единственное решение: $A = 1$, $B = 0$ и $C = 1$. Это значит, что в море вышли корабли **A** и **C**.

***Задача 5****. На вопрос «Кто из твоих учеников изучал логику?» учитель ответил: «Если логику изучал Андрей, то изучал и Борис. Однако неверно, что если изучал Семен, то изучал и Борис». Кто же изучал логику?*

Обозначим буквами высказывания: **A** – логику изучал Андрей; **B** – логику изучал Борис и **C** – логику изучал Семен. Оба высказывания учителя можно записать в виде импликаций

*«Если логику изучал Андрей, то изучал и Борис».* $A \to B = 1$

*«Неверно, что если изучал Семен, то изучал и Борис».* $C \to B = 0$

Дальше есть два варианта решения. Во-первых, можно поступить так же, как и в предыдущей задаче: применить операцию «НЕ» ко второму высказыванию и составить уравнение с помощью логического произведения:

$$(A \to B) \cdot \overline{(C \to B)} = 1 .$$

Теперь представляем импликацию через базовые операции и применяем закон де Моргана

$$(\overline{A} + B) \cdot (\overline{\overline{C} + B}) = (\overline{A} + B) \cdot C \cdot \overline{B} = \overline{A} \cdot C \cdot \overline{B} = 1 .$$

Это уравнение имеет единственное решение: $A = 0$, $B = 0$ и $C = 1$. Значит, логику изучал только Семен.

Можно поступить иначе, вспомнив, что импликация ложна только в том случае, когда первое высказывание истинно, а второе ложно. Поэтому из условия $C \to B = 0$ сразу следует, что $B = 0$ и $C = 1$. Тогда первое условие, $A \to B = A \to 0 = 1$ сразу дает $A = 0$.

## *Задачи*

1. Три школьника, Миша, Коля и Сергей, остававшиеся в классе на перемене, были вызваны к директору по поводу разбитого в это время окна в кабинете. На вопрос директора о том, кто это сделал, мальчики ответили следующее:

   *Миша: "Я не бил окно, и Коля тоже…"*

   *Коля: "Миша не разбивал окно, это Сергей разбил футбольным мячом!"*

   *Сергей: "Я не делал этого, стекло разбил Миша".*

   Выяснилось, что один из ребят сказал чистую правду, второй в одной части заявления соврал, а другое его высказывание истинно, а третий оба раза соврал. Кто разбил стекло в классе?

   (Ответ: Миша)
2. В финал соревнований по настольному теннису вышли Наташа, Маша, Люда и Рита. Болельщики высказали свои предположения о распределении мест в дальнейших состязаниях. Один считает, что первой будет Наташа, а Маша будет второй. Другой болельщик на второе место прочит Люду, а Рита, по его мнению, займет четвертое место. Третий считает, что Рита займет третье место, а Наташа будет второй. Когда соревнования закончились, оказалось, что каждый из болельщиков был прав только в одном из своих прогнозов. Как распределились места?

   (Ответ: I – Наташа, II – Люда, III – Рита, IV – Маша)
3. На одной улице стоят в ряд 4 дома, в каждом из них живет по одному человеку. Их зовут Алексей, Егор, Виктор и Михаил. Известно, что все они имеют разные профессии: рыбак, пчеловод, фермер и ветеринар. Известно, что

   *(1) Фермер живет правее пчеловода.*

   *(2) Рыбак живет правее фермера.*

*(3) Ветеринар живет рядом с рыбаком.*
*(4) Рыбак живет через дом от пчеловода.*
*(5) Алексей живет правее фермера.*
*(6) Виктор – не пчеловод.*
*(7) Егор живет рядом с рыбаком.*
*(8) Виктор живет правее Алексея.*

Определите, кто где живет.
(Ответ: пчеловод Михаил, фермер Егор, рыбак Алексей, ветеринар Виктор)

4. Дочерей Василия Лоханкина зовут Даша, Анфиса и Лариса. У них разные профессии и они живут в разных городах: одна в Ростове, вторая – в Париже и третья – в Москве. Известно, что
   *(1) Даша живет не в Париже, а Лариса – не в Ростове.*
   *(2) Парижанка – не актриса.*
   *(3) В Ростове живет певица.*
   *(4) Лариса – не балерина.*
   Определите, где живет каждая из дочерей и чем занимается.
   (Ответ: Даша – певица, Ростов; Анфиса – балерина, Париж; Лариса – актриса, Москва)
5. В состав экспедиции входят Михаил, Сергей и Виктор. На обсуждении распределения обязанностей с руководителем проекта были высказаны предположения, что командиром будет назначен Михаил, Сергей не будет механиком, а Виктор будет утвержден радистом, но командиром не будет. Позже выяснилось, что только одно из этих четырех утверждений оказалось верным. Как распределились должности?
   (Ответ: Виктор – командир, Михаил – механик, Сергей – радист)
6. В ходе заседания суда выяснилось, что:
   *(1) Если Аськин не виновен или Баськин виновен, то виновен Сенькин.*
   *(2) Если Аськин не виновен, то Сенькин не виновен.*
   Виновен ли Аськин?
   (Ответ: виновен)
7. Аськин, Баськин и Васькин стали свидетелями ограбления банка. Во время расследования Аськин сказал, что взломщики приехали на синей «Тойоте». Баськин считает, что это был красный «BMW», а Васькин утверждает, что это был «Форд-Фокус», но не синий. Выяснилось, что каждый из них назвал неправильно либо марку, либо цвет машины. На каком автомобиле приехали преступники?
   (Ответ: красная «Тойота»)

# Глава 5. Устройство компьютера

**Компьютер** – это программируемый автомат для обработки данных.

Из этого определения можно сделать вывод, что компьютер состоит из двух важнейших составляющих: *аппаратной части* и *программного обеспечения* (ПО). В технической литературе их часто называют английскими терминами *hardware* и *software*[1].

Поскольку одно и то же оборудование может быть перенастроено на выполнение новых задач простой заменой ПО, такие универсальные компьютеры можно выпускать большими партиями, и это делает их производство проще и дешевле. За счет этого во многих областях они заменили специализированные устройства.

Исторически существовало два принципиально разных типа вычислительных машин – *аналоговые* и *цифровые*. Они различались по способу представления обрабатываемых данных: в аналоговой или цифровой форме. Цифровая техника быстро доказала свои преимущества:

- высокую точность вычислений;
- универсальность и быстроту перехода от одной задачи к другой;
- способность хранить большие объемы данных.

В результате почти все современные компьютеры работают только с дискретной (цифровой) информацией. Поэтому в этой главе рассматривается только цифровая вычислительная техника.

## 5.1. История развития вычислительной техники

История появления и развития вычислительной техники – это обширная тема, которой посвящено множество книг. С точки зрения курса информатики исторические сведения интересны, прежде всего, тем, что позволяют отследить основные направления развития компьютерной техники и попытаться предвидеть ее ближайшее будущее.

В отечественной технической литературе приблизительно до 80-х годов прошлого века везде использовался термин *электронно-вычислительная машина* (ЭВМ). Позднее это словосочетание стало постепенно вытесняться новым более коротким названием *компьютер*. Все разновидности современной вычислительной техники сейчас называются только компьютерами, но, тем не менее, старые модели по традиции именуются ЭВМ.

### 5.1.1. Вехи истории

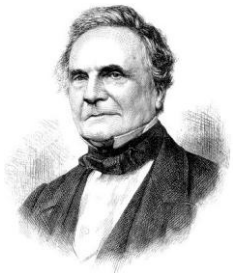
Блез Паскаль (1623-1662)
(*www.thocp.net*)

Первая механическая машина, с помощью которой можно было производить вычисления, была изготовлена известным французским ученым *Блезом Паскалем* (1623-1662) в 1645 году. Чтобы отдать дань уважения этому изобретению, один из языков программирования впоследствии был назван именем *Паскаль*.

Идея о реализации вычислений в автоматическом (без участия человека) режиме впервые была предложена и детально развита английским ученым *Чарльзом Бэббиджем* (1791-1871). Он спроектировал и описал аналитическую машину, состав и принципы действия которой фактически повторились в будущих ЭВМ.

[1] Дословный перевод этих терминов затруднителен: ware – это изделия, hardware изначально – металлические изделия, скобяные товары, а применительно к компьютеру – его детали (платы, монитор и прочее «железо»); термин *software* исключительно компьютерный, возник как противопоставление слов soft (мягкий, гибкий, податливый) и *hard* (твердый, жесткий, негнущийся), т.е. *software* гибко «подстраивает» *hardware* к решению разнообразных задач

Бэббидж посвятил всю свою жизнь работе над машиной, но построить ее из механических деталей не удалось: уровень техники XIX века не позволял изготовить столь сложный и точный механизм.

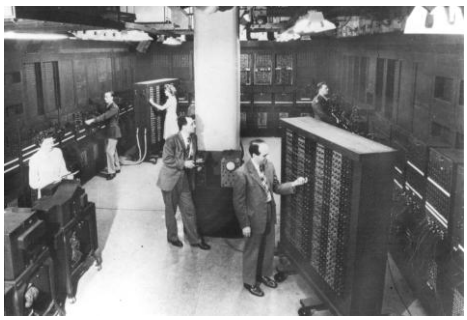


Чарльз Бэббидж (1791-1871)
(*ru.wikipedia.org*)


Ада Лавлейс (1815-1852)
(*www.science20.com*)

Программированием для аналитической машины Бэббиджа занималась *Ада Лавлейс* (дочь английского поэта Д.Г. Байрона, 1815-1852). Ее идеи оказали большое влияние на развитие программирования. Например, ей принадлежат термины «цикл» и «рабочая ячейка». В честь первого в мире программиста один из языков программирования получил имя *Ада*.

Первой ЭВМ[2], продемонстрировавшей на практике возможность автоматических расчетов по программе, считается ЭНИАК (сокращение от английского словосочетания *Electronic Numeric Integrator and Computer*). Он был построен в 1944 году в США под руководством *Джона Моучли*; главным инженером проекта был *Преспер Эккерт*. ЭНИАК содержал 18000 электронных ламп и, занимая зал 9×15 м$^2$, потреблял около 150 кВт электроэнергии; он выполнял более 350 умножений и 5000 сложений за секунду. Данные вводились в машину с помощью перфокарт, а программа обработки набиралась с помощью штекеров на специальных панелях.



Компьютер «ЭНИАК»
(*www.fi.edu*)

Джон фон Нейман (1903-1957)
(*startupgallery.org*)

Опыт построения первой ЭВМ был проанализирован *А. Берксом*, *Г. Голдстайном* и *Д. фон Нейманом*. В 1946 году они опубликовали работу «Предварительное рассмотрение логической конструкции электронного вычислительного устройства», ставшую классической. Сформулированные в ней принципы построения вычислительных машин используются и сейчас, несмотря на то, что со времени публикации прошло более полувека. В компьютерной литературе эти принципы часто называют *фон-неймановской архитектурой* ЭВМ, хотя Джон фон Нейман не был ее единоличным автором.

Дальнейший прогресс вычислительной техники во многом определялся развитием ее элементной базы. Важной вехой на этом пути стало создание в 1947 году *транзистора* (*У. Шокли*, *Д. Бардин* и *У. Браттейн*). *Транзистор* – это полупроводниковый прибор для управления электрическими сигналами. На основе транзисторов могут собираться цифровые электронные схемы, такие как логические элементы и триггеры.

Транзисторы
(*ru.wikipedia.org*)

В 1958 году *Дж. Килби* разработал первую *интегральную*

[2] Вопросы приоритета в конструировании самой первой вычислительной машины до сих пор являются предметом непрекращающихся споров, в том числе и судебных.

*микросхему* – кристалл, в котором размещается не один транзистор, а целая схема на нескольких транзисторах – например, один или несколько триггеров. Все ее вспомогательные радиодетали (резисторы, конденсаторы и другие) также изготовляются с помощью полупроводниковых технологий.

Первый *микропроцессор* Intel 4004 был разработан под руководством инженера *М. Хоффа* и выпущен в 1971 году. Он был четырехбитным, поскольку предназначался для микрокалькуляторов и должен был производить вычисления над десятичными числами (см. разд. 2.10 о двоично-десятичном кодировании). Интересно, что цель его создания была чисто технологическая: заменить 12 специализированных микросхем одной универсальной, но программируемой. Достоинства микропроцессора как главного узла для компьютеров были оценены позднее.

С. Джобс и С. Возняк с компьютером Apple-I (*cedmagic.com*)

В ходе совершенствования элементной базы вычислительные машины становились все более мощными и компактными. В результате появились персональные компьютеры (ПК), которыми мы сейчас пользуемся. В 1976 году два молодых приятеля *С. Джобс* и *С. Возняк* в гараже родителей Джобса собрали ПК *Apple*, положивший начало известному ныне семейству компьютеров. А в 1981 году был продемонстрирован первый компьютер другого семейства – IBM PC (*IBM Personal Computer*), потомки которого в нашей стране особенно широко распространены.

## 5.1.2. Поколения ЭВМ (совершенствование элементной базы)

Неудачная попытка Ч. Бэббиджа построить механическую аналитическую машину показала, насколько важную роль играет элементная база. Именно поэтому дальнейшую историю вычислительной техники принято делить на периоды в соответствии с теми элементами, из которых изготовлялись машины.

*Первое поколение* ЭВМ относят к периоду примерно 1945-1955 годов. Эти машины были построены на базе электронных ламп[3]. Открыл его уже описанный ранее *ЭНИАК*. В нашей стране машинами первого поколения были *МЭСМ* (малая электронная счетная машина, 1951 год), *БЭСМ* (большая электронная счетная машина, 1952 год), *Стрела* (1953 год), *Урал* (1954 год), *М-20* (1959 год). Все эти машины были огромными, неудобными и дорогими.

ЭВМ первого поколения *МЭСМ* (фото из Единой коллекции цифровых ресурсов)

*Второе поколение* (примерно 1955-1965 годы) появилось, когда на смену лампам в электронных схемах пришли транзисторы. Первый экспериментальный компьютер на транзисторах *ТХ-0* был создан в 1955 году в Массачусетском технологическом институте

[3] В середине XX века было разработано несколько счетных машин на электромагнитных реле, но их из-за малого количества не принято включать в классификацию поколений.

(США). ЭВМ на транзисторах были значительно меньше и имели существенно более высокое быстродействие; они потребляли гораздо меньше энергии, были надежнее и не требовали таких громоздких систем отвода тепла, как ламповые машины. Многие машины второго поколения уже помещались в обычной комнате среднего размера, например, ЭВМ серии *Наири* (1964 год) или *МИР* (машина инженерных расчетов, 1965 год). Наиболее производительными ЭВМ этого поколения стали *Стретч* (США, 1960 год), *Атлас* (Великобритания, 1961 год), *CDC 6600* (США, 1964 год) и *БЭСМ-6* (СССР, 1967 год).

ЭВМ второго поколения БЭСМ-6
(фото из Единой коллекции цифровых ресурсов)

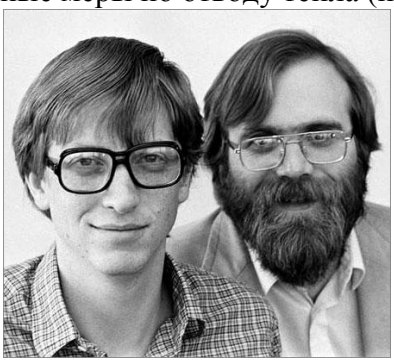
*Третье поколение* ЭВМ (примерно 1965-1975 годы) связано с появлением интегральных микросхем. Казалось бы, размеры этих ЭВМ снова должны существенно уменьшиться, но этого не произошло. Дело в том, ЭВМ третьего поколения были предназначены для коллективной (многопользовательской) работы. Это было время крупных вычислительных центров, предоставлявших услуги огромному числу пользователей из многих организаций. Поэтому главное внимание уделялось не уменьшению размеров и стоимости машин, а повышению их вычислительной мощности и эффективности обработки больших объемов данных.

Уменьшение размеров ячеек ЭВМ первого – третьего поколений (в каждой из них – по два триггера)

Отличительная черта третьего поколения – выпуск *семейств* вычислительных машин, которые совместимы между собой как аппаратно (все устройства сконструированы по одинаковым стандартам), так и программно (имеют одинаковую систему команд). Впервые идею общей *архитектуры*, обеспечивающей выполнение написанных ранее программ на любой новой модели, предложила фирма IBM, которая разработала семейства больших ЭВМ *IBM/360* и *IBM/370*. В этот период в СССР было принято решение перейти к копированию зарубежной техники ради обеспечения совместимости. В результате в странах Восточной Европы были выпущены «аналоги» упомянутых выше семейств под общим названием *ЕС ЭВМ* (Единая система ЭВМ). Одновременно появились мини-ЭВМ семейства *СМ ЭВМ* (Семейство малых ЭВМ), аналогичное известному зарубежному семейству *PDP* фирмы *DEC*.

*Четвертое поколение* берет свое начало примерно с 1975 года. Прогресс в электронике дал возможность существенно увеличить плотности «упаковки» элементов на кристалле, и в одной микросхеме теперь удавалось собрать целый узел, например, *микропроцессор*. Микросхемы такого уровня стали называть БИС (большие интегральные схемы, от 1000 до 10000 элементов на кристалле), а позднее – СБИС (сверхбольшие ИС, более 10000 элементов). Именно они стали основой четвертого поколения ЭВМ, которое существует вплоть до настоящего времени.

Увеличение плотности схемы позволило, в первую очередь, повысить быстродействие компьютеров[4]. Кроме того, возросла и надежность, поскольку значительная часть электрических соединений выполнена внутри кристалла. Однако при высокой плотности монтажа ухудшается теплоотдача от миниатюрных деталей, поэтому требуются специальные меры по отводу тепла (например, установка вентиляторов охлаждения).



Б. Гейтс и П. Аллен (chernykh.net)

Первый восьмиразрядный процессор *Intel 8080*, предназначенный специально для компьютеров, был выпущен в 1974 году. На его базе был разработан микрокомпьютер *Альтаир*, имевший большой коммерческий успех. Он вошел в историю еще и потому, что в 1975 году молодой студент Билл Гейтс со своим другом Полом Алленом реализовали на нем язык программирования BASIC. Чуть позднее они создали известную компанию *Microsoft*.

Кроме персональных компьютеров, к четвертому поколению относятся *серверы* – мощные вычислительные машины, которые используются для управления компьютерными сетями. Они предоставляют свои ресурсы (например, принтеры, файлы или программы) в коллективное пользование. Серверы могут эффективно обслуживать большое количество пользователей одновременно. Например, два сервера *Hewlett-Packard T600* (по 12 процессоров в каждом), установленные в системе резервирования билетов *Amadeus*, способны практически без задержек обслуживать примерно 60 миллионов запросов в сутки (система имеет около 180 тысяч терминалов в более чем ста странах мира)[5].

Важное направление в компьютерах четвертого поколения – *параллельная* (одновременная) обработка данных. Если решаемую задачу удается разбить на независимые друг от друга действия, то их не обязательно делать друг за другом, а можно для экономии времени выполнять одновременно. Правда, для этого требуется несколько процессоров, но современный уровень техники это позволяет. Более того, в последнее время были сконструированы многоядерные процессоры, т.е. фактически несколько процессоров в одном кристалле.

Мощные многопроцессорные компьютеры, в которых выполняется параллельная обработка данных, называют *суперкомпьютерами.* Это уникальные устройства, поэтому они изготавливаются штучно.

В литературе часто упоминаются суперкомпьютеры серии *CRAY*, разработанные под руководством *Сеймура Крэя*. Первая модель этой серии, *CRAY-1*, была построена в США в 1976 году и имела огромный коммерческий успех.

Все развитые страны ведут жесткую конкуренцию в области суперкомпьютеров, поскольку обладание такой техникой позволяет решать стратегически важные вычислительные задачи:

- исследование геофизики Земли, прогнозирование изменений климата на планете;
- создание математических моделей молекул (полимеров, кристаллов и т.п.), синтез новых материалов и лекарств;
- расчет процессов горения и взрыва, а также моделирование других физических задач (обтекание летательных аппаратов, прочность кузовов автомобилей);
- расчеты процессов нефте- и газодобычи, а также сейсморазведки недр;
- проектирование новых электронных устройств.

Приведем несколько примеров применения суперкомпьютеров. Исследователи фирмы *IBM* на протяжении десятилетий изучают деятельность мозга и пытаются моделиро-

[4] При быстродействии $10^9$ элементарных операций в секунду (типичное по порядку величины значение для современного компьютера) за время каждой из них электрический сигнал со скоростью $3 \cdot 10^8$ м/с успевает пройти путь всего 30 см.

[5] http://parallel.ru/vvv/lec1.html

вать ее. В 2009 году появилось сообщение[6] о том, что полученная в рамках проекта модель мозга по своим возможностям превзошла уровень кошки: моделируется 1 миллиард нейронов и 10 триллионов связей между ними! Модель работает на базе суперкомпьютера *Blue Gene/P*, имеющего 147 456 процессоров и 144 Тбайт оперативной памяти.

По данным компании *Ford Motor*, благодаря детальному моделированию на суперкомпьютере, количество разрушаемых в крэш-тестах[7] автомобилей удается сократить на треть.[8]

Применение суперкомпьютеров для расчета состава лекарств позволяет уменьшить срок их разработки с нескольких лет до полугода.

Мощные компьютеры используются при создании компьютерных спецэффектов в кино. Например, в фильме «Властелин колец» фирме *WETA Digital* потребовалось столько суперкомпьютеров, что Новая Зеландия вышла на первое место в мире по их количеству на душу населения.[9]


Суперкомпьютер Ломоносов

В 2009 году в МГУ введен в строй самый мощный российский суперкомпьютер Ломоносов производительностью около 400 Тфлопс[10]. В его состав входят 8892 многоядерных процессора (общее число ядер – 35776). На момент запуска, Ломоносов занимал в мировом рейтинге суперкомпьютеров Top500 двенадцатое место.

Много шума наделал японский проект *пятого поколения* (1982-1992 годы). Было заявлено, что в основе компьютеров пятого поколения будут уже не вычисления, а логические заключения, т.е. произойдет переход от обработки данных к обработке знаний. Машину обещали научить воспринимать речевые команды человека, читать рукописный текст, анализировать графические изображения и делать многие другие нетривиальные для компьютера вещи. Планы проекта были грандиозны. Но, несмотря на щедрое финансирование и передовые позиции японских технологий, успехи оказались весьма скромными. На основе программного моделирования на компьютерах четвертого поколения удалось реализовать лишь отдельные детали проекта, причем реальная машина, работающая на базе логических выводов, так и не вышла за стены лабораторий.

Таким образом, создание принципиально новых компьютеров пятого поколения закончилось неудачей. Все компьютеры, используемые в настоящее время, по-прежнему построены на базе идей четвертого поколения. Классификация поколений «замерла» в ожидании новых революционных идей. Такие идеи особенно необходимы еще и потому, что электронная техника уже подошла к пределу быстродействия, который определяется законами физики: для увеличения скорости передачи данных требуется уменьшать размеры электронных деталей, но плотность упаковки транзисторов в полупроводниковом кристалле и так уже практически достигла максимально возможной. Поэтому идет поиск неэлектронных средств хранения и обработки данных.

В первую очередь, ученые попытались использовать в качестве носителя информации свет – так появились оптические процессоры[11]. В них можно применять *параллельную обработку данных,* например, одновременно выполнять какую-то операцию со всеми пик-

---

[6] Computerworld, 2009, N 39

[7] Крэш-тесты – это тесты, исследующие поведение машин при сильном ударе о бетонное препятствие.

[8] http://parallel.ru/vvv/lec1.html

[9] http://www.edu.ru/grants/?newsid=5248

[10] Флопс (FLOPS – floating point operations per second) – единица измерения количества операций с вещественными числами за 1 секунду; приставка «тера» добавляется по тем же правилам, что и при измерении информации; очевидно, что операции над вещественными числами гораздо сложнее и выполняются гораздо дольше, чем над целыми.

[11] http://ysa.ifmo.ru/data/publications/BOOK008/paper1-001.doc

селями изображения. В 2003 году был выпущен оптический процессор *Enlight256*[12], у которого оптическое ядро, а входные и выходные данные представлены в электронном виде. Быстродействие этого процессора – 8 триллионов операций в секунду. Он состоит из 256 лазеров, набора линз и фотоприемников. Оптические процессоры используются в военной технике и при обработке видеоданных в реальном времени.

Большие надежды связаны с разработкой квантовых компьютеров, в которых применяются идеи квантовой физики, описывающей законы микромира и поведение отдельных элементарных частиц. Данные для обработки в квантовом компьютере записываются в систему *кубитов* – квантовых битов. Затем с помощью специальных операций состояние этой системы изменяют по определенному алгоритму. Конечное состояние системы кубитов – это и есть ответ в задаче. Особые свойства кубитов[13] позволяют организовать параллельную обработку данных, так же, как и в многопроцессорных системах. Поэтому многие задачи, для решения которых сейчас не хватает вычислительных ресурсов (например, раскрытие шифров), будут достаточно быстро решены, как только квантовый компьютер будет построен.

В некоторых лабораториях ведется разработка биологических компьютеров (*биокомпьютеров*), которые работают как живой организм. Ячейки памяти биокомпьютеров – это молекулы сложных органических соединений, например, молекулы ДНК, в которых хранится наследственная информация. Сам процесс вычислений – это химическая реакция, результат – состав и строение получившей молекулы.

Проводятся также исследования и в области нанотехнологий, с помощью которых можно будет построить транзистор размером с молекулу.

### 5.1.3. Развитие возможностей от поколения к поколению

С каждым поколением вычислительных машин развиваются их *аппаратные возможности*. ЭВМ становятся более мощными и универсальными. Расширяется количество обрабатываемых **типов данных**:

*1 поколение* – только числовые данные;

*2 поколение* – добавляется простейшая обработка символов;

*3 поколение* – числа, текстовые и графические данные;

*4 поколение* – добавляются аудио- и видеоданные.

Появившись как устройство для облегчения вычислений, компьютер сейчас все более активно обрабатывает разнообразную нечисловую информацию. Чтобы подчеркнуть широкие возможности современных компьютеров, введён специальный термин – *мультимедиа*.

**Мультимедиа** (от латинских слов *multum* – много и *medium* – средства) – одновременное использование различных форм представления информации (графика, текст, видео, фотографии, анимация, звук и т.д.) и их объединение в одном объекте[14].

Как правило, при использовании технологий мультимедиа человек может влиять на показ материалов: перейти вперёд или вернуться назад, изменить настройки, выбрать один из предложенных вариантов и т.п. Подобное взаимодействия человека и компьютера называют *интерактивностью* (взаимной активностью). Пример мультимедиа-объекта – компьютерная презентация.

Другое направление в развитии аппаратной части – это увеличение разнообразия и одновременно рост сложности **внешних устройств**, присоединяемых к ЭВМ:

*1 поколение* – штекеры и переключатели, индикаторные лампочки, устройства ввода с перфокарт;

---

[12] http://dkws.narod.ru/linux/etc/optical/cpu.html

[13] В отличии от привычного нам бита, кубит устроен так, что способен вместить в себя гораздо больше информации.

[14] Термин «мультимедиа» имеет также несколько связанных с приведенным определением значений: мультимедиа-компьютер, мультимедиа-носитель, программные и аппаратные средства мультимедиа и др.

*2 поколение* – перфоленты, магнитные ленты и барабаны, печатающие устройства;
*3 поколение* – магнитные диски, текстовые и графические мониторы, графопостроители;
*4 поколение* – огромное разнообразие внешних устройств, в том числе

- накопители на лазерных дисках;
- мышь, джойстик, шлемы виртуальной реальности и др.;
- возможность подключения бытовой электроники (фотоаппаратов, музыкальных плееров, сотовых телефонов и др.) с помощью кабелей и беспроводных соединений.

Каждое новое поколение компьютеров расширяет возможности **программного обеспечения**. Использовать современный компьютер без ПО практически невозможно. Его стоимость нередко намного превышает стоимость аппаратной части (в первых поколениях было наоборот!).

*1 поколение*. Программы разрабатывали хорошо подготовленные специалисты на машинном языке, сама программа представляла собой последовательность чисел (машинных кодов). Стандартного программного обеспечения практически не было.

*2 поколение*. Появились первые языки программирования. Некоторые из них были разработаны для конкретных машин, но значительно удобнее оказались машинно-независимые языки, такие как *Фортран* (1957) и *Алгол* (1960). Написать программу на таком языке было значительно проще: с этим уже вполне мог справиться рядовой научный работник, причем не обязательно с математическим образованием. В конце второго поколения появились *мониторы* – специальные программы, управляющие последовательным прохождением заданий через ЭВМ в автоматическом режиме. Их дальнейшее развитие в следующем поколении привело к появлению операционных систем.

*3 поколение*. Созданы первые операционные системы (ОС), которые обеспечивали работу компьютеров в многопользовательском режиме и управляли большим количеством сложных внешних устройств (в первую очередь, магнитными дисками). Для «общения» с ОС разработаны специальные языки управления заданиями. Широкое распространение получили созданные ранее языки программирования, например, *Фортран* для математических вычислений и *Кобол* для экономических расчетов. Начали появляться пакеты прикладных программ для решения задач в конкретных областях.

*4 поколение*. Для управления компьютером пользователь теперь использует не язык программирования, а различные меню и кнопки. Например, необходимую команду можно выбрать из меню (перечня доступных в данной ситуации возможностей) на естественном языке. Стало реально освоить компьютер после очень короткой подготовки. Программное обеспечение для ПК становится необычайно разнообразным – написано столько программ, что их трудно даже просто систематизировать. Казалось бы, такое разнообразие ПО должно сделать программирование ненужным, но нередко проще написать собственную небольшую программу решения конкретной задачи, чем тратить время на поиск и освоение возможностей готовых универсальных пакетов.

Таким образом, при переходе от поколения к поколению возрастает вычислительная мощность компьютеров. Значительная часть новых возможностей направляется на повышение удобства работы пользователя. В человеко-машинном общении отчетливо прослеживается движение от машинного языка к языкам, естественным для человека. В результате расширяется область применения и круг пользователей компьютерной техники.

## ? Контрольные вопросы

1. Что такое компьютер?
2. Охарактеризуйте программную и аппаратную части компьютера.
3. Почему универсальный компьютер с изменяемой программой удобнее, чем специализированная техника? Ответ обоснуйте.
4. Что такое цифровая и аналоговая техника?
5. Почему цифровая техника вытеснила аналоговую?
6. Перечислите основные вехи в истории развития вычислительной техники.

7. Какова заслуга Чарльза Бэббиджа?
8. В честь кого названы языки программирования Ада и Паскаль? Какое отношение они имеют к вычислительной технике?
9. *Подберите дополнительный материал по вопросу о приоритете создания первой ЭВМ.
10. Что такое транзистор и микросхема? Из чего они изготавливаются?
11. С какой целью разрабатывались первые микропроцессоры?
12. *Почему микропроцессор *Intel 4004* был специально спроектирован для работы только с четырехбитными данными? *Указание*: вспомните, как можно хранить отдельные десятичные цифры числа. (Ответ: для вычислений в калькуляторах удобен 4-разрядный двоично-десятичный код).
13. По какому принципу ЭВМ делятся на поколения?
14. Почему время существования того или иного поколения всегда указывается приблизительно?
15. Перечислите все поколения ЭВМ и назовите элементную базу каждого из них.
16. Что дает уменьшение базовых элементов вычислительной техники?
17. *Почему электронные схемы требуют охлаждения? Все ли элементы нуждаются в дополнительном охлаждении?
18. Какие поколения вычислительных машин построены на базе полупроводниковых технологий? Чем отличается друг от друга их элементная база?
19. Объясните, почему большинство ЭВМ третьего поколения имели крупные габариты, несмотря на очередное уменьшение размеров элементной базы.
20. Когда появились первые семейства ЭВМ? Какая фирма предложила идею? В чем преимущества выпуска совместимых моделей?
21. Компьютеры какого поколения сейчас стоят на полках магазинов?
22. Какие разновидности компьютеров входят в четвертое поколение?
23. Как вы понимаете термин «персональный компьютер»?
24. Какие семейства персональных компьютеров вы знаете? Какое из них появилось раньше?
25. Перечислите бытовые приборы, в которых применяются микропроцессоры.
26. Что такое суперкомпьютеры? Зачем они используются?
27. Найдите в Интернете рейтинг суперкомпьютеров *Top500*. Какие страны занимают в нем лидирующее положение? Есть ли там российские компьютеры?
28. *Зачем в суперкомпьютерах так много процессоров? Подумайте, любая ли задача может быть решена быстрее, если ее считать параллельно на множестве процессоров? (В качестве помощи можно воспользоваться аналогией с распределением частей одного большого задания между учениками класса.)
29. Назовите примеры вычислительных машин каждого из четырех поколений. Найдите дополнительный материал об этих машинах.
30. Что вы можете сказать о судьбе пятого поколения?
31. *Почему, по-вашему, уже довольно давно не происходило смены поколений?
32. Найдите дополнительный материал о разрабатываемых в лабораториях принципиально новых компьютерах.
33. Какие типы данных обрабатывались на ЭВМ каждого из поколений?
34. Как изменялся набор внешних устройств при переходе от одного поколения к другому?
35. Опишите, как происходило развитие программного обеспечения.
36. Что вы можете сказать по поводу роли программного обеспечения: уменьшается она или увеличивается по сравнению с предыдущими поколениями?
37. Предположим, что появился процессор с каким-то принципиально новым свойством. Как быстро этим свойством смогут воспользоваться потребители? Какова роль программного обеспечения в этом?

38. Быстродействие вычислительной техники постоянно растет. Как же тогда объяснить, что пользователи жалуются на «медлительные» компьютеры и все время стараются купить новые, еще более производительные?
39. *Влияет ли развитие программных средств на развитие аппаратной части? (Ответ: ПО все усложняется, что требует все более мощную аппаратуру. Кроме того, во многих случаях функции оборудования перекладываются на программы, что позволяет упростить аппаратную часть компьютера.)
40. Что представляли собой программы для первых машин? Почему для их записи было удобно использовать не двоичную систему счисления, а восьмеричную или шестнадцатеричную?
41. Зачем были созданы языки программирования? Когда они появились?
42. Когда появились операционные системы и с чем это связано?
43. Попробуйте назвать положительные и отрицательные последствия огромного разнообразия существующих программ.
44. Почему развитие ПО расширяет количество пользователей компьютера?
45. *Насколько сейчас, по-вашему, актуально умение программировать? Попробуйте найти аргументы «за» и «против» (учитывайте разные цели работы на компьютере у людей).

## 5.2. Фундаментальные принципы устройства компьютеров

В предыдущем разделе вы увидели, что вычислительная техника в своем развитии прошла целый ряд характерных этапов. Несмотря на это, некоторые фундаментальные (базовые, основные) принципы устройства ЭВМ почти не изменились. Поэтому логично начать знакомство с устройством компьютера именно с них.

Классические принципы построения ЭВМ были предложены в работе *А. Беркса, Г. Голдстайна* и *Д. фон Неймана* «*Предварительное рассмотрение логической конструкции электронного вычислительного устройства*». Обычно выделяют[15] следующие наиболее важные идеи этой работы:

- состав основных компонентов вычислительной машины;
- принцип двоичного кодирования;
- принцип адресности памяти;
- принцип иерархической организации памяти;
- принцип хранимой программы;
- принцип программного управления.

Рассмотрим их подробнее.

### 5.2.1. Общие принципы

**Основные компоненты машины**. В самом первом разделе с таким названием фон Нейман с соавторами определили и обосновали состав ЭВМ:

> *«Так как законченное устройство будет универсальной вычислительной машиной, оно должно содержать несколько основных органов, таких как орган арифметики, памяти, управления и связи с оператором. Мы хотим, чтобы машина была полностью автоматической, т.е. после начала вычислений работа машины не зависела от оператора».*

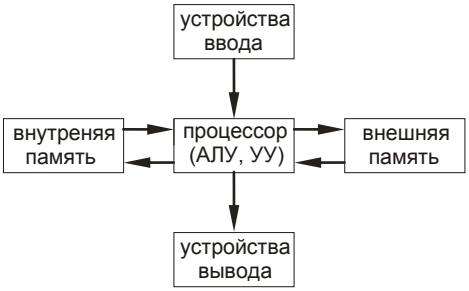
Таким образом, ЭВМ должна состоять из нескольких блоков, каждый из которых выполняет вполне определённую функцию. Эти блоки есть и в сегодняшних компьютерах:

- *арифметико-логическое устройство* (АЛУ), в котором выполняется обработка данных;

[15] Эта техническая статья не содержит отдельного пронумерованного перечня принципов, поэтому в учебной литературе встречаются непринципиальные отличия в их формулировке и описании.

- *устройство управления* (УУ), обеспечивающее выполнение программы и организующее согласованное взаимодействие всех узлов машины; сейчас АЛУ и УУ изготавливают в виде единой интегральной схемы – *микропроцессора*;
- *память* – устройство для хранения программ и данных; память обычно делится на *внутреннюю* (для временного хранения данных во время обработки) и *внешнюю* (для длительного хранения между сеансами обработки);
- *устройства ввода*, преобразующие входные данные в форму, доступную компьютеру;
- *устройства вывода*, преобразующие результаты работы ЭВМ в форму, удобную для восприятия человеком.

В классическом варианте все эти устройства взаимодействовали через процессор:




**Принцип двоичного кодирования**. Устройства для хранения двоичной информации и методы ее обработки наиболее просты и дешевы. Поскольку в ЭВМ используется двоичная система, необходимо переводить данные из десятичной формы в двоичную (при вводе) и наоборот (при выводе результатов). Однако такой перевод легко автоматизируется, и многие пользователи даже не знают об этих внутренних преобразованиях.

В первых машинах использовались только числовые данные. В дальнейшем ЭВМ стали обрабатывать и другие виды информации (текст, графика, звук, видео), но это не привело к отмене принципа двоичного кодирования. Даже цифровые сигнальные процессоры[16], предназначенные для обработки цифровых сигналов в реальном времени, используют двоичное представление данных.

В истории известен пример успешной реализации троичной ЭВМ «*Сетунь*» (1959 год, руководитель проекта *Н. П. Брусенцов*), но он так и остался оригинальным эпизодом и не оказал влияния на эволюцию вычислительной техники. В первую очередь, это связано с серьёзными проблемами, которые возникают при изготовлении элементов троичного компьютера (триггеров – ячеек с тремя устойчивыми состояниями, сумматоров и т.д.) на основе полупроводниковых технологий. Эти проблемы так и не были решены, тогда как наладить массовое производство аналогичных устройств для двоичных компьютеров оказалось значительно проще.

## 5.2.2. Принципы организации памяти

**Принцип адресности памяти**. Оперативная память машины состоит из отдельных битов. Для записи или считывания группы соседних битов объединяется в *ячейки памяти*, каждая из которых имеет свой *адрес* (номер). Нумерацию ячеек принято начинать с нуля.

**Адрес ячейки памяти** – это её номер.

Ячейка – это минимально возможный считываемый из памяти объем данных: невозможно прочитать меньшее количество бит, а тем более отдельный бит.

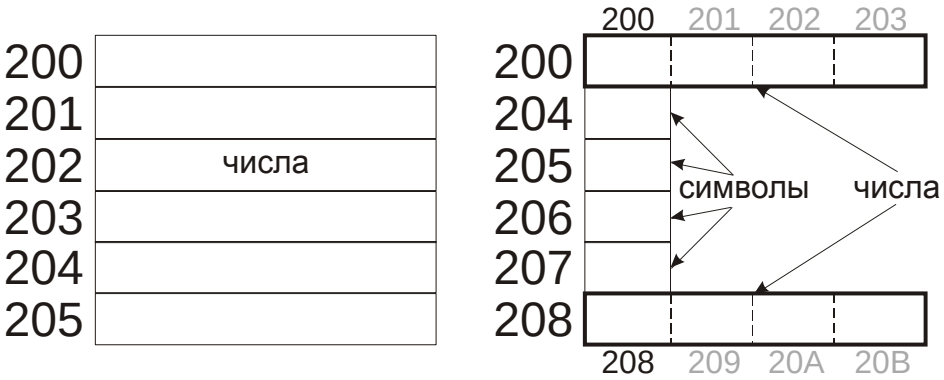
Использование чисел для нахождения в памяти требуемых ячеек выглядит абсолютно естественно: в компьютерах любая информация кодируется числами, так что адреса ячеек не исключение из этого фундаментального правила. Если номера соседних ячеек отличаются на единицу, удобно организовывать их последовательную обработку.

[16] В англоязычной литературе их называют *DSP = Digital Signal Processor*.

Разрядность ячеек памяти (количество бит в ячейке) в разных поколениях была различной. Первоначально ЭВМ были построены исключительно для математических расчетов. Числа желательно было представлять как можно точнее, поэтому ячейки ОЗУ в первых машинах были длинными. Кроме чисел, машина должна была хранить в памяти еще и команды программы; как правило, в то время размер числовой ячейки совпадал с размером команды, что существенно упрощало устройство памяти.

Примерно на стыке второго и третьего поколений ЭВМ стали использовать для обработки символьной информации, что привело к серьезному неудобству: в существующую числовую ячейку памяти помещалось 4-5 символов. Инженеры выбрали наиболее простое решение проблемы – уменьшить размер ячейки так, чтобы можно было обращаться к каждому символу отдельно. *Байтовая* память, основой которой стала восьмибитная ячейка, прекрасно зарекомендовала себя и используется в компьютерной технике до настоящего времени.

В результате перехода к «коротким» ячейкам памяти числа стали занимать несколько ячеек (байт), каждая из которых имеет собственный адрес. На рисунке слева показана организация ячеек памяти первых ЭВМ, а на рисунке справа – современная (байтовая) структура памяти.




На левом рисунке числа занимают по одной ячейке, причем номера этих ячеек отличаются на единицу. Справа показано два 32-битных числа, которые хранятся в байтах 200-203 и 208-20B (адреса указаны в шестнадцатеричной системе). По принятому правилу за адрес числа принимается наименьший из адресов, так что в данном случае адреса чисел – 200 и 208. Кроме того, на правом рисунке между числами (в байтах с 204 по 207) размещены четыре символа. Заметим, что современные компьютеры могут извлекать из памяти до восьми соседних байтовых ячеек за одно обращение к памяти.

Очень важно, что информация может считываться из ячеек и записываться в них в произвольном порядке, поэтому организованную таким образом память принято называть *памятью с произвольным доступом* (англ. *RAM = random access memory*). Чтобы лучше понять смысл этого термина, сравните такую память с магнитной лентой, данные с которой можно получить только путем последовательного чтения.

Часто термин RAM отождествляют с русским термином ОЗУ – *оперативное запоминающее устройство*. Это не совсем точно. Дело в том, что кроме ОЗУ существует еще одна разновидность памяти с произвольным доступом – *постоянное запоминающее устройство* (ПЗУ, англ. *ROM = Read Only Memory* – память только для чтения). Главное отличие ПЗУ от ОЗУ заключается в том, что при решении задач пользователя содержимое ПЗУ не может быть изменено. ПЗУ гораздо меньше ОЗУ по объему, но это очень важная часть компьютера, поскольку в нем хранится доступное в любой момент программное обеспечение. Благодаря этому ПО компьютер сохраняет работоспособность даже тогда, когда в ОЗУ нет никакой программы.

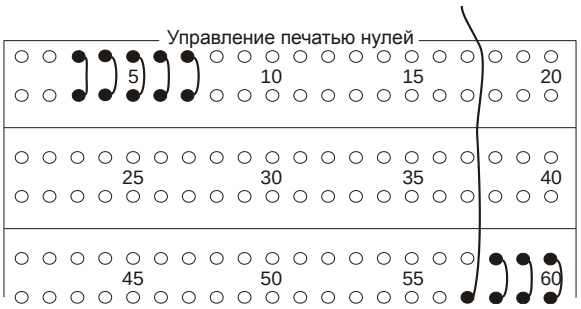
Таким образом, ОЗУ и ПЗУ – это два вида памяти с произвольным доступом, обращение к данным в которых построено на основе принципа адресности.

**Принцип иерархической организации памяти.** К памяти компьютера предъявляется два противоречивых требования: ее объем должен быть как можно больше, а ско-

рость работы – как можно выше. Ни одно реальное устройство не может удовлетворить им одновременно. Любое существенное увеличение объема памяти неизбежно приводит к уменьшению скорости ее работы. Действительно, если память большая, то обязательно усложняется механизм нахождения в ней требуемых данных[17], а это сразу замедляет чтение из памяти. Кроме того, чем быстрее работает память, тем она дороже, и, следовательно, меньше памяти можно установить за приемлемую для потребителей стоимость.

Чтобы преодолеть противоречие между объемом памяти и ее быстродействием, используют несколько различных видов памяти, связанных друг с другом. Когда в 1946 году впервые формулировался этот принцип, в состав ЭВМ предполагалось включить всего два вида памяти: оперативную память и память на магнитной проволоке (предшественник устройств хранения данных на магнитной ленте). Дальнейшее развитие вычислительной техники подтвердило необходимость построения иерархической памяти: в современном компьютере уровней иерархии гораздо больше.

## 5.2.3. Выполнение программы




Фрагмент коммутационной панели устройства IBM-557; требуемая операция получается соединением (коммутацией) соответствующих отверстий

**Принцип хранимой программы.** Первые ЭВМ программировались путем установки перемычек на специальных панелях, так что процесс подготовки к решению задачи мог растянуться на несколько дней. Такое положение дел никого не устраивало, и в фон-неймановской архитектуре было предложено представлять команды в виде двоичного кода. Код программы, записанный заранее[18] на перфокарты или магнитную ленту, можно было ввести в машину достаточно быстро.

Поскольку команды программы и данные по форме представления стали одинаковыми, их можно хранить в *единой памяти*[19] вместе с данными. Не существует принципиальной разницы между двоичными кодами машинной команды, числа, символа и т.д. Это утверждение иногда называют **принципом однородности памяти**. Из него следует, что команды одной программы могут быть получены как результат работы другой. Именно так текст программы на языке высокого уровня переводится (транслируется) в машинные коды конкретной машины.

Код программы может сохраняться во внешней памяти (например, на дисках) и затем загружаться в оперативную память для повторных вычислений. Благодаря простоте замены программ, ЭВМ стали универсальными устройствами, способными решать самые разнообразные задачи в произвольном порядке и даже одновременно.

**Принцип программного управления.** Любая обработка данных в вычислительной машине происходит по программе. Принцип программного управления определяет наиболее общий механизм автоматического выполнения программы.

---

[17] Например, память большого объема требует многоразрядного адреса, что, в свою очередь, приводит к очень большому количеству линий связи. В итоге приходится как-то изменять способ адресации, например, передавать адрес по частям.

[18] До появления персональных компьютеров для этого использовались специальные *устройствах подготовки данных*. Такая схема ускоряла процесс ввода и исключала простои ЭВМ, связанные с длительным набором программ.

[19] Известна также так называемая *гарвардская архитектура*, в которой программы и данные хранятся в разных областях памяти. Несмотря на повышение надежности, она не получила широкого распространения.

Важным элементом устройства управления в машине фон-неймановской архитектуры является специальный регистр – *счетчик адреса команд*[20]. В нем в любой момент хранится адрес команды программы, которая будет выполнена следующей.

Используя значение из счетчика, процессор считывает из памяти очередную команду программы, расшифровывает ее и выполняет. Затем процесс повторяется для следующей команды и т.д. Процессор выполняет команды по следующему алгоритму (его часто называют *основным алгоритмом работы процессора*):

1) из ячейки памяти, адрес которой записан в счетчике адреса команд, выбирается очередная команда программы; на время выполнения она сохраняется в специальном *регистре команд*;
2) значение счетчика адреса команд увеличивается так, чтобы он указывал на следующую команду;
3) выбранная команда выполняется (например, при сложения двух чисел оба слагаемых считываются в АЛУ, складываются и результат операции сохраняется в регистре или ячейке памяти);
4) далее весь цикл повторяется сначала.

Таким образом, автоматически выполняя одну команду программы за другой, компьютер может исполнить любой линейный алгоритм. Для того, чтобы в программе можно было использовать ветвления и циклы, необходимо нарушить естественную последовательность выполнения команд. Для этого существуют специальные *команды перехода,* которые на этапе 3 заносят в счетчик адреса новое значение – адрес перехода. Чаще всего в программах используется *условный* переход, то есть переход происходит только при выполнении определенного условия.

Легко понять, что запуска основного алгоритма работы процессора в счетчик адреса команд должно быть предварительно занесено начальное значение – адрес первой выполняемой команды. В первых ЭВМ оператор вводил этот адрес вручную. В современных компьютерах при включении питания в счетчик аппаратно заносится некоторое значение, которое указывает на начало программы, хранящейся в ПЗУ. Эта программа тестирует устройства компьютера и приводит их в рабочее состояние, а затем загружает в ОЗУ *начальный загрузчик* операционной системы (как правило, с диска). Ему и передается дальнейшее управление, а стартовая программа из ПЗУ завершает свою работу. Начиная с этого момента, поведение компьютера уже определяется установленным на нем программным обеспечением (см. гл. 6).

Чтобы ускорить выполнение программы, основной алгоритм работы процессора был значительно усовершенствован. Идея была заимствована из конвейерного производства, где несколько рабочих одновременно выполняют различные операции (каждый над своим экземпляром изделия). Аналогично в современных микропроцессорах для каждого этапа выполнения команды создан отдельный аппаратный блок. Выполнив свою операцию, он передает результаты следующему блоку, а сам начинает выполнять очередную команду.

Проще всего понять этот механизм на примере первого этапа – выборки команды из ОЗУ. Специализированный блок выборки извлекает из памяти последовательно расположенные команды, не дожидаясь окончания их обработки. Прочитанные команды размещаются в специальной рабочей памяти внутри микропроцессора. В итоге, когда первая из выбранных команд будет завершена, за следующей не придется обращаться к ОЗУ, т.к. она уже находится во внутренней памяти микропроцессора. Учитывая, что обращение к ОЗУ занимает значительно большее времени, чем пересылка данных внутри процессора, такая *опережающая выборка* значительно ускоряет выполнение программы.

На практике применение конвейерного метода не так просто. Например, следующую команду часто не удается выполнить, поскольку она использует результат предыдущей, или сразу нескольким командам потребуется одновременно обратиться к ОЗУ. Тем

[20] В различных процессорах этот регистр может называться по-разному: например, в семействе *Intel* он обозначается *IP = Instruction Pointer* (указатель на инструкцию).

не менее, этот метод широко применяется в микропроцессорах. В некоторых моделях используются параллельные конвейеры, так что в некоторых случаях к моменту завершения выполнения одной команды уже готов результат следующей.

### 5.2.4. Что называют архитектурой

Описанные фон Нейманом и его соавторами классические принципы построения вычислительных устройств применялись во всех поколениях ЭВМ. В дополнение к ним в каждом конкретном семействе (*PDP, ЕС ЭВМ, Apple, IBM PC* и др.) формулируются свои собственные принципы устройства, благодаря которым обеспечивается аппаратная и программная совместимость моделей. Для пользователей это означает, что все существующие программы будут работать и на новых моделях того же семейства компьютеров. В литературе общие принципы построения конкретного семейства компьютеров называют *архитектурой*. К архитектуре обычно относят:

- принципы построения системы команд и их кодирование;
- форматы данных и особенности их машинного представления;
- алгоритм выполнения команд программы;
- способы доступа к памяти и внешним устройствам;
- возможности изменения конфигурации оборудования.

Стоит обратить внимание на то, что архитектура описывает именно общее устройство вычислительной машины, а не особенности изготовления конкретного компьютера (набор микросхем, тип жесткого диска, емкость памяти, тактовая частота). Например, наличие видеокарты как устройства для организации вывода информации на дисплей входит в круг вопросов архитектуры. А вот является ли видеокарта частью основной платы компьютера или устанавливается на нее в виде отдельной платы, с точки зрения архитектуры значения не имеет. Иначе могло бы получиться, что для интегрированной в плату видеокарты потребовалась бы отдельная версия графического редактора!

### ? Контрольные вопросы

1. Найдите материалы, подтверждающие, что Джон фон Нейман не был единоличным автором «фон-неймановской» архитектуры ЭВМ.
2. *Чем еще известен Джон фон Нейман?
3. Перечислите принципы фон-неймановской архитектуры и кратко объясните каждый из них.
4. Назовите основные компоненты вычислительного устройства. Каково их назначение? Согласны ли вы с тем, что полученный набор узлов логичен и обоснован?
5. В чем состоит принцип двоичного кодирования?
6. *Найдите материалы о троичной ЭВМ «Сетунь». Сравните двоичные и троичные ЭВМ.
7. Вспомните, как кодируются в компьютере числа, тексты, графика. Соблюдается ли принцип двоичного кодирования?
8. По какому алгоритму вводимые в компьютер десятичные числа можно перевести во внутреннее двоичное представление? Как перевести обратно результаты расчёта?
9. Что такое ячейка памяти? Что такое адрес ячейки?
10. Что вы знаете о разрядности ячеек ОЗУ разных поколений?
11. Почему появилась байтовая память?
12. Можно ли заменить в ячейке памяти содержимое одного бита, не затрагивая значений соседних? Почему? (Ответ: нет, только путем считывания, обработки в процессоре и обратной записи)
13. Приведите примеры различных типов данных и назовите их разрядность. Сколько байт памяти потребуется для хранения каждого из этих типов данных?
14. Что такое иерархическая организация памяти?

15. Почему большая по объему память обычно работает медленнее, чем маленькая?
16. В чем состоит принцип хранимой программы?
17. Где может храниться программа?
18. Можно ли к нечисловым данным (символам, графическим и звуковым данным) применять арифметические операции? (Ответ: При определенных условиях можно, потому что все виды информации закодированы в виде двоичных чисел. Очень важно делать это осмыслено, учитывая закономерности кодирования данных. Вот несколько разумных примеров. Путем формального прибавления к коду символа определенной константы можно переходить от заглавных букв к строчным. Вычисляя средние значения интенсивности цветов соседних точек можно получить примерный цвет «промежуточной» точки. Умножая все значения интенсивности звукового сигнала на некоторый множитель, можно сделать общее звучание громче.)
19. Как вы понимаете фразу «любая обработка данных в вычислительной машине происходит по программе»? Чем компьютер в этом отношении отличается от простого калькулятора?
20. Сформулируйте основной алгоритм выполнения команды в компьютере.
21. Что такое счетчик адреса команд и какова его роль в основном алгоритме?
22. Опишите, что происходит в момент включения компьютера с точки зрения принципа программного управления.
23. Можно ли нарушить последовательность выполнения команд в программе? Для чего это может потребоваться?
24. Всегда ли в новом компьютере есть какая-либо программа? (Ответ: Минимальный набор программ, включая программу начальной загрузки, хранится в ПЗУ.)
25. Что такое конвейер и как он работает при выполнении программы?
26. *Почему команды перехода нарушают работу конвейера?
27. Какие из предложенных в «Предварительном рассмотрении…» принципов продолжают применяться в современных компьютерах без всяких изменений, а какие сохранились, но в несколько измененном виде? Объясните, почему потребовались эти изменения.
28. Что такое архитектура? Какие детали устройства компьютера к ней не относятся?
29. В чем преимущества единой архитектуры семейств ЭВМ для пользователей и для производителей?
30. Какие семейства вычислительных машин вы знаете?

## Задачи

1. * Используя условные команды «считать байт из памяти», «записать байт в память», а также стандартные логические операции «И», «ИЛИ» и «НЕ», составьте алгоритм который в байте, хранящемся в памяти, а) установит младший бит данных в единицу, не затрагивая содержимого остальных двоичных разрядов; б) сбросит его в ноль.
2. Описанная в «Предварительном рассмотрении…» конструкция ЭВМ имела ячейки памяти, состоящие из 40 двоичных разрядов. Сколько современных байтовых ячеек потребуется для хранения числа такой же разрядности?
3. В языке Паскаль есть тип чисел **`word`**, значением которого являются целые положительные 16-разрядные числа. Сколько байт занимает такое число в памяти? Какое максимальное значение может иметь этот тип чисел? (65535)
4. В микропроцессорах семейства *Intel* для увеличения на единицу одного из регистров процессора используется команда, имеющая код $41_{16}$. Пользуясь таблицей символов, определите, какая буква соответствует этому же самому коду. Если рассматривать этот код как целое число, чему оно будет равно в десятичной системе счисления?
5. *Изучите содержимое текстового файла, используя программу, которая способна отображать данные в виде шестнадцатеричных кодов. Попробуйте найти коды из-

вестных вам символов. Сделайте то же самое с графическим файлом: удастся ли вам найти там те же самые коды («символы»)?

6. *Найдите таблицу кодов команд процессора *Intel*. Сравните эти коды с кодами, которые вы нашли в предыдущем задании. Удалось ли найти совпадения?

# 5.3. Магистрально-модульная организация компьютера

## 5.3.1. Что значит «устройство компьютера»?

Компьютер – это пример очень сложной техники. При изучении таких систем возможно несколько разных подходов. Например, можно изучать:

- устройство конкретного экземпляра компьютера: набор микросхем, тип основной платы, конструкцию и разновидности модулей памяти и т.п.;
- семейство компьютеров, например, IBM-совместимые персональные компьютеры;
- различные конструкции компьютеров (настольные компьютеры, портативные компьютеры, карманные компьютеры);
- *функциональное устройство* компьютера, т.е. его основные узлы и способы взаимодействия между ними.

Каждый из этих подходов полезен при решении определенных задач. Так для настройки конкретного компьютера необходимо точно знать марки и параметры его устройств. Определить эти данные можно с помощью специального программного обеспечения. К сожалению, любые знания в этой области очень быстро устаревают, поскольку аппаратура постоянно меняется.

Если изучать особенности одного *семейства компьютеров*, мы получим «однобокое» представление об устройстве компьютерной техники, так как каждое семейство имеет свои особенности.

Современные компьютеры очень разнообразны и поэтому имеют самую различную *конструкцию* и внешний вид. Настольный ПК состоит из системного блока и подключенных к нему внешних устройств. Такая конструкция удобна для пользователя, поскольку все устройства можно разместить на столе так, как ему хочется.

В переносных компьютерах весь минимально необходимый набор устройств собран в одном корпусе. Сейчас такие компьютеры называют *ноутбуками* (англ. *notebook* – тетрадь, блокнот). По своим вычислительным возможностям они практически не уступают настольным ПК.

Растет популярность так называемых *нетбуков* (от слов «Интернет» и «ноутбук») – так называют очень маленькие и легкие переносные компьютеры. Кроме меньшего размера и веса, нетбуки отличаются от ноутбуков бóльшим временем автономной работы и меньшей стоимостью. Нетбуки предназначены для пользователей, которые применяют компьютер главным образом для работы в Интернете и подготовки простых документов. Их используют люди, совершающие большое число поездок.

*Карманные персональные компьютеры* (КПК)[21] умещаются на ладони. У большинства из них даже нет клавиатуры, а для ввода информации нажимают пластиковой палочкой (она называется *стилус*) на сенсорный (реагирующий на прикосновение) экран.

С другой стороны, мощные серверы и суперкомпьютеры по-прежнему собираются в виде крупных «шкафов», напоминающих ЭВМ предыдущих поколений. Наконец, нельзя не упомянуть и о бытовой электронике, которая все больше и больше приближается к традиционным компьютерам.

Разнообразие типов современных компьютеров говорит о том, что конструкция – это не самое главное. В то же время, как показано в п. 5.2, их функциональное устройство

[21] Их называют также наладонниками (англ. *palmtop*) и PDA (англ. *Personal Digital Assistant* – персональный цифровой помощник).

практически не изменяется. Поэтому далее мы подробно рассмотрим основные узлы компьютера (процессор, память и устройства ввода и вывода) и взаимодействие между ними.

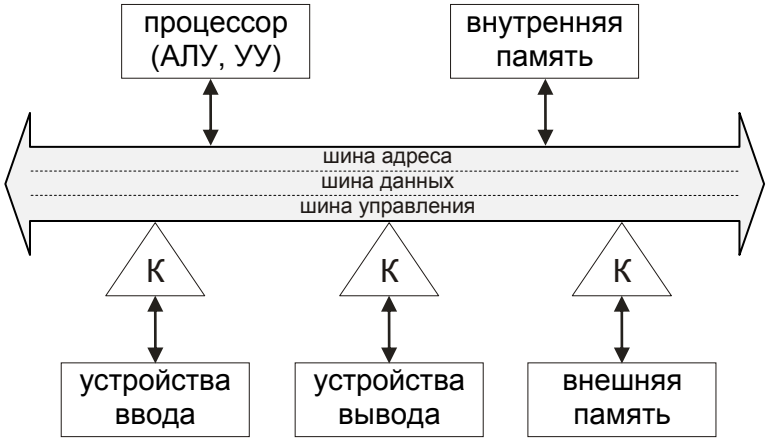
### 5.3.2. Взаимодействие устройств

Процессор должен обмениваться данными с внутренней памятью и устройствами ввода и вывода. Выделить отдельные каналы для связи процессора с каждым из многочисленных устройств нереально. Вместо этого сделана общая линия связи, доступ к которой имеют все устройства, использующие ее по очереди. Такой информационный канал называется *шиной*.

**Шина** (или *магистраль*) – это группа линий связи для обмена данными между несколькими устройствами компьютера.

Традиционно шина делится на три части:

- шина данных, по которой передаются данные;
- шина адреса, определяющая, куда именно передается информация;
- шина управления, которая организует процесс обмена (несет сигналы чтение/запись, обращение к внутренней/внешней памяти, данные готовы/не готовы и т.п.).




Рассмотрим процесс записи данных из процессора в память. Процессор выставляет на шину данных информацию для записи, на шину адреса – нужный адрес памяти, а на шину управления – сигналы для записи информации в память. Далее он вынужден ожидать, пока данные будут «взяты» с шины. В это время все остальные устройства постоянно «слушают» шину (проверяют ее состояние). В нашем примере по сигналам на шине память обнаруживает, что для нее имеются данные. Она сохраняет их по заданному адресу и должна по шине управления сообщить процессору, что операция завершена. На практике, учитывая высокую надежность работы памяти, сигнал подтверждения часто не используется: процессор просто выжидает определенное время и продолжает выполнение программы. Из этого примера понятно, что для успешного обмена данными по шине должны быть введены четкие правила (их принято называть *протоколом шины*), которые должны соблюдать все устройства.

По сравнению с первыми ЭВМ, взаимодействие процессора с внешними устройствами организуется теперь по-другому. В классической архитектуре процессор контролировал все процессы ввода-вывода. Получалось так, что быстродействующий процессор тратил много времени на ожидание при работе с значительно более медленными внешними устройствами. Поэтому появились специальные электронные схемы, которые руководят обменом информацией между процессором и внешними устройствами. В третьем по-

колении такие устройства назывались *каналами ввода-вывода*, а в четвертом – *контроллерами*[22] (на схеме они обозначены буквой К).

**Контроллер** – это электронная схема для управления внешним устройством и для простейшей предварительной обработки данных.

Современный контроллер – это микропроцессор, предназначенный специально для обслуживания одного (или даже нескольких однотипных) устройств ввода-вывода или внешней памяти. Нагрузка на центральный процессор при этом существенно снижается, и это увеличивает эффективность работы всей системы в целом. Контроллер, собранный в виде отдельной микросхемы называют *микроконтроллером*.

В качестве примера рассмотрим контроллер современного жесткого диска. Его основная задача – по принятым от процессора координатам найти на диске требуемые данные, прочитать их и передать в ОЗУ. Но контроллер способен выполнять и другие, порой весьма нетривиальные функции. Так он сохраняет в служебной области диска информацию обо всех имеющихся на магнитной поверхности некачественно изготовленных секторах (а их при современной высокой плотности записи избежать не удается!) и способен «на ходу» подменять их резервными, что создает видимость диска, который полностью свободен от дефектов[23].

Как видно из приведенной выше схемы, теперь данные могут передаваться между внешними устройствами и ОЗУ напрямую, минуя процессор. Кроме того, наличие шины существенно упрощает подсоединение к ней новых устройств. Архитектуру, которую можно легко расширять за счет подключения к шине новых устройств, часто называют *магистрально-модульной*.

Если спецификация на шину (детальное описание всех ее логических и физических параметров) является открытой (опубликована), то производители могут разрабатывать к такой шине любые дополнительные устройства. Такой подход называют *принципом открытой архитектуры*. При этом в компьютере предусмотрены стандартные разъемы для подключения новых устройств, удовлетворяющих стандарту. Поэтому пользователь может собрать такой компьютер, который ему нужен. Необходимо только помнить, что при подключении любого нового устройства нужно установить специальную программу – *драйвер*, которая обеспечивает обмен данными между этим устройством и процессором.

В современных компьютерах для повышения эффективности работы используется несколько шин, например, одна – между процессором и памятью, другая – от процессора к видеосистеме и т.д.

### 5.3.3. Обмен данными с внешними устройствами

Существуют три режима обмена данными между центральным процессором (ЦП) и внешними устройствами:

- программно-управляемый ввод/вывод;
- обмен с устройствами по прерываниям;
- прямой доступ к памяти (ПДП).

При **программно-управляемом** обмене все действия по вводу или выводу предусмотрены в теле программы. Процессор полностью руководит ходом обмена, включая ожидание готовности периферийного устройства и прочие временны**е** задержки, связанные с процессами ввода/вывода. Достоинства этого метода – простота и отсутствие дополнительного оборудования, недостаток – большие потери времени из-за ожидания быстро работающим процессором более медленных устройств ввода/вывода.

При обмене **по прерываниям** устройства ввода-вывода в случае необходимости сами требуют внимания процессора. Например, клавиатура оповещает процессор каждый

[22] Это название происходит от английского слова *control* – управление; не следует путать с русским словом «контролёр».

[23] http://spider.nrcde.ru/music/articles/hardware/hdd_outsins.html

раз, когда была нажата или отпущена клавиша; все остальное время процессор выполняет программу, вообще «не отвлекаясь» на клавиатуру. Когда прерывание произошло, ЦП «откладывает» на некоторое время выполнение основной программы и переходит на служебную программу обработки прерывания. Завершив его обработку, ЦП снова возвращается к тому месту программы, где она оказалась прервана. При этом основная программа даже «не заметит» возникшей задержки. Этот режим обмена более сложен, но зато значительно эффективнее – процессор не тратит время на ожидание.

Представим себе, что в кабинете начальника идет совещание, и в этот момент по телефону поступает важная информация, требующая немедленного принятия решения. Секретарша, не дожидаясь конца совещания, сообщает начальнику о звонке. Тот, прервав свое выступление, снимает трубку, выясняет суть дела и сообщает свое решение. Затем он продолжает совещание, как ни в чем не бывало. Здесь роль ЦП играет начальник, а телефонный звонок – это запрос (требование) на прерывание. «Секретарша» в компьютере тоже предусмотрена – это контроллер прерываний, анализирующий и сортирующий все поступающие прерывания с учетом их важности (*приоритета*).

Механизм прерываний используется не только в аппаратной части, но и в программах, которые основаны на обработке *событий* (нажатий на клавиши, команд управления от мыши и т.п.). Такая технология лежит в основе современных операционных систем и применяется в системах разработки программ *MS Visual Studio, Visual Basic, Delphi, Lazarus* и им подобных.

В обоих описанных выше вариантах управление обменом выполнял центральный процессор. Именно он извлекал из памяти выводимые данные (или записывал туда вводимые), подсчитывал их количество и полностью контролировал работу шины. Если передаваемые данные не требуют сложной обработки, ЦП напрасно расходует время на проведение обмена. Чтобы освободить процессор от этой работы и увеличить скорость передачи *крупных блоков* данных от устройства ввода в память и обратно, применяется **прямой доступ к памяти** (ПДП, англ. *DMA = **D**irect **M**emory **A**ccess*).

Принципиальное отличие ПДП состоит в том, что в этом режиме процессор не *производит* обмен, а только *подготавливает* его, программируя контроллер ПДП: устанавливает режим обмена, а также передает начальный адрес ОЗУ и количество циклов обмена. Далее контроллер в ходе ПДП самостоятельно наращивает первое значение и уменьшает второе, что позволяет освободить центральный процессор.

Изложенный материал о режимах ввода/вывода может быть сведен в таблицу (здесь УВВ обозначает устройство ввода-вывода):

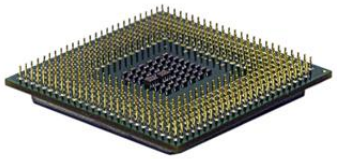
| вид обмена | начинает обмен | руководит обменом | текущая программа | программа обмена |
|---|---|---|---|---|
| программный | ЦП | ЦП | программа обмена – *часть* текущей программы | |
| прерывания | УВВ | ЦП | прерывается | специальная подпрограмма |
| ПДП | УВВ, ЦП | контроллер ПДП | выполняется параллельно | отсутствует (обмен идет аппаратно) |

## ? Контрольные вопросы

1. Как можно определить, какие именно платы и устройства установлены в вашем компьютере? Для чего это может потребоваться?
2. Как вы думаете, что более полезно для глубокого понимания работы компьютера: изучение функционального устройства компьютера или изучение его конструкции?
3. Как устройства компьютера обмениваются данными?
4. Что такое шина? Почему обмен данными между устройствами компьютера с помощью шины оказался наилучшим решением?
5. Из каких частей состоит шина? Охарактеризуйте каждую из них.
6. Что такое магистрально-модульная архитектура и в чем ее главное достоинство?

7. В чем заключается принцип открытой архитектуры?
8. Используя приведенное в тексте объяснение процесса записи данных в память, попробуйте объяснить, как происходит считывание данных из ячейки памяти с заданным адресом.
9. Что такое контроллер и для чего он нужен?
10. Объясните, как использование контроллеров позволяет повысить быстродействие компьютера в целом.
11. Сравните магистрально-модульную архитектуру компьютера с классической. Выделите наиболее перегруженный блок на каждой из схем.
12. Почему в современном компьютере несколько шин?
13. Что требуется для успешного присоединения к компьютеру нового устройства?
14. Расскажите о разных режимах обмена данными с внешними устройствами.
15. Как расшифровывается сокращение ПДП и что это такое?
16. Как выполняется обмен данными в режиме ПДП?
17. Предложите наиболее подходящий режим обмена данными с клавиатурой. (Ответ: лучше всего подходит обмен по прерываниям; при программированном обмене ЦП будет тратить слишком много времени на напрасный опрос клавиатуры, а для ПДП у «медленной» клавиатуры просто не хватит данных.)
18. Какой режим лучше всего подходит для обмена данными с жестким диском? (Ответ: режим ПДП.)
19. Где в программировании применяются принципы обработки прерываний?

## 5.4. Процессор



Вид микропроцессора со стороны выводов

Центральным устройством, во многом определяющим возможности компьютера, является процессор.

**Процессор** – это блок, предназначенный для автоматического считывания команд программы, их расшифровки и выполнения.

Название «процессор» происходит от английского глагола «*to process*» – обрабатывать. Иными словами, процессор – это блок компьютера, который автоматически обрабатывает информацию по заданной программе.[24]

Процессор, изготовленный в виде большой или сверхбольшой интегральной схемы (БИС, СБИС), называется *микропроцессором*.

Любой процессор обязательно включает в себя две важные части, каждая из которых решает свои задачи:

- *арифметико-логическое устройство* (АЛУ), выполняющее обработку данных, и
- *устройство управления* (УУ), которое управляет выполнением программы и обеспечивает согласованную работу всех узлов компьютера.

### 5.4.1. Арифметико-логическое устройство

В простейшем случае АЛУ состоит из двух регистров, сумматора и схем управления операциями (об устройстве сумматора и регистров можно прочитать в главе 3). При выполнении операций в регистры помещаются исходные данные, а в сумматоре они складываются.

Как показано в главе 4, все остальные арифметические операции могут быть тем или иным способом сведены к сложению. Тем не менее, нередко для ускорения умножения и деления инженеры идут на усложнение АЛУ. Например, в процессорах широко использу-

[24] Не следует путать процессор как аппаратный блок с мощными программами обработки, которые также часто называют процессорами (например, текстовый или табличный процессор).

ется метод умножения чисел с использованием таблиц, в которых записаны готовые произведения небольших чисел.

АЛУ не только выполняет вычисления, но и анализирует полученный результат. Обычно проверяется два свойства: равенство нулю (совпадение всех разрядов сумматора с нулем) и отрицательность результата (определяется проверкой знакового разряда – см. главу 4). Результаты этого анализа заносятся в определенные биты *регистра состояния* процессора. Используя эти значения, можно сделать вывод об истинности или ложности условий $R = 0$, $R \neq 0$, $R > 0$, $R < 0$, $R \geq 0$, $R \leq 0$, где R обозначает результат операции. Это позволяет организовать ветвления в программе, например, для неотрицательного числа вычислять квадратный корень, а иначе – выдать сообщение об ошибке.

Как правило, АЛУ работает только с целыми числами. Операции с вещественными числами выполняются в *математическом сопроцессоре*, который встроен внутрь современных микропроцессоров.

### 5.4.2. Устройство управления

Главная задача устройства управления – обеспечить автоматическое выполнение последовательности команд программы в соответствии с основным алгоритмом работы процессора (см. 5.2.3). УУ выполняет следующие действия:

- извлечение из памяти очередной команды;
- расшифровка команды, определение необходимых действий;
- определение адресов ячеек памяти, где находятся исходные данные;
- занесение в АЛУ исходных данных;
- управление выполнением операции;
- сохранение результата.

Таким образом, выполнение каждой машинной команды состоит из элементарных действий, которые называются *микрокомандами*.

В зависимости от сложности, машинная команда может быть выполнена за различное число микрокоманд. Например, пересылка числа из одного внутреннего регистра микропроцессора требует значительно меньшего числа действий, чем умножение. Команды, работающие с оперативной памятью, выполняются дольше, чем команды, работающие только с регистрами процессора.

Каждая из микрокоманд машинной инструкции запускается с помощью управляющего импульса. Опорную последовательность импульсов для этих целей УУ получает от *генератора тактовых импульсов*. Интервал между двумя соседними импульсами называется *тактом*.

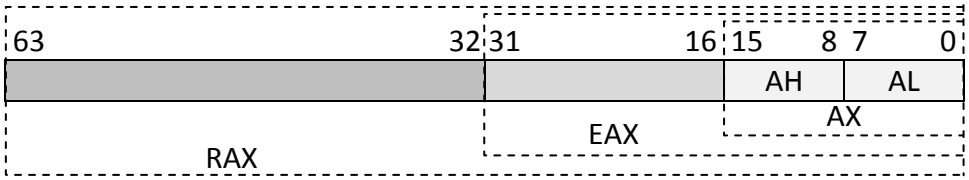
Если две микрокоманды полностью независимы друг от друга, то их можно выполнить одновременно (за один такт), даже если они принадлежат к разным командам программы. Такая оптимизация широко применяется в современных процессорах для организации конвейерной обработки ради увеличения быстродействия.

### 5.4.3. Регистры процессора

Кроме регистров АЛУ и УУ, в микропроцессоре есть много других регистров. Большинство из них – внутренние, они недоступны программисту. Однако есть несколько регистров, специально предназначенных для использования программным обеспечением. Их часто называют *регистрами общего назначения* (РОН), подчеркивая тем самым универсальность их функций. В РОН могут храниться не только сами данные (числа, коды символов и т.д.), но и адреса ячеек памяти, где эти данные находятся. Например, если требуется обработать последовательные ячейки памяти, то к содержимому такого регистра нужно каждый раз прибавлять размер ячейки.

Количество регистров и их устройство в разных процессорах отличается друг от друга. Например, в процессорах семейства *Intel* имеется небольшой набор 64-разрядных

РОН. Ради обеспечения программной совместимости со старыми (32- и 16-разрядными) процессорами, эти РОН имеют вложенную структуру, напоминающую матрешку:




На рисунке показана структура 64-разрядного регистра RAX. Его младшие 32 бита (с нулевого по 31-й) образуют регистр EAX для 32-разрядных вычислений. К младшим 16 битам EAX (0-15), в свою очередь, можно также обращаться как к самостоятельному регистру AX. Наконец, биты 0-7 и 8-15 образуют два 8-разрядных регистра AH и AL. Отчетливо видно, что наращивание разрядности процессоров семейства Intel происходило постепенно. Такая структура регистров обеспечивает совместимость с предыдущими моделями и позволяет процессору легко обрабатывать 8-, 16-, 32- и 64-разрядные данные.

Кроме рассмотренного выше регистра RAX, в процессорах Intel имеются также аналогичным образом устроенные регистры RBX, RCX и RDX, а также некоторые другие. Это регистры неравноценны, по справочникам можно определить, как и с каким регистром работает та или иная команда.

### 5.4.4. Основные характеристики процессора

Как вы уже знаете, для организации выполнения команд в компьютере есть генератор импульсов, каждый из которых «запускает» очередной такт машинной команды. Очевидно, что чем чаще следуют импульсы от генератора, тем быстрее будет выполняться операция. Следовательно, *тактовая частота*, измеряемая количеством тактовых импульсов в секунду, может быть характеристикой быстродействия процессора.

**Тактовая частота** – количество тактовых импульсов за одну секунду.

В настоящее время тактовая частота измеряется в гигагерцах, т.е. в миллиардах ($10^9$) импульсов за секунду. Эту частоту нельзя установить сколь угодно высокой, поскольку процессор может просто не успеть выполнить действие очередного такта до прихода следующего импульса.

Нужно понимать, что использовать тактовую частоту для сравнения быстродействия процессоров можно только в том случае, если оба процессора устроены одинаково. Например, если какая-то команда в одном из процессоров выполняется за два такта, а в другом – за три, то при равенстве частот первый будет работать в полтора раза быстрее.

Приближенно можно считать, что процессор выполняет за один такт одну простую команду (типа пересылки регистр-регистр). Тогда при тактовой частоте 4 ГГц за 1 сек выполняется около 4 миллиардов таких операций. Это примерная оценка, потому что при конвейерном методе скорость выполнения команд сильно зависит от множества факторов, например, от порядка следования команд в программе.

Другая характеристика, позволяющая судить о производительности процессора, – это его *разрядность*.

**Разрядность** – это максимальное количество двоичных разрядов, которые процессор способен обрабатывать за одну команду.

Чаще всего разрядность определяют как размер регистров процессора в битах.

Однако, важны также разрядности шины данных и шины адреса, которые поддерживает процессор. *Разрядность шины данных* – это максимальное количество бит, которое может быть считано за одно обращение к памяти. *Разрядность шины адреса* – это количество адресных линий; она определяет максимальный объем памяти, который способен поддерживать процессор. Этот объем памяти часто называют величиной *адресного пространства*, он вычисляется по формуле $2^R$, где $R$ – количество разрядов шины адреса.

Все три разрядности могут не совпадать. Так, у процессора Pentium II были 32-разрядные регистры, разрядность шины данных – 64 бита, а шины адреса – 36 бит.

### 5.4.5. Система команд процессора

Каждая модель процессора имеет собственную систему команд. Поэтому, как правило, процессоры могут выполнять только программы, написанные специально для них. Тем не менее, обычно новые процессоры одной и той же серии (например, процессоры *Intel*) поддерживают все команды предыдущих моделей.

В системах команд разных процессоров есть много общего. Они обязательно включают следующие группы машинных команд:

- команды передачи (копирования) данных;
- арифметические операции;
- логические операции, например, «НЕ», «И», «ИЛИ», «исключающее ИЛИ»;
- команды ввода и вывода;
- команды переходов.

Существует два основных подхода к построению системы команд процессора:

- процессоры с полным набором команд (англ. *CISC = Complex Instruction Set Computer*);
- процессоры с сокращенным набором команд (англ. *RISC = Reduced Instruction Set Computer*).

CISC-процессоры содержат широкий набор разнообразных команд. При этом на скорость их выполнения обращают меньшее внимание, главное – удобство программирования. При разработке RISC-процессоров набор команд, наоборот, весьма ограничен, но это позволяет значительно ускорить их выполнение. Многие современные процессоры (например, процессоры *Intel*) – гибридные, у них полный набор команд, которые выполняются RISC-ядром. Это позволяет совместить достоинства обоих подходов.

Почти все инструкции, входящие в систему команд компьютера, состоят из двух частей – *операционной* и *адресной*. Операционная часть – *код операции* – указывает, какое действие необходимо выполнить. Адресная часть описывает, где хранятся исходные данные и куда поместить результат. Часто исходные данные для команды (содержимое регистра или ячейки памяти, константа) называют *операндами*.

Рассмотрим для примера одну из наиболее простых команд процессора Intel, которая состоит из четырех байт и имеет шестнадцатеричный код **`81 C2 01 01`**. Она может быть разбита на три неодинаковые по длине части:

- код операции **`81C`** обозначает сложению регистра с константой;
- первый операнд **`2`** – это условное обозначение регистра DX;
- константа **`0101`**, которая добавляется к регистру.

Отметим, что система команд процессоров *Intel* очень сложна и плохо подходит для изучения в школьном курсе информатики.

### ? Контрольные вопросы

1. Для чего нужен процессор? Почему он так называется?
2. Какие узлы входят в состав процессора? Зачем нужны АЛУ и УУ?
3. Как выполняется АЛУ в простейшем случае? Как в АЛУ используется сумматор?
4. Почему удобно, что АЛУ автоматически сравнивает результат действия с нулем?
5. Подумайте, как с помощью логических операций с битами сумматора установить факт его равенства или неравенства нулю.
6. Для чего служит математический сопроцессор?
7. Какую роль играет УУ в автоматическом выполнении программ?
8. Как называется элементарное действие в машинной команде?
9. Зачем нужен генератор тактовых импульсов?

10. Что такое РОН? Для каких целей он может использоваться?
11. *Найдите информацию о регистрах процессора *Intel*. Постарайтесь разобраться в назначении наиболее важных из них.
12. Что такое тактовая частота и как она влияет на быстродействие компьютера?
13. Тактовые частоты двух процессоров, изготовленных фирмами *Intel* и *AMD* равны. Означает ли это, что их быстродействие одинаково?
14. *Объясните, как применение конвейера влияет на количество команд, выполняемых за один такт.
15. На что влияет разрядность процессора? Какие разновидности разрядности вы знаете? Что характеризует каждая их них?
16. Какие группы операций входят в систему команд любого процессора?
17. Что такое RISC- и CISC-процессоры? Чем они отличаются?
18. Какие части можно выделить в команде процессора?

### Задачи

1. Обозначим символом Z бит, определяющий факт равенства результата R нулю, а символом N – бит, фиксирующий отрицательность R:
   - Z=1 при R=0, и Z=0 в противном случае;
   - N=1 при R<0, и N=0 в противном случае.

   Напишите логическое выражение, включающее значения N и Z, которое дает:
   а) 1 при R≤0, и 0 в противном случае;
   б) 1 при R>0, и 0 в противном случае.
2. Оцените, сколько миллиардов простых операций типа пересылки регистр-регистр может выполнить за 1 мин. процессор с тактовой частотой 1 ГГц.
3. Сопоставьте тактовую частоту процессора с максимальной частотой звуковых колебаний, которые слышит человек. Что можно сказать о возможностях современного компьютера в обработке звуковой информации?
4. Какое максимальное десятичное целое число без знака можно поместить в 32-разрядный регистр? (Ответ: $2^{32}-1 = 4\ 294\ 967\ 295$)
5. Сколько символов, закодированных в двухбайтной кодировке UNICODE, можно загрузить одновременно в 64-разрядный регистр?
6. Процессор Pentium II имеет 36-разрядную шину адреса. Какой объем памяти он может адресовать? (Ответ: 64 Гб)

## 5.5. Память

Как мы уже знаем, процессор способен выполнять программу, но ее команды хранятся в памяти. Таким образом, память – это другое устройство, без которого вычислительный автомат не может быть построен. Кроме того, память одновременно используется (что не менее важно) и для хранения обрабатываемых данных.

**Память** – это устройство компьютера, которое используется для записи, хранения и выдачи по запросу команд программы и данных.

Существует большое количество видов памяти, которые различаются по устройству, организации, функциям и т.д. Обычно выделяют *внутреннюю* и *внешнюю* память. Термины эти имеют историческое происхождение, связанное с конструкцией первых ЭВМ: одна часть памяти находилась внутри главного шкафа (в котором размещался процессор), а другая – вне его.

Современные компьютеры, конечно, выглядят совсем по-другому, из-за чего названия утратили свою прежнюю наглядность. Тем не менее, деление памяти на два типа по-прежнему сохраняется. Различие между ними кроется, прежде всего, в назначении. Внутренняя память предназначена для хранения программ и данных, которые используются для задач, решаемых в данный момент. А внешняя память служит для того, чтобы сохра-

нить данные на длительный срок, пока они не потребуются, именно поэтому ее еще часто называют *долговременной*.

### 5.5.1. Внутренняя память

**Внутренняя память** – часть памяти компьютера, которая используется для хранения программ и данных во время решения задачи.

Часто ее называют *основной памятью*. В состав внутренней памяти входят ОЗУ и ПЗУ.

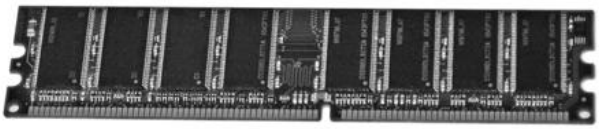
Внутренняя память строится в соответствии с базовыми принципами, описанными ранее в п.5.2. Основное отличие внутренней памяти от внешней – произвольный доступ к отдельным ячейкам памяти по их адресам (обращение к внешней памяти происходит иначе, см. далее).

Информация, хранящаяся в ОЗУ, считается временной (оперативной), поэтому пользователь должен сам сохранять необходимые данные во внешней памяти.

Часто говорят, что при выключении питания информация в ОЗУ пропадает. Строго говоря, это не совсем правильно, поскольку существуют элементы памяти, способные сохранять свое состояние даже после отключения питания. Однако при повторном включении (или перезагрузке) компьютера программное обеспечение неспособно восстановить, где и какая информация находилась «в прошлый раз». Именно поэтому если при наборе текста перезагрузить компьютер, работу придется повторять заново.

Магнитное ОЗУ: слева – биты памяти, справа – устройство выборки нужного адреса

Внутренней память может быть построена на основе самых разных технологий. Самые первые ЭВМ имели ОЗУ на электронно-лучевых трубках, причем их количество соответствовало разрядности памяти (каждый бит числа считывался из отдельной трубки). Затем появилась память на магнитных сердечниках. Намагниченное состояние сердечника соответствовало единичному состоянию бита, ненамагниченное – нулевому. Заметим, что данные в магнитных ячейках памяти полностью сохранялась и после выключения питания. Наконец, развитие микроэлектроники позволило изготовить компактную полупроводниковую память, которая сейчас и применяется в ПК.



Модуль полупроводниковой памяти

Существуют два типа оперативной памяти, отличающиеся по технологии изготовления – *статическая* и *динамическая*. Первая строится на триггерах (об устройстве триггера говорилось в главе 3), а вторая – на полупроводниковых конденсаторах. Конденсатор намного проще и меньше триггера, так что на одном и том же кристалле можно сделать гораздо больше запоминающих элементов динамического типа, чем статического. Поэтому динамическая память имеет б*о*льшую емкость и меньшую стоимость, чем статическая. К сожалению, у нее есть очень существенный недостаток: она работает намного медленнее статической. Сейчас в персональных компьютерах используется динамическая оперативная память.

Что касается ПЗУ, то технологии их изготовления также постепенно совершенствовались. Первоначально информация в ПЗУ заносилась только на заводе. Затем появились *программируемые* ПЗУ, которые потребитель мог заполнить сам, поместив «чистую» («пустую») микросхему в специальное устройство – *программатор*. В некоторых микросхемах этого типа в качестве запоминающих элементов использовали тонкие токопрово-

дящие перемычки. Наличие перемычки означало единицу. Программатор мощными импульсами тока пережигал нужные перемычки, тем самым устанавливая биты в нулевое состояние[25]. Очевидно, что процесс записи информации таким способом был необратимым.

Флэш-BIOS на плате компьютера

Позднее появились *перепрограммируемые* ПЗУ, в которых очистка информации сначала производилось ультрафиолетовыми лучами, а затем – с помощью электрических импульсов. Современные перепрограммируемые ПЗУ используют *флэш-память*. Каждый элемент такой памяти изготовлен на основе особой разновидности транзисторов, так что это тоже полупроводниковая память. Изменить содержимое такого ПЗУ можно даже без программатора, запустив специальную программу.

Внутри компьютеров семейства *IBM PC* есть еще один особый вид памяти – *память конфигурации* (CMOS-память). В ней хранятся разнообразные настройки аппаратного обеспечения, а также часы и календарь, благодаря которым компьютер всегда знает текущую дату и время. Данные сохраняются благодаря питанию от небольшой батарейки. CMOS-память – это особая память, которая не входит в адресное пространство внутренней памяти. Поэтому к ней невозможно обратиться просто по адресу, и в этом смысле она скорее похожа на внешнюю память. Для работы с памятью конфигурации в ПЗУ современного ПК предусмотрена специальная программа (она называется *BIOS Setup*), причем работать с ней можно только до загрузки операционной системы (при включении компьютера).

## 5.5.2. Внешняя память

**Внешняя память** – часть памяти компьютера, которая используется для долговременного хранения программ и данных.

Этот вид памяти позволяет повторно использовать программы и данные. Благодаря этому текст достаточно набрать один раз, а цифровые фотографии можно рассматривать в течение многих лет.

К внешней памяти относятся разнообразные устройства хранения данных, начиная от накопителей на магнитных дисках и кончая современными внешними запоминающими устройствами на основе полупроводниковой флэш-памяти.

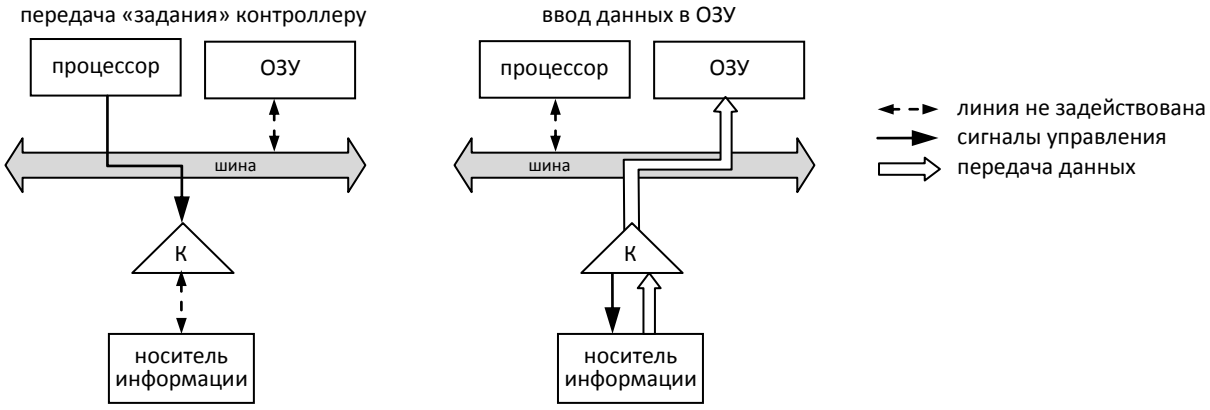
Любой тип внешней памяти состоит из некоторого носителя информации (например, диска или полупроводникового кристалла) и электронной схемы управления (*контроллера*).

*Машинный носитель информации* – это средство длительного хранения данных в компьютерном формате. Носитель может быть съемным (как в накопителях на оптических дисках), а может быть помещён внутрь неразборного устройства (жесткий магнитный диск – винчестер).

В переносных устройства внешней памяти, например, во внешних винчестерах и флэш-накопителях, носитель и схема управления объединены в единый блок. Такие устройства подключаются к компьютеру снаружи через разъем.

Центральный процессор не может непосредственно обращаться к данным на носителе, он работает с ними через контроллер внешней памяти. На рисунке схематично показано, как читаются данные с внешнего носителя информации в ОЗУ[26].

[25] Так сгорают плавкие предохранители в бытовой аппаратуре.
[26] В действительности процесс обмена более сложен, в нём участвует еще и контроллер ПДП.

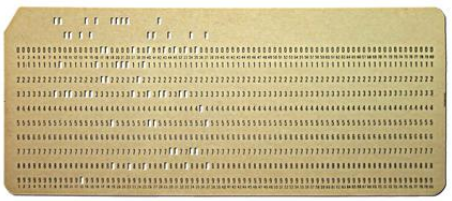



Для связи с контроллером процессор использует *порты* – регистры контроллера, к которым процессор может обратиться по номеру. Процессор передает контроллеру «задание» на передачу данных, и контроллер берет руководство процессом на себя. В это время центральный процессор может параллельно выполнять программу дальше или решать другую задачу. Таким образом, выполнить чтение (и запись) данных из внешней памяти гораздо сложнее, чем из внутренней памяти.

Для внешней памяти характерны следующие черты:

- обменом данными управляют контроллеры;
- прежде чем процессор сможет непосредственно использовать программу или данные, хранящиеся во внешней памяти, их нужно предварительно загрузить в ОЗУ;
- данные располагается блоками (на дисках их принято называть *секторами*); блок данных читается и пишется как единое целое, что существенно ускоряет процедуру обмена; работать с частью блока невозможна.

В качестве внешней памяти используются самые разные носители. Первоначально программы и данные сохранялись на бумажных *перфокартах* и *перфолентах*. Подписанные обычной ручкой или карандашом, они сортировались программистами вручную. Затем произошел переход к магнитным носителям: магнитным лентам, барабанам и дискам.


Перфокарта

На *магнитных дисках* биты данных хранятся в виде небольших намагниченных (или ненамагниченных) областей. Сектора размещаются на концентрических окружностях (имеющих общий центр), которые называются *дорожками*. Поскольку длина дорожки зависит от положения на диске, количество секторов на дорожках может быть неодинаковым. Доступ к секторам диска – произвольный, максимальная скорость достигается тогда, когда читаемые или записываемые сектора располагаются подряд.

Управление такой сложной системой очень трудоемко – поэтому, как нам уже известно из истории вычислительной техники, появление магнитных дисков вызвало создание специального ПО для работы с ними – операционных систем (ОС). ОС берет на себя все технические детали, предоставляя пользователю работу с некоторыми наборами данных – *файлами*. Таким образом, начиная с дисковых накопителей, наличие файловой системы – это характерная черта внешней памяти, которая существенно отличает ее от внутренней.

Следующей технологией хранения информации стали *оптические компакт-диски* (англ. *CD = Compact Disk*). При записи данных (одним из способов) луч лазера «выжигает» на поверхности диска дорожку, в которой чередуются впадины и возвышения. При считывании также применяется луч лазера, только меньшей интенсивности, чтобы не разрушить данные. Для распознавания нулей и единиц используется различное отражение от перепадов глубины и ровной поверхности диска. В отличие от магнитных дисков, где ин-

формация хранится в виде на отдельных замкнутых дорожках, данные на оптическом диске записываются вдоль непрерывной спирали, как на старых грампластинках[27].

Сейчас широко используются оптические диски следующих поколений: DVD (англ. *Digital Versatile Disk* – цифровой многоцелевой диск, емкость до 17 Гбайт) и *Blu-ray*-диски (емкостью до 66 Гбайт). Они имеют тот же диаметр, что и *CD*-диски, но для повышения плотности записи используют лазер с меньшей длиной волны.

Были разработаны также комбинированные магнитооптические диски. Носителем информации в них служит магнитное вещество, которое при нагреве лазером ориентируется в магнитном поле и меняет оптические свойства поверхности диска. После восстановления нормальной температуры такие диски необычайно устойчивы к внешним воздействиям. Тем не менее, они не получили распространения из-за высокой стоимости и малой скорости записи.

Отметим, что на всех видах дисков есть разметка на сектора, благодаря которой контроллер может быстро находить нужную информацию. Сами данные помещаются между «заголовком» сектора и его завершающей записью.

Наконец, последнее достижение в области устройств внешней памяти – запоминающие устройства на базе флэш-памяти. В ней нет движущихся частей, а носителем информации служит полупроводниковый кристалл. Данные во флэш-памяти обновляются только блоками, но для устройств внешней памяти это вполне естественно. Максимальное количество перезаписей данных для каждого блока хотя и велико, но все же ограничено. Поэтому встроенный контроллер при записи использует специальный алгоритм для выбора свободных блоков, стараясь загружать сектора диска как можно более равномерно.

Флэш-карта

Кроме широко распространенных флэш-дисков («флэшек»), этот вид памяти используется в картах памяти для фотоаппаратов, плееров и сотовых телефонов, а также в твердотельных винчестерах (англ. *SSD = Solid State Disk*). Напомним, что ПЗУ также может изготовляться на базе флэш-памяти.

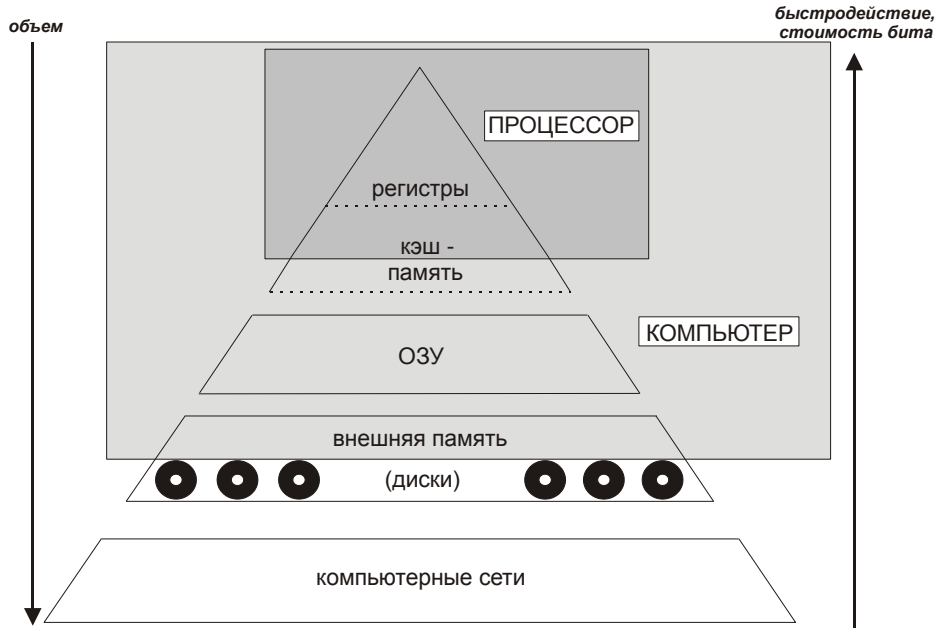
### 5.5.3. Взаимодействие разных видов памяти

Итак, мы познакомились с разными видами внутренней и внешней памяти. Осталось разобраться, как они взаимодействуют между собой.

**Иерархия памяти. Кэширование.** Как следует из обсуждения в п.5.2.2, невозможно создать память, которая имела бы как большой объем, так и высокое быстродействие. Поэтому используют многоуровневую (иерархическую) систему из нескольких типов памяти. Как правило, чем больший объем имеет память, тем медленнее она работает.

Самая быстрая (и очень небольшая) память – это регистры процессора. Гораздо больше по объему, но заметно медленнее, внутренняя память (ОЗУ и ПЗУ). Далее следует огромная, но еще более медленная внешняя память. Наконец, последний уровень – это данные, которые можно получить из компьютерных сетей.

[27] В отличие от грампластинок, спираль раскручивается от центра к краям.




Для редактирования файла с диска (внешняя память) программа обработки загружает его в ОЗУ (внутренняя память), а конкретные символы, с которыми в данные доли секунды работает процессор, «поднимаются» по иерархии выше – в регистры процессора.

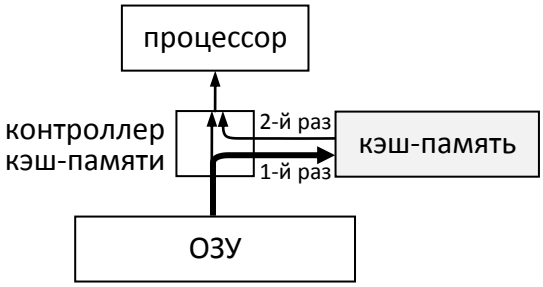
Производительность компьютера в первую очередь зависит от «верхних» уровней памяти – процессорной памяти и ОЗУ. Быстродействие процессоров значительно выше, чем скорость работы ОЗУ, поэтому процессору приходится ждать, пока до него дойдут данные из оперативной памяти. Чтобы улучшить ситуацию, между процессором и ОЗУ добавляют еще один слой памяти, который называют кэш-памятью (от англ. *cache* – тайник, прятать).

**Кэш-память** – это память, ускоряющая работу другого (более медленного) типа памяти, за счёт сохранения прочитанных данных на случай повторного обращения к ним.

Кэш-память – это статическая память, которая работает значительно быстрее динамического ОЗУ. В ней нет собственных адресов, она работает не по фон-неймановскому принципу адресности.

При чтении из ОЗУ процессор обращается к контроллеру кэш-памяти, который хранит список всех ячеек ОЗУ, копии которых находятся в кэше. Если требуемый адрес уже есть в этом списке, то запрашивать ОЗУ не нужно, и контроллер передает процессору значение, связанное (*ассоциированное*) с этим адресом[28]. Такой принцип организации памяти называется *ассоциативным*.

Если нужных данных нет в кэш-памяти, они читаются из ОЗУ, но одновременно попадают и в кэш – при следующем обращении их уже не нужно читать из ОЗУ.




Обычно в кэш-память заносится содержимое не только запрошенной ячейки, но и ближайших к ней (эта стрелка на рисунке показана более толстой линией). Таким образом, в кэше хранятся копии часто используемых ячеек ОЗУ, и передача этих данных в процессор происходит быстрее.

[28] Это напоминает поиск в Интернете содержимого документа по его названию.

В работе кэш-памяти есть две основные трудности. Во-первых, объем кэш намного меньше объема ОЗУ и он быстро заполняется – приходится заменять наиболее «ненужные» (например, редко используемые) данные. Во-вторых, если считанные из кэш-памяти данные обрабатываются процессором и сохраняются в ОЗУ, нужно обновлять и содержимое кэша. Обе эти задачи решает контроллер кэш-памяти. Несмотря на трудности, кэширование во многих случаях повышает скорость выполнения программы в несколько раз.

Сама кэш-память также строится по многоуровневой схеме: в современных процессорах есть, по крайней мере, 2-3 уровня. Некоторые из них входят в состав процессора, а остальные выполнены в виде отдельных микросхем (поэтому на схеме многоуровневой памяти кэш только частично расположен внутри процессора). Кэш для программы и для данных изготовляется раздельно. Это удобно потому, что считываемую программу, в отличие от данных, не принято изменять, поэтому кэш команд можно делать проще.

Подчеркнем, что термин «кэширование» в вычислительной технике имеет довольно широкий смысл: речь идет о сохранении информации в более быстродействующей памяти с целью повторного использования. Например, браузер кэширует файлы, полученные из Интернета, сохраняя их на винчестере в специальной папке. В накопителе на жестком диске также используется кэширование. Таким образом, кэш может быть организован как с помощью аппаратных средств (кэш процессора), так и программно (кэш браузера).

**Виртуальная память.** Пользователям хочется, чтобы программное обеспечение было интеллектуальным и дружественным, и чтобы в нем были предусмотрены все самые мелкие детали, которые им (пользователям) могут потребоваться. Программистам хочется написать программу с наименьшими затратами сил и времени, поэтому они широко используют среды быстрой разработки программ (англ. *RAD = Rapid Application Development*). В результате программы все больше увеличиваются в размере. Кроме того, объем обрабатываемых данных постоянно растет. Поэтому компьютерам требуется все больше и больше памяти, особенно в многозадачном режиме, когда одновременно запускается сразу несколько программ.

Как же согласовать эти требования с ограниченным объемом ОЗУ? Современные операционные системы используют для этого идею *виртуальной памяти*. Предполагается, что компьютер обладает максимально допустимым объемом памяти, а реально установленное ОЗУ – лишь некоторая *часть* этого пространства. Оставшаяся часть размещается в специальном системном файле или отдельном разделе жесткого диска. Если емкости ОЗУ не хватает для очередной задачи, система копирует «наименее нужную» (дольше всего не использовавшуюся) часть ОЗУ на диск, освобождая необходимый объем памяти. Когда, наоборот, потребуются данные с диска, они будут возвращены в освобожденное таким же образом место ОЗУ (и это совсем не обязательно будет то самое первоначальное место!).

При использовании виртуальной памяти выполнение программ замедляется, но зато они могут выполняться на компьютере с недостаточным объемом ОЗУ. В этом случае установка дополнительного ОЗУ может повысить быстродействие во много раз.

Использование виртуальной памяти еще раз подтверждает, что деление памяти на внутреннюю и внешнюю память – это искусственная мера. Она вызванная тем, что невозможно создать идеальную память, удовлетворяющую всем требованиям сразу.

## 5.5.4. Основные характеристики памяти

Для пользователя важны, прежде всего, объем памяти, ее быстродействие и стоимость.

**Информационная емкость** – это максимально возможный объем данных, который может сохранить данное устройство памяти.

Емкость памяти измеряется в тех же самых единицах, что и объем информации, т.е. в битах, байтах и производных единицах (чаще всего – в мегабайтах или гигабайтах).

Для дисков часто говорят о *форматированной* и *неформатированной* емкости. Первая величина – это объем «полезной» памяти, а вторая включает еще и ту область диска, которую занимает служебная разметка.

Для оценки *быстродействия* памяти используют несколько величин. Любая операция обмена данными включает не только саму передачу данных, но и подготовительную часть. Это может быть, например, поиск нужного сектора диска или установка адреса внутри микросхемы ОЗУ. Время подготовки соизмеримо со временем передачи, так что пренебрегать им нельзя. Общее время обмена данными от начала подготовки до окончания передачи называют *временем доступа*.

**Время доступа** – интервал времени от момента посылки запроса информации до момента получения результата на шине данных.

При измерении этой величины обычно рассматривают самый сложный случай, когда данные считываются или записываются в случайных местах памяти. На практике байты или сектора часто читаются по порядку, поэтому время ввода или вывода уменьшается.

Для ОЗУ время доступа измеряется в наносекундах (1 нс = $10^{-9}$ с), а для винчестеров – в миллисекундах (1 мс = $10^{-3}$ с). Такая разница связана с тем, что дисковод должен сначала переместить считывающую головку в нужное положение.

Поскольку устройства внешней памяти работают с целыми блоками данных, для их характеристики требуется какой-то дополнительный показатель.

**Средняя скорость передачи данных** – это количество передаваемых за единицу времени данных после непосредственного начала операции чтения (т.е. без учета подготовительной стадии).

Эта характеристика обычно измеряется в мегабайтах в секунду (Мбайт/с).

Для оценки стоимости памяти используют отношение стоимости модуля памяти к его информационной емкости. Часто говорят о стоимости одного бита или *стоимости одного гигабайта*.

Для дисковых накопителей часто указывают *частоту вращения* (в оборотах в минуту). Чем быстрее вращается диск, тем выше может быть скорость считывания и записи.

## ? Контрольные вопросы

1. Зачем в компьютере нужна память?
2. С какой целью память делится на внутреннюю и внешнюю?
3. Верно ли, что внешняя память располагается вне корпуса компьютера? Приведите примеры. (Ответ: нет, например, жесткий диск находится внутри системного блока).
4. Назовите все виды компьютерной памяти, которые вы знаете. Зачем они используются? Какими свойствами обладают?
5. К каким видам памяти применим принцип адресности фон Неймана?
6. Что означает термин «произвольный доступ к памяти»?
7. Зачем нужно ПЗУ в компьютере? Можно ли при необходимости изменить его содержимое на домашнем компьютере?
8. Что такое носитель информации? Какие носители вы можете назвать?
9. Какими носителями внешней памяти вы пользовались? Каков их объем и какую примерно его часть вы использовали?
10. Перечислите все известные вам виды дисков.
11. Что такое сектор диска?
12. Можно ли считать с диска отдельно взятый байт? Как его все-таки получить? (Ответ: только программным путем: считать весь сектор в буфер в ОЗУ и скопировать оттуда требуемый байт).
13. Какую роль играет контроллер при считывании данных с диска?

14. Почему любую программу перед выполнением требуется загрузить в оперативную память?
15. Что такое флэш-память, и в каких запоминающих устройствах она используется?
16. Какие разновидности полупроводникового ОЗУ существуют? Что служит в них запоминающим элементом? Каковы свойства названных вами типов ОЗУ?
17. Для чего создана кэш-память, как она работает и как повышает производительность компьютера?
18. Может ли программа обращаться к ячейкам кэш-памяти? Подумайте, относится ли кэш-память к архитектуре компьютера? Почему? (Ответ: не может; детали, недоступные программисту, обычно не относят к архитектуре.)
19. *Почему кэш называют ассоциативной памятью? Сравните с человеческой памятью, которую тоже часто называют ассоциативной.
20. Кэширование – это аппаратный или программный прием? Приведите примеры.
21. Перечислите все известные вам уровни иерархии компьютерной памяти и кратко охарактеризуйте их. Как меняются объем и быстродействие памяти при переходе на другой уровень иерархии (вверх или вниз)?
22. Почему компьютерам требуются все большие объемы памяти?
23. Как работает механизм виртуальной памяти?
24. *Чем ограничен объем виртуальной памяти?
25. Какие основные характеристики используются для памяти? В каких единицах они измеряются?
26. Какая характеристика используется только для внешней памяти?

### *Задачи*

1. Определите информационный объем каждого вида памяти в вашем домашнем компьютере (ОЗУ, кэш-память, жесткий диск, примерный суммарный объем CD-дисков с данными и т.п.). Оцените отношение объемов этих уровней памяти.
2. Сравните приведенные в тексте данные о времени доступа для ОЗУ и дисковых накопителей: на сколько порядков они различаются?
3. Пусть емкость жесткого диска составляет 137 438 953 472 байт. Пользуясь калькулятором, переведите этот объем в гигабайты по правилам, принятым в информатике. Сравните ответ с тем числом, которое получится, если коэффициенты при переводе принять равными 1000 (так делают, например, в физике: в 1 кг ровно 1000 г!). Какие конфликты в связи с этим могут возникнуть (и реально возникали!) у потребителей и фирм-изготовителей жестких дисков? (Ответ: объем составляет 128 Гб при делении на 1024 и 137 Гб при «десятичной» оценке, т.е. отличие превышает 7%)

## 5.6. Устройства ввода

### 5.6.1. Что относится к устройствам ввода?

Информация в компьютер может вводиться с помощью самых разнообразных устройств, но не каждое из них называют устройством ввода. Например, *не являются* устройствами ввода устройства внешней памяти, рассмотренные в предыдущем разделе. Прием данных по сети также не является вводом, поскольку здесь (как и в случае внешней памяти) данные уже были введены ранее и сохранены в компьютерном формате, а теперь просто копируются.

Но, даже исключив из рассмотрения все перечисленные устройства, мы по-прежнему будем иметь довольно пёструю картину. Сравните между собой подключенный к компьютеру датчик температуры, веб-камеру, осуществляющую съемку для круглосуточной трансляции городского пейзажа в Интернет, мышку, которой пользователь запус-

кает программы или выбирает из меню требуемое действие, и, наконец, клавиатуру, с помощью которой можно не только набрать текст, но и отдавать команды компьютеру.

**Устройством ввода** называется устройство, которое:
- позволяет человеку отдавать компьютеру команды и/или
- выполняет первичное преобразование данных в форму, пригодную для хранения и обработки в компьютере.

К устройствам ввода относятся
- клавиатура;
- манипуляторы: мышь, трекбол, сенсорная панель, джойстик, трекпойнт;
- сканер;
- микрофон, видеокамера и другие источники мультимедийных данных;
- световое перо и графический планшет – специализированные устройства ввода графической информации;
- датчики.

Заметим, что некоторые устройства ввода, например, датчики и веб-камеры, работают без непосредственного участия человека.

## 5.6.2. Клавиатура

Одним из первых устройств ввода была клавиатура. С ее помощью человек вводит в компьютер текст. Текст может быть записью числа: тогда компьютер по программе преобразует текстовую строку в соответствующее двоичное число, с которым может работать процессор.

Кроме символьных клавиш, на клавиатуре есть *дополнительные* (управляющие) клавиши. Значения некоторых из них жёстко задано (например, клавиши управления курсором, *Page Up, Home, Delete, Print Screen* и др.), функции других (в первую очередь, функциональных клавиш *F1-F12*) программист может назначить сам. Клавиши *Shift, Caps Lock, Ctrl* и *Alt* изменяют результат нажатия остальных клавиш. С их помощью можно, например, вводить заглавные буквы.

В простейших типах клавиатур при нажатии клавиши соединяются два контакта и замыкается электрическая цепь. Роль контактов в наиболее распространенных моделях играет специальное токопроводящее напыление, наносимое на гибкую изолирующую полимерную пленку. Более качественные клавиатуры могут использовать, например, *герконы* (герметичные контакты), срабатывающие от приближающегося к ним магнита. Еще один вариант – это емкостные клавиатуры, где при нажатии клавиши сближаются две небольшие пластины, образующие конденсатор. Емкостные клавиатуры более долговечны, так как в них нет механического контакта деталей.

Работой современной клавиатуры руководит встроенный в нее микроконтроллер, который:
- опрашивает все клавиши и фиксирует изменение их состояния: нажатие или отпускание;
- временно (до момента передачи в центральный процессор) хранит коды нескольких последних нажатых или отпущенных клавиш (*скан-коды*)[29];
- при наличии данных посылает требование прерывания центральному процессору и затем (по его запросу) передает имеющиеся данные;
- управляет световыми индикаторами клавиатуры;
- выполняет диагностику неисправностей клавиатуры.

Контроллер клавиатуры выполняет лишь минимальную обработку информации: в компьютер уходят исключительно данные о нажатии или отпускании клавиши с заданным

[29] Скан-коды представляют собой номера клавиш и не имеют ничего общего с кодовыми таблицами символов, изученными в главе 2

номером. Распознавание кода набранного символа с учетом состояния клавиш сдвига выполняет программа, принимающая данные. Такое решение в очередной раз показывает, что аппаратная часть компьютера всегда делается максимально универсально, а все особенности работы компьютера определяются программным обеспечением.

Клавиатура имеет определенные технические характеристики, такие как усилие нажатия клавиш (в ньютонах) и ход клавиш (в миллиметрах).

### 5.6.3. Манипуляторы

Для ввода команд и данных в компьютер широко используются *манипуляторы* – разнообразные по конструкции устройства, воздействуя на которые (путем их перемещения, давления на их чувствительную поверхность и т.п.), пользователь может управлять компьютером, не набирая текста.

Общая идея работы манипуляторов заключается в следующем. На экране монитора отображается указатель (*курсор*), с помощью которого человек может «показать» компьютеру интересующий его объект. Манипулятор перемещает курсор вслед за определенными перемещениями руки человека. Когда курсор установлен в нужное место экрана, человек сообщает об этом компьютеру, обычно нажимая кнопку на манипуляторе. В принципе манипулятор позволяет даже набирать тексты, используя нарисованную на экране клавиатуру.

Самый распространенный манипулятор – *компьютерная мышь*. Это название принято связывать с кабелем («хвостом»), соединяющим устройство с компьютером. Многим современным мышам «хвост» уже не нужен: они передают данные о своем движении с помощью электромагнитных волн (к компьютеру при этом подсоединяется специальное устройство для приема и декодирования радиоволн). Такие мыши более удобны, хотя стоят дороже и используют дополнительные элементы питания (батарейки или аккумуляторы).

Первоначально датчики движения мыши были механическими: при перемещении мыши вращался находящийся внутри нее шарик. Шарик, в свою очередь, вращал два взаимно перпендикулярных колесика, и их поворот фиксировался электронным устройством. Полученная информация об изменении координат передавалась в компьютер. Такая механическая конструкция была неудобна, так как шарик и колесики приходилось часто очищать от пыли и грязи.

*Оптические* мыши, которые используются сейчас, не содержат механических частей, поэтому они долговечны и обладают высокой точностью. Расположенная «под брюхом» миниатюрная видеокамера снимает изображение поверхности стола через небольшие промежутки времени (для подсветки используется светодиод или портативный лазер). Сравнивая полученные картинки, специальный микропроцессор вычисляет перемещение мыши по двум осям координат. Этот метод даёт плохие результаты, когда поверхность очень гладкая и однородная (например, стекло). В таких случаях значительно лучше работают лазерные мыши, потому что подсветка лазером дает более контрастное изображение.

Наиболее интересная характеристика оптической мыши – это *разрешение оптического сенсора* (видеокамеры). Оно определяется как количество точек, которые способно различить устройство на отрезке заданной длины. Чем выше разрешение, тем точнее мышь способна отслеживать перемещение (это важно, например, при точной обработке изображений в графическом редакторе). Разрешение обычно измеряется в точках на дюйм (англ. ***dpi*** = ***d****ots* ***p****er* ***i****nch*). Обычное разрешение мыши – около 1000 dpi, а у некоторых особо «точных» экземпляров – в несколько раз больше.

Кроме разрешения, на качество работы мыши влияет количество кадров, которые делает видеокамера за одну секунду (до десяти тысяч). Размеры каждого кадра определя-

ются датчиком, обычно они находятся в пределах от 16×16 до 30×30 пикселей[30]. Зная эти данные, можно найти скорость обработки изображения в мегапикселях в секунду (Мп/с). Для игровых мышей важна также *максимальная скорость движения* – она может достигать нескольких метров в секунду.

Так выглядит трекбол (*www.mousearena.com*)

Шаровой манипулятор – *трекбол* – это перевернутая мышь. Его чувствительный элемент – закрепленный шар, который вращается вокруг своего центра. Название «трекбол» происходит от английских слов *track* – направляющее устройство и *ball* – шар. Для портативных компьютеров он удобнее мыши, потому что не требует дополнительного ровного пространства. Кроме того, трекболы могут работать там, где есть вибрация. Сейчас трекболы практически не используются.

В ноутбуках в качестве встроенного «заменителя» мыши устанавливают еще один тип манипулятора – *сенсорную панель* (англ. *touchpad*), воспринимающая движение по ней пальца. Панель состоит из небольшой чувствительной к давлению поверхности и двух кнопок. Короткое касание чувствительной панели заменяет щелчок мышью (можно использовать также кнопки рядом с панелью). Современные панели способны воспринимать не только перемещение, но и другие команды. Например, для прокрутки документа можно проводить пальцем вдоль правой или нижней границы панели (там, где в окне принято располагать полосы прокрутки). Некоторые панели даже способны анализировать сочетания движений по ним двух пальцев.

«Менее серьезный» манипулятор – *джойстик* (англ. *joy stick* – «весёлая» рукоятка) – используется, в основном, в компьютерных играх и может быть оформлен самым причудливым образом. Джойстик имеет ручку, при повороте которой внутри корпуса замыкаются контакты, соответствующие направлению наклона ручки. В некоторых моделях дополнительно установлен датчик давления, и чем сильнее пользователь наклоняет ручку, тем быстрее движется указатель по экрану.

В некоторых ноутбуках в середине клавиатуры устанавливается *трекпойнт* (это слово можно перевести с английского как указатель курса или маршрута). Трекпойнт – это кнопка, которая определяет направление давления пальца и преобразует эту команду в перемещение курсора на экране.


Трекпойнт (*www.globalnerdy.com*)

## 5.6.4. Сканер

**Сканер** – это устройство для ввода в компьютер графической информации.

С его помощью можно преобразовать в компьютерные данные рисунки, фотографии, снимки на фотоплёнке (негативы и слайды), а также получить снимки объектов не слишком большой толщины.

Часто при помощи сканера в компьютер вводят офисные документы. При этом сканер передает в компьютер изображение документа в виде картинки. Чтобы отсканированный текст можно было редактировать, нужно превратить эту картинку в коды символов. Для этого используют программы оптического распознавания символов (англ. *OCR = Optical Character Recognition*). OCR-программы пытаются «угадать» в пикселях рисунка очертания букв, и определить, какие это буквы, сверяя контуры с имеющимися у них об-

[30] http://computer.howstuffworks.com/mouse5.htm, http://home.comcast.net/~richardlowens/OpticalMouse/

разцами. В принципе, можно распознавать и рукописный текст, однако программы справляются с ним значительно хуже, чем с печатным (подумайте почему?).

Принцип работы сканера достаточно прост. Луч света от яркого источника пробегает вдоль сканируемой поверхности, а светочувствительные датчики при этом воспринимают отраженные лучи и определяют их интенсивность и цвет. Можно сказать, что сканер – это очень сильно упрощенный цифровой фотоаппарат.

Сканеры могут иметь разную конструкцию:

- *ручные*, где считывающую головку перемещает пользователь (вспомните, как считываются штрих-коды);
- *планшетные*, в которых неподвижный объект кладется на стекло, а светочувствительная головка перемещается внутри сканера;
- *рулонные*, протягивающие бумагу с изображением мимо неподвижной головки;
- *барабанные*, где сканируемый объект наклеивается на вращающийся барабан, который медленно вращается мимо неподвижной головки; такие сканеры обеспечивают наилучшее качество сканирования и применяются в издательской деятельности.

Сканеры часто объединяют в одном корпусе с лазерным принтером, копировальным аппаратом и факсом – получается *многофункциональное устройство* (МФУ).

Самая важная характеристика сканера – разрешающая способность.

**Разрешающая способность** – это максимальное количество точек на единицу длины, которые способен различить сканер.

Разрешающая способность сканера измеряется в пикселях на дюйм (англ. *ppi = pixels per inch*). При сканировании совсем не обязательно устанавливать максимально возможное разрешение. Конечно, чем оно выше, тем лучше качество, но зато и файл займет больше места на диске! Рекомендуемое разрешение зависит от того, зачем сканируется материал (см. таблицу).

| Применение | Разрешение, ppi |
|---|---|
| **Сканирование в отраженном свете:** | |
| иллюстрации для Web-страниц | 75-150 |
| сканирование текста без распознавания | 150-200 |
| сканирование текста для распознавания | 300-400 |
| цветное фото для печати на струйном принтере | 200 |
| цветное фото для типографской печати | не менее 300 |
| **Сканирование в проходящем свете:** | |
| 35-мм пленка, для Web-страниц | 200-600 |
| 35-мм пленка, для печати на струйном принтере | 600-2000 |

Другая важная характеристика режима сканирования – *глубина цвета* (разрядность), то есть количество двоичных разрядов, которое используется для кодирования цвета одного пикселя (см. главу 2). Если для кодирования цвета использовано $N$ двоичных разрядов, то общее количество возможных цветов равно $2^N$. Высококачественные устройства позволяют сканировать изображения с глубиной цвета 48 бит.

## 5.6.5. Цифровые датчики

**Датчик** – устройство, регистрирующее какую-либо физическую величину и преобразующее ее в сигналы (обычно электрические).

Компьютер способен не просто хранить большое количество данных, полученных от многочисленных датчиков, но и проводить их математическую обработку. Таким образом, на основе компьютера может быть построена мощная цифровая лаборатория.

Многие датчики вырабатывают аналоговые данные. Поэтому для их подсоединения к компьютеру необходимо устройство, преобразующее аналоговые сигналы в цифровые – *аналого-цифровой преобразователь* (АЦП).

## Контрольные вопросы

1. Зачем нужны устройства ввода?
2. Можно ли сетевую карту, через которую компьютер получает данные, назвать устройством ввода? Почему?
3. Перечислите все известные вам устройства ввода. С какими из них вы работали?
4. Что можно ввести в компьютер с помощью клавиатуры?
5. Что такое управляющие клавиши? Зачем они используются?
6. Что такое функциональные клавиши?
7. Нажатие одной и той же клавиши вызывает разную реакцию компьютера в зависимости от состояния клавиш сдвига – *Shift, Alt* и *Ctrl*. Сколько различных команд можно ввести с помощью одной клавиши?
8. Работая в электронной таблице, пользователь нажал клавиши «3», «2» и «1». Какие операции должен выполнить компьютер, чтобы соответствующее число было записано в память? Указание: вспомните пример 1 из п. 4.3.4. (Ответ: анализ правильности записи числа, преобразование кодов символов в значения цифр, перевод десятичной записи числа в двоичное число)
9. Зачем в клавиатуре установлен микроконтроллер?
10. Почему клавиатура не передает в компьютер готовые коды символов?
11. Как выполняется локализация (использование национальных символов) на клавиатуре? Найдите сведения по этому вопросу в Интернете.
12. Что такое манипулятор?
13. Какие разновидности манипуляторов вы знаете? С какими из них непосредственно работали? Сравните приемы работы с ними. Обоснуйте свой ответ.
14. Как устроена компьютерная мышь?
15. *В старых моделях мышей вращение колесиков фиксировалось с помощью источника света и фотодатчика: при прохождении прорези на диске датчик фиксировал появление света. Докажите, что с одним таким датчиком можно зафиксировать движение и даже измерить его скорость, но нельзя определить направление вращения. Предложите, что нужно сделать, чтобы его узнать? (Поставить рядом *два* датчика).
16. Каким образом движение мыши управляет перемещением курсора на экране?
17. Вспомните все известные вам приемы работы с мышью.
18. Что такое беспроводная мышь?
19. Что такое трекбол и как он работает?
20. Что такое сенсорная панель?
21. Как устроен джойстик?
22. Что такое трекпойнт?
23. Что можно делать с помощью сканера?
24. Можно ли с помощью сканера получить фотографию реального объекта?
25. Как происходит распознавание отсканированного текста?
26. Какие бывают сканеры по конструкции?
27. Назовите наиболее важные характеристики сканеров.
28. Какое устанавливать разрешение при сканировании? На что оно повлияет?
29. Что такое датчики? Каковы возможности компьютера в автоматизации эксперимента?

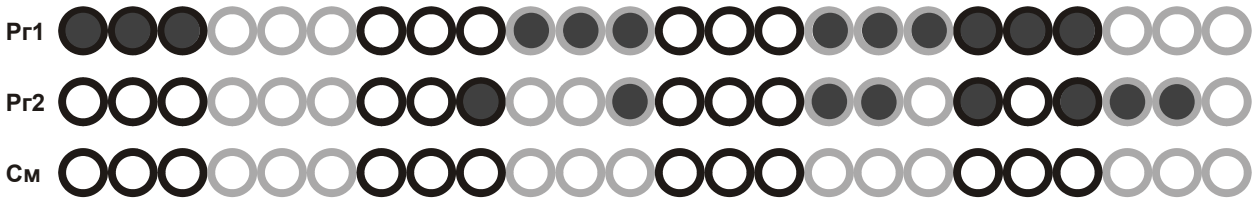
## Задачи

1. Видеокамера оптической мыши имеет 16×16 пикселей, за одну секунду обрабатывается 9000 кадров. Рассчитайте скорость обработки данных в мегапикселях? (Ответ: около 2,3 Мп/с)
2. Вычислите, сколько точек получится при сканировании изображения 10×10 см с разрешением 300 ppi? Оцените объем полученного файла при сканировании в режиме 256 оттенков серого. Проделайте аналогичную оценку для режима 24-битного цвета.

## 5.7. Устройства вывода

**Устройства вывода** – это устройства, которые представляют компьютерные данные в форме, доступной для восприятия человеком.

Первыми устройствами вывода были панели индикаторных лампочек. Каждая из них показывала состояние отдельного бита: горящая лампочка обозначала единицу, а выключенная – ноль. Для чтения результата нужно было хорошо знать двоичную систему.




Этот схематический рисунок изображает индикаторную панель на пульте ЭВМ первых поколений. Разноцветные колпачки патронов с лампочками помогали правильно считывать результат: каждая группа из трёх битов – это одна восьмеричная цифра. Если считать, что горящие лампочки на рисунке обозначены тёмным цветов, то в регистре *Рг1* читается восьмеричное число $70070770_8$, а сумматор *См* очищен (заполнен нулями).

Такие панели использовались для обслуживающего персонала вплоть до третьего поколения ЭВМ, однако для большинства пользователей такой вывод данных был непонятен. Первые «настоящие» устройства вывода печатали числа в десятичном виде на бумагу. Затем печатающие устройства научились печатать не только цифры, но и буквы. Они работали по принципу печатающей машинки: рельефный шаблон символа ударял по красящей ленте, прижатой к бумаге, и оставлял отпечаток.

Кроме устройств, печатающих символы, появились графопостроители (плоттеры), которые рисовали перьями на бумаге графики функций и простейшие картинки. Так как современные принтеры могут выводить текст и графику (в том числе и цветную), специальные устройствах для вывода графики практически не используются.

Революционным событием стало создание мониторов. Это позволило избавиться от ненужного расхода бумаги – теперь можно было выводить на печать только самое необходимое. Кроме того, управление и обслуживание ЭВМ стало более удобным.

Компьютеры четвертого поколения начали обрабатывать мультимедийную информацию – звуковые и видеоданные. Поэтому к компьютерам стали подключать устройства для вывода такой информации: звуковые колонки, наушники, телевизор и т.п. Некоторые из этих устройств (например, наушники) – аналоговые, поэтому для вывода необходимо преобразовать дискретные компьютерные данные в аналоговую форму. Для этого используется специальная электронная схема, которую называют *цифро-аналоговый преобразователь* (ЦАП). ЦАП для вывода звуковой информации входит в состав звуковой карты.

Эволюция устройств вывода не остановилась – все время разрабатываются устройства новых типов, порой весьма экзотические. Например, появились сообщения о создании «3D-принтеров», которые способны под управлением компьютера создавать объемные тела из различных материалов (прежде всего, из пластика).

## 5.7.1. Монитор

Компьютерный монитор состоит из дисплея (панели, на которую смотрит человек) и электронных схем, позволяющих выводить на этот дисплей текстовую и графическую информацию.

Мониторы во многом используют телевизионные технологии. В конце XX века для компьютеров применялись мониторы на основе электронно-лучевых трубок (ЭЛТ), но они были вытеснены жидкокристаллическими (ЖК) мониторами, которые обладают рядом *преимуществ*:

- малый вес и размеры;
- в 2-4 раза меньшее потребление электроэнергии;
- нет искажений изображения, характерных для электронно-лучевых трубок;
- значительно ниже уровень электромагнитного излучения.

Тем не менее, некоторые профессионалы по-прежнему работают на электронно-лучевых мониторах. Дело в том, что ЖК-мониторы имеют *недостатки*, которые трудно устранить в производстве:

- цветопередача хуже, чем у ЭЛТ-мониторов; например, очень трудно получить чисто черный цвет;
- контраст и цвета изображения меняются в зависимости от угла, под которым мы смотрим на монитор;
- при быстром изменении изображения заметно «запаздывание» (жидкие кристаллы не могут поворачиваться слишком быстро);
- при существующих технологиях изготовления у многих мониторов есть дефектные точки, которые не работают (так называемые «битые пиксели»);
- могут отображать чёткую картинку только в одном разрешении, совпадающем с размерами матрицы.

Независимо от конструкции, экран любого монитора строится из отдельных точек, причем каждая из них образована близко расположенными областями трех основных цветов – красного, зелёного и синего. Для ЖК-монитора эти области имеют форму прямоугольников, слегка вытянутых по вертикали. Расстояние между их центрами – доли миллиметра, поэтому глаз человека воспринимает все три составляющие как одну точку «суммарного» цвета.

Элементы экрана часто называют пикселями, что не совсем точно. Строго говоря, пиксель – это точка рисунка, а не экрана. Например, на одном и том же мониторе можно легко устанавливать разные размеры рабочего стола в пикселях. Компьютер «проектирует» пиксели картинки на экран с учетом установленного разрешения. При этом одному пикселю может соответствовать одна или несколько точек экрана.

Элемент ЖК-экрана – это жидкий кристалл, способный под воздействием электрического напряжения менять свои оптические свойства. Каждая точка управляется своим полупроводниковым транзистором. Подчеркнем, что сам жидкий кристалл не способен светиться, он лишь регулирует пропускание света от расположенного за ним источника света.

Наиболее важные характеристики мониторов – это *размер диагонали* (в дюймах) и *максимальное разрешение* (количество точек экрана по ширине и высоте). Для ЖК-мониторов максимальное разрешение – это количество элементов матрицы. Если установить другое (более низкое) разрешение, то качество изображения будет хуже, т.к. видеосистеме придется «растягивать» картинку на реально существующие точки.

Процессор передает данные для вывода *видеокарте* (*видеоконтроллеру*), которая управляет выводом изображения на монитор. Современные видеокарты содержат микропроцессор для обработки графической информации (*графический ускоритель*) и собственную *видеопамять*. Можно считать, что видеокарта – это специализированный компь-

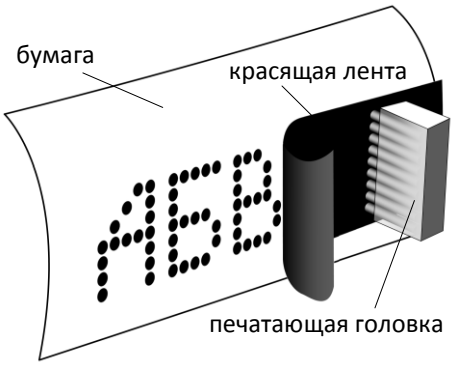
ютер, который существенно ускоряет построение и вывод на монитор графических изображений, особенно трехмерных.

Инженеры активно работают над созданием все более совершенных устройств вывода. Большое внимание уделяется разработке так называемых 3D-дисплеев, которые смогут отображать информацию в трех измерениях.

## 5.7.2. Печатающие устройства

Печатающие устройства (*принтеры*) служат для вывода текстовой и графической информации на бумагу или плёнку. Современные принтеры обрабатывают символы как графику, т.е. рисуют их. На принтерах можно печатать очень сложные изображения, в том числе цветные фотографии.

В настоящее время существует четыре основных типа принтеров: матричные, струйные, лазерные и сублимационные.

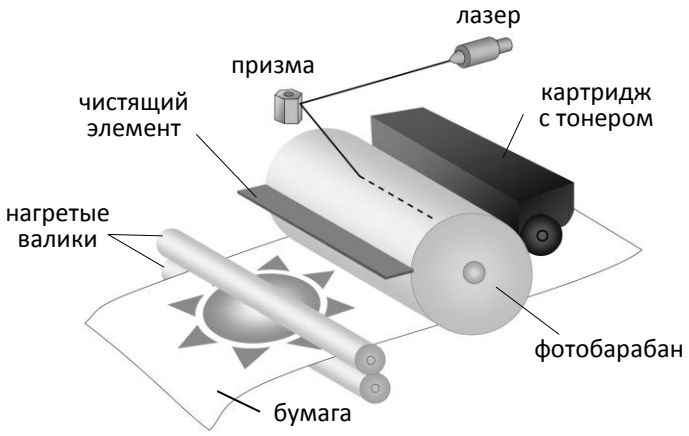



*Матричные* принтеры – это последнее поколение принтеров с ударным принципом работы. Печатающая головка содержит вертикальный ряд иголок, которые под воздействием управляющих сигналов ударяют по красящей ленте, оставляя на бумаге отпечатки в виде маленьких точек.

Головка движется в горизонтальном направлении, что позволяет сформировать строку из символов произвольного вида. На таком принтере можно получать не только тексты, но черно-белые рисунки, однако вывод графики происходит очень медленно. Достоинства матричных принтеров – дешевизна самих принтеров и расходных материалов (красящих лент), а также способность печатать практически на любой бумаге. Однако они не могут обеспечить высокое качество печати, работают медленно и сильно шумят.

Печатающая головка *струйных* принтеров содержит крошечные отверстия, через которые под большим давлением на бумагу выбрасываются чернила. Диаметр получаемых при этом точек гораздо меньше, чем у матричных принтеров, что позволяет получить значительно лучшее качество печати. В цветных принтерах чаще всего устанавливается два картриджа: один с черной краской, а второй – с голубой, фиолетовой и желтой (вспомните цветовую модель CMYK). Изображение строится из точек этих цветов. В некоторых моделях для повышения качества используют шесть базовых цветов. Для печати на струйных принтерах необходима качественная бумага, кроме того, напечатанное изображение расплывается при попадании воды.

*Лазерные* принтеры обеспечивают очень высокое качество печати. Компьютер строит в памяти полный образ страницы и предает его принтеру. Тот с помощью лазерного луча построчно переносит изображение на вращающийся барабан – строит электростатическую копию картинки. Затем к барабану притягиваются мелкие частицы красящего порошка – *тонера*, причем, чем сильнее наэлектризован участок барабана, тем больше краски он получает. На следующем этапе бумага прижимается к барабану, в результате на ней строится отпечаток картинки. Чтобы краска не осыпалась, на выходе нагретый валик вплавляет частицы тонера в бумагу. Поскольку лазерные принтеры используют достаточно сложные технологии, они стоят дороже матричных и струйных, и потребляют больше электроэнергии.




*Светодиодные* принтеры (их тоже часто называют лазерными) работают по такому же принципу, но изображение переносится на барабан не лазером, а светодиодной матрицей.

*Сублимационный* принтер печатает изображение совсем иначе: головка принтера нагревает поверхность, размягчая ее, а затем «впрыскивает» крохотные частицы красителя. Сверху наносится защитный слой, который предохраняет краску от разрушения солнечными лучами, и в итоге образуется очень стойкое изображение. Сублимационные принтеры прекрасно подходят для печати на пластиковых картах и компакт-дисках, часто используются для печати фотографий. Их недостатки – низкая скорость печати (более 1 минуты на одну фотографию) и высокая стоимость.

Важнейшей характеристикой принтера является его разрешающая способность.

**Разрешающая способность принтера** – это максимальное количество точек, которые он способен напечатать на единицу длины.

По традиции разрешающая способность измеряется в точках на дюйм (англ. *dpi* = *dots per inch*). Все современные струйные и лазерные принтеры имеют разрешающую способность не ниже 300 dpi, что обеспечивает высококачественную печать. Некоторые принтеры позволяют пользователю менять разрешающую способность, регулируя тем самым качество печати.

Обратим внимание на разницу обозначений *ppi* (пиксели на дюйм) и *dpi* (точки на дюйм). В *ppi* измеряется разрешение цифрового изображения (например, отсканированного), а в *dpi* – качество печати принтера. Каждый пиксель картинки может изображаться принтером в виде нескольких точек. Вспомните, что примерно то же самое происходит при выводе пикселей изображения на монитор (см. 5.7.1).

Принтеры также часто сравнивают по *скорости* печати (в количестве страниц в минуту). Наименьшая скорость печати у сублимационных и матричных принтеров, а наибольшая – у лазерных. Цветная печать, как правило, выполняется дольше, чем более простая чёрно-белая.

## 5.8. Устройства ввода/вывода

Некоторые компьютерные устройства нельзя однозначно отнести ни к устройствам ввода, ни к устройствам вывода. Пример такого «гибрида» – сенсорный экран. С одной стороны, на него выводится информация, а с другой – пользователь вводит команды, нажимая на нужный участок изображения. Сенсорные экраны применяют в портативных компьютерах, платёжных и информационных терминалах, а также для представления презентаций.

В некоторых мобильных телефонах, карманных и планшетных персональных компьютерах сенсорный экран заменил клавиатуру и занимает всю переднюю панель. Многие из этих устройств (например, смартфон *iPhone* и планшетный компьютер *iPad* фирмы *Apple*) используют технологию *мультитач* (англ. *multitouch*). Это значит, что сенсорный экран отслеживает нажатия и движения пальцев в нескольких точках одновременно. На-

пример, сближая пальцы рук, пользователь уменьшает масштаб изображения на дисплее, а раздвигая — увеличивает.

## Контрольные вопросы

1. Зачем нужны устройства вывода?
2. В чем сходство и различие устройств ввода-вывода и внешней памяти?
3. Перечислите все известные вам устройства вывода. С какими из них вы работали?
4. Что такое МФУ?
5. Что такое АЦП и ЦАП?
6. Что такое монитор и каковы его возможности?
7. Что является элементом изображения в мониторе?
8. Отличается ли пиксель от точки экрана?
9. Компьютер выводит на экран монитора число, хранящееся в ячейке памяти. Какие действия он должен выполнить для этого? (Ответ: перевод двоичного числа в десятичное – получение его десятичных цифр и преобразование их в коды соответствующих символов).
10. Назовите наиболее важные характеристики мониторов?
11. Зачем нужна видеокарта? Каким образом она позволяет разгрузить центральный процессор?
12. Что такое видеопамять?
13. Какие типы принтеров вам известны? Опишите каждый из них.
14. Как работает лазерный принтер?
15. Что такое разрешающая способность принтера?
16. В чем отличие единиц *dpi* и *ppi*?
17. Как пересчитать сантиметры в дюймы?

## Задачи

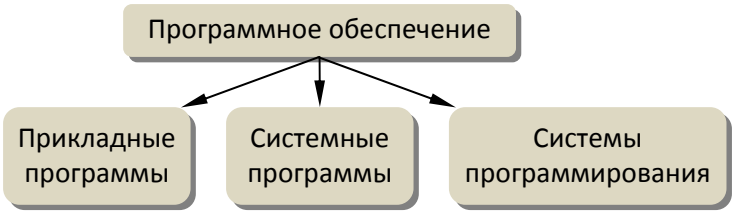
1. Рассмотрите схему индикаторной панели в содержимого регистров, приведенный в начале раздела 5.7. Прочитайте коды чисел, хранящиеся в регистрах *Рг1* и *Рг2*. Считая числа беззнаковыми целыми, сложите их, пользуясь правилами восьмеричной арифметики. Как будет выглядеть сумматор, когда в нем появится код суммы? (Ответ: $70201646_8$)

# Глава 6. Программное обеспечение

## 6.1. Что такое программное обеспечение?

Компьютер – это электронное устройство, которое само по себе ничего не умеет. Чтобы использовать его для решения каких-то задач, необходимо **программное обеспечение** (англ. *software* – «мягкое оборудование») – программы, в которых заложены алгоритмы ввода, обработки и вывода данных.

Обычно выделяют три вида программного обеспечения (ПО): *прикладные программы*, *системные программы* и *системы программирования*.

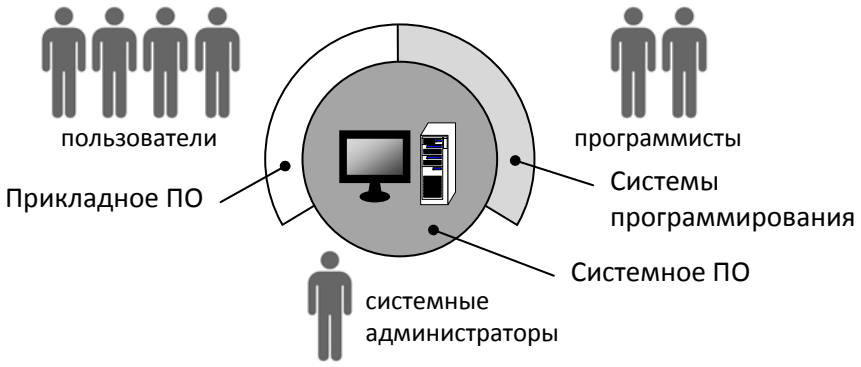



Всех, кто работает с компьютерами, можно разделить на *пользователей*, *системных администраторов* и *программистов*.

Пользователи решают свои задачи с помощью прикладных программ (к ним относятся текстовые и графические редакторы, электронные таблицы, базы данных, игры и т.п.).

*Системные программы* выполняют вспомогательную роль – они обеспечивают пользователю и прикладным программам удобный *интерфейс* (способ обмена данными) с аппаратными средствами. К этой группе относятся операционные системы, *драйверы* (программы для управления внешними устройствами) и *утилиты* (служебные программы). Задача системных администраторов – настроить системное и прикладное ПО, чтобы пользователи смогли нормально работать.

Программисты создают новые программы с помощью *систем программирования* (инструментальных средств).




Программное обеспечение освобождает человека от необходимости работать напрямую с компьютерным «железом» (аппаратными средствами, англ. *hardware* – «жесткое оборудование»).

Часто термин «программное обеспечение» понимают в широком смысле как целую отрасль, включающую все этапы разработки программ, в том числе тестирование (проверку программ, поиск ошибок) и разработку документации.

### *Контрольные вопросы*

1. Назовите три типа программного обеспечения. Чем они отличаются?
2. Какие задачи решают пользователи, программисты, системные администраторы?
3. Что означает слово «интерфейс»?
4. Что такое драйверы, утилиты?
5. Что обозначают английские термины *hardware* и *software*?

## 6.2. Прикладные программы

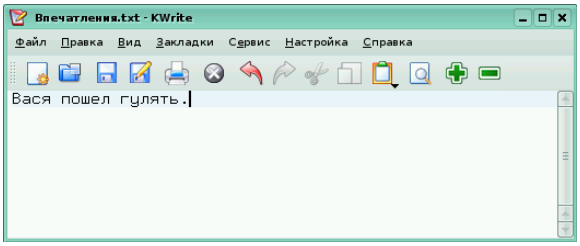
### 6.2.1. Текстовые редакторы

Многие пользователи используют компьютер, прежде всего, для работы с текстами. Обычно различают *редактирование текста* (изменение *содержания*; замена, вставка и удаление символов и слов; разбивка на абзацы) и *форматирование* (изменение *внешнего вида* текста – выбор шрифта, изменение размера, цвета и т.п.).

Простейшие программы этого класса – **текстовые редакторы** – умеют только редактировать текст. Они работают с файлами в формате «только текст» (англ. *plain text*), в которых хранятся коды символов без оформления. Современные редакторы умеют сохранять текст в разных кодировках, но чаще всего используются кодировки семейства UNICODE: UTF-16 (2 байта на символ) или UTF-8 (с переменным числом байт на символ). Примеры текстовых редакторов:

- *Блокнот* в операционной системе *Windows*;
- *nano, gedit, KWrite* и *Kate* в системе *Linux.*

На рисунке справа показано окно текстового редактора **`KWrite`**.




Основные **возможности** современных текстовых редакторов:

- ввод и редактирование текста;
- создание, открытие, редактирование, сохранение и печать документов типа «только текст»;
- работа с буфером обмена (копирование, вырезание, вставка);
- отмена последних операций;
- поиск и замена фрагментов текста;
- подсветка ключевых слов языков программирования (*Си*, *Паскаль* и др.) и разметки (*XML*, *HTML*, *LaTeX*);
- проверка орфографии.

Текстовые редакторы часто используют системные администраторы для редактирования файлов с настройками программ (файлов конфигурации). Тексты программ тоже хранятся в формате «только текст», поэтому программисты набирают их в текстовых редакторах.

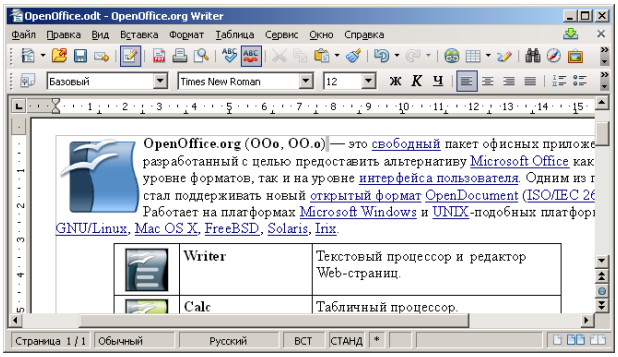
### 6.2.2. Офисные пакеты

Для подготовки офисных документов возможностей простейших редакторов недостаточно – нужно менять шрифт, выделять фрагменты жирным шрифтом, курсивом и подчеркиванием, добавлять таблицы, графики, рисунки и т.п. Кроме того, возникают и другие задачи: выполнение табличных расчетов, подготовка презентаций для выступлений и докладов, работа с базами данных и т.п. Набор программ для подготовки электронных документов называют «офисным пакетом». В офисный пакет обычно включается

- *текстовый процессор*, который позволяет не только редактировать текст, но и оформлять его по стандартам современного делопроизводства;
- *табличный процессор* – программа для выполнения расчетов с табличными данными;
- программа для подготовки *презентаций*;
- программа для работы с *базами данных*.

Самые известные офисные пакеты – *Microsoft Office* (*www.microsoft.com*), *OpenOffice.org* (*openoffice.org*) и *WordPerfect Office* (*www.corel.com*). Пакеты *Microsoft Office* и *WordPerfect Office* – коммерческие, а *OpenOffice.org* можно установить и использовать бесплатно. Кроме того, пакет *OpenOffice.org* – это *кроссплатформенное ПО*, то есть существуют его версии для разных операционных систем (*Windows, Linux*).

**Текстовые процессоры** – это следующий шаг в развитии редакторов текста. На рисунке показано окно текстового процессора *OpenOffice.org Writer*:




В состав пакета *Microsoft Office* входит текстовый процессор *Microsoft Word*, который фактически считается стандартным средством для оформления офисных документов.

С помощью текстовых процессоров можно не только редактировать, но и *форматировать* текст (изменять его оформление). Кроме того, они позволяют

- создавать составные документы, включающие списки, рисунки, таблицы, диаграммы;
- использовать стили оформления (например, заголовки разного уровня);
- использовать шаблоны (заранее оформленные заготовки) документов;
- выполнять несложные вычисления в таблицах;
- сохранять документ в разных форматах, в том числе в *HTML* (как веб-страницу) и *PDF* (универсальный формат документов).

**Табличные процессоры** (электронные таблицы, англ. *spreadsheet*) – это программы для обработки табличных данных. В отличие от текстовых процессоров, они не только хранят данные, но и позволяют выполнять с ними достаточно сложные вычисления, строить диаграммы, проводить анализ, делать прогнозы. Сейчас электронные таблицы – незаменимый рабочий инструмент экономистов, бухгалтеров, менеджеров. В состав пакета *Microsoft Office* включен табличный процессор *Excel*, а в пакете *OpenOffice.org* есть близкая по возможностям программа *OpenOffice.org Calc.*

**Компьютерная презентация** (лат. *praesentatio* – представление) – это набор изображений (*слайдов*), который предназначен для иллюстрации доклада или выступления. Задача презентации – улучшить восприятие информации. В современных презентациях применяют технологии *мультимедиа* (от лат. *multum* – множество, *medium* – средство), то есть в одном документе используют различные формы представления информации: текст, графика, звук, анимация, видео.

Для создания презентаций в пакете *Microsoft Office* применяется программа *Microsoft PowerPoint*, а в пакете *OpenOffice.org* – программа *OpenOffice.org Impress*.

**Система управления базами данных (СУБД)** – это ПО для поиска информации в базах данных, а также для создания и изменения баз данных. В пакет *Microsoft Office* входит СУБД *Access*, а в пакет *OpenOffice.org* – СУБД *OpenOffice.org Base*.

## 6.2.3. Онлайн-офис

Бурное развитие Интернете привело к появлению «онлайн-офисов» – специальных сайтов (Интернет-сервисов), которые предоставляют основные возможности офисных пакетов: текстовый редактор, электронные таблицы, средства для создания презентаций. Для использования такой службы необходим компьютер с доступом в Интернет, причем не имеет значения, какая операционная система на нем установлена. Документы пользователей хранятся на сервере, для доступа к ним нужно ввести пароль. Самый известный «онлайн-офис» – *Google Docs* (*docs.google.com*).

Одно из достоинств «онлайн-офисов» – возможность совместной работы над документами через Интернет. Другим пользователям можно открыть доступ к отдельным документам для просмотра и/или изменения. Любой документ может быть *экспортирован* (сохранен) в файл на диске компьютера.

«Онлайн-офисы» используют технологию, известную под названием «облачные вычисления» (англ. *cloud computing*). Ее суть в том, что пользователь хранит свои данные на серверах Интернета и не должен заботиться о способе их хранения, операционной системе и программном обеспечении. Слово «облако» – это метафора, образ достаточно сложной системы, детали работы которой знать не обязательно. Несмотря на удобства «облачных» сервисов, существуют опасения, что пользователь может потерять контроль над своими данными, и это чревато серьезными проблемами. Например, иногда не удается полностью удалить данные, которые человек сам же разместил. Кроме того, возможны утечки информации и потеря данных.

## 6.2.4. Графические редакторы

Графические редакторы – это программы для создания и редактирование изображений. Изображения, хранящиеся в компьютере, делятся на растровые и векторные (см. разд. 2.13). С ними нужно работать по-разному, поэтому различают растровые и векторные графические редакторы[1].

**Растровые редакторы** предназначены для

- обработки фотографий;
- подготовки изображений к печати;
- создания и редактирования изображений для веб-сайтов.

Лучшим профессиональным растровым редактором считается программа *Adobe Photoshop* (*www.adobe.com*). Существуют ее версии для операционных систем *Windows* и *Mac OS X* (для компьютеров фирмы *Apple*). В состав *Windows* входит растровый редактор *Paint*, но для сложной обработки (например, для коррекции фотографий) его возможности недостаточны.

Бесплатная программа *Gimp* (*gimp.org*) – кроссплатформенная, она работает как в *Windows,* так и в *Linux*. На рисунке слева показано окно программы *Gimp*.

[1] Более точно называть их редакторами растровой и векторной графики.

Среди бесплатных растровых редакторов широкими возможностями обладает *Paint.NET* (*www.getpaint.net*), но он пока устойчиво работает только в среде *Windows*.

В последнее время были созданы бесплатные «онлайновые» редакторы (например, *www.pixlr.com*), которые позволяют обрабатывать изображения на специальной веб-странице в Интернете, без установки дополнительного ПО на компьютер пользователя.

**Векторные редакторы** используются для подготовки

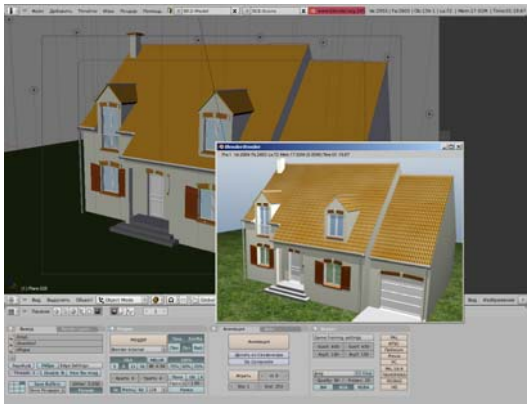
- художественных иллюстраций;
- технических иллюстраций (схем, графиков);
- логотипов, визиток, плакатов;
- изображений для веб-сайтов (иконок, кнопок).

Среди профессиональных векторных редакторов можно назвать двух лидеров – программы *Adobe Illustrator* (*www.adobe.com*) и *CorelDraw* (*www.corel.com*). В свободно распространяемый пакет *OpenOffice.org* входит векторный редактор *OpenOffice.org Draw.* Бесплатный редактор *Inkscape* (*www.inkscape.org*) – еще одна кроссплатформенная программа, работающая как в *Windows*, так и в *Linux*. На рисунке справа показано окно редактора *Inkscape.*

Во многих графических редакторах, например, в *Adobe Photoshop* и *Corel Draw*, можно создавать документы, содержащие как растровую, так и векторную графику. Текстовые процессоры (*Microsoft Word*, *OpenOffice.org Writer*) позволяют вставлять в документ растровые и векторные рисунки.

Во многих областях двухмерных рисунков недостаточно, и необходимо представить объект в трехмерном пространстве. Такие задачи возникают, прежде всего, в архитектуре, кино, телевидении и компьютерных играх. С помощью трехмерной графики (англ. *3D = 3 dimensions*, «3 измерения») создаются многие современные мультфильмы.

Для работы с трехмерными объектами используют программы специального класса (программы 3D-моделирования), которые позволяют:



- определить форму (геометрию) объектов сцены;
- задать материалы для объектов;
- установить источники света;
- определить точки наблюдения (виртуальные камеры);
- создать анимацию с трехмерными объектами;
- выполнить *рендеринг*, то есть построить плоскую картинку или последовательность кадров анимации с учетом свойств объектов и источников света.

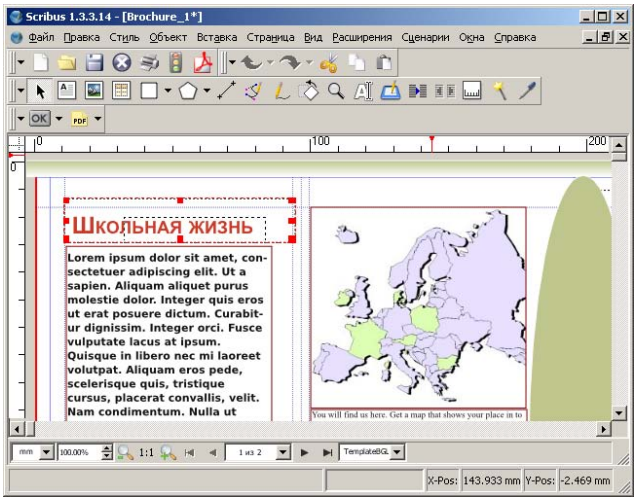
Среди программ 3D-моделирования, работающих в системе *Windows,* наиболее популярна *3D Studio MAX* (*usa.autodesk.com*). В области кино и телевидения стандартом считается кроссплатформенная программа *Maya* (*www.autodesk.com/maya*), версии которой существуют для ОС *Linux, Windows* и *Mac OS*. Среди свободно распространяемых программ наиболее известны кроссплатформенная программа *Blender* (*www.blender.org*), а также *K-3D* (*www.k-3d.org*) и *Wings-3D* (*www.wings3d.com*).

Алгоритмы работы с трехмерной графикой достаточно сложны и требуют большого объема вычислений. Поэтому для нормальной работы программ 3D-моделирования, особенно для выполнения рендеринга, требуется быстродействующий процессор и много оперативной памяти.

## 6.2.5. Настольно-издательские системы

В любой современной типографии для подготовки к печати книг и журналов используется специальное программное обеспечение – **настольно-издательские системы** (англ. *DTP = Deskop publishing*, «настольное издательство»).

Основное отличие этих программ от текстовых процессоров состоит в том, что в них можно выполнять *вёрстку* – точно задавать расположение текста, рисунков, таблиц и другого материала на странице в соответствии с правилами типографского дела. В настольно-издательских системах готовят *оригинал-макет* (изображение, точно совпадающее с будущим отпечатком) и отправляют его в типографию.




Долгое время промышленным стандартом была настольно-издательская система *QuarkXPress* (*www.quark.com*), сейчас конкуренцию ей составляют *Adobe InDesign* (*www.adobe.com*) и бесплатная программа *Scribus* (*www.scribus.net*), окно которой показано на рисунке.

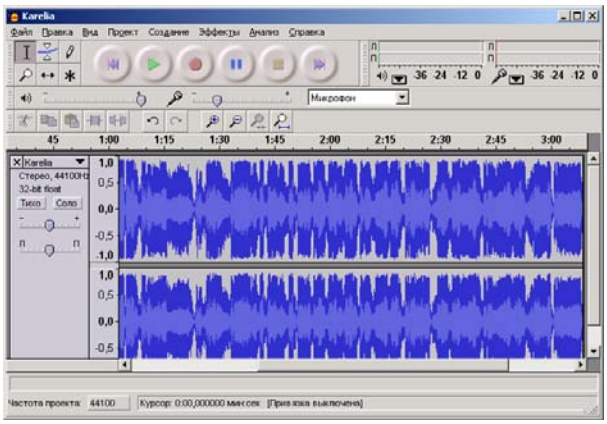
В состав пакета *Microsoft Office* входит программа верстки *Microsoft Publisher* с несколько меньшими возможностями. Верстку несложных изданий (например, визиток, буклетов) можно выполнить в программе *CorelDraw*, но профессионалы не рекомендуют ее использовать.

Лучшая система подготовки математических текстов называется *TeX* (*ctan.org*). Ее разработал известный американский математик и специалист по теоретической информатике Д. Кнут. Система *TeX* – бесплатная и кроссплатформенная, она принята в качестве стандарта во многих российских и зарубежных издательствах, выпускающих математическую литературу.

## 6.2.6. Редакторы звука и видео

**Аудиоредакторы** – это программы для редактирования звуковых файлов. С их помощью можно

- загружать, редактировать и сохранять звуковые файлы разных форматов;
- записывать звук с микрофона или другого источника;
- вырезать фрагменты из файла;
- соединять звуковые фрагменты в один файл;
- изменять громкость и темп звука;
- удалять шумы.




Самые известные аудиоредакторы – это *Adobe Audition* (*www.adobe.com*), *Sound Forge* (*www.sonycreativesoftware.com*), *Audacity* (*audacity.sourceforge.net*). В отличие от первых

двух программ, *Audacity* – бесплатный продукт, существующий для многих операционных систем (*Windows, Linux, Mac OS X*). На рисунке показано окно программы *Audacity*.

**Видеоредакторы** предназначены для создания и редактирования цифрового видео. Их основные возможности:

- ввод данных с видеокамеры;
- коррекция цвета;
- добавление, перестановка, удаление фрагментов фильма;
- добавление звука и титров;
- сохранение фильма в различных цифровых видеоформатах;
- создание DVD-дисков.


Среди коммерческих программ этого типа наиболее популярны *Adobe Premier* (*www.adobe.com*), *Pinnacle Studio* (*www.pinnaclesys.com*), *VideoStudio Pro* (*www.corel.com*).

На компьютерах фирмы *Apple* используется видеоредактор *iMovie* (*www.apple.com*).

Существуют и бесплатные видеоредакторы, например, программа *Kino* для операционной системы *Linux* (*kinodv.org*), окно которой показано на рисунке.

## 6.2.7. ПО для работы в Интернете

Для просмотре веб-страниц в Интернете нужна специальная программа, которую называют **браузер** (англ. *browser*). Пользователи Интернета чаще всего используют браузеры

*Internet Explorer* (входит в состав операционной системы *Windows*);
*Firefox* (*www.mozilla-russia.org*);
*Chrome* (*www.google.com/chrome*);
*Safari* (*www.apple.com/safari*);
*Opera* (*www.opera.com*).

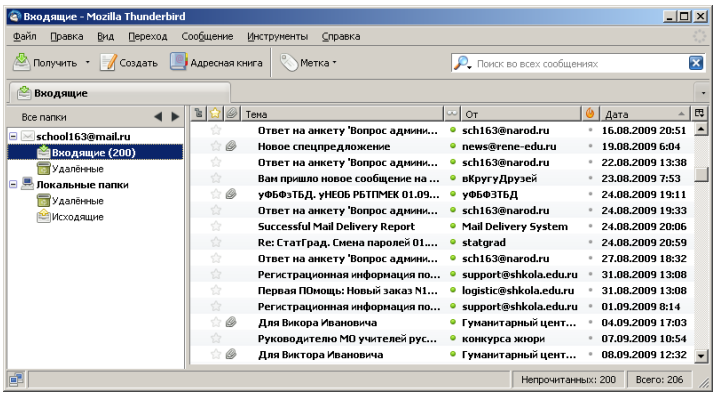
Большинство браузеров бесплатные, многие из них кроссплатформенные. Например, версии браузеров *Firefox* и *Opera* существуют для большинства современных операционных систем. В состав *Linux* входит браузер *Konqueror*, но пользователи чаще всего предпочитают *Firefox*.

Кроме браузера, большинство пользователей используют **почтовые программы** (*почтовые клиенты*), предназначенные для работы с электронной почтой. Их основные возможности:

- создание, отправка и прием сообщений;
- автоматическая проверка почты через заданный интервал;
- сортировка сообщений по папкам;
- ведение адресной книги (списка контактов).

В состав современных версий операционной системы *Windows* входит почтовая программа *Почта Windows.* Несколько большими возможностями обладают профессиональные программы *Microsoft Outlook* (входящая в пакет *Microsoft Office*) и *TheBat* (*www.ritlabs.com*). На компьютерах фирмы *Apple* устанавливается почтовый клиент *Apple Mail* (*www.apple.com*). В состав браузера *Opera* входит встроенный почтовый клиент *Opera Mail*, возможностей которого вполне достаточно для простых задач.

Самая известная из бесплатных программ – *Mozilla Thunderbird* (*www.mozilla-russia.org*) – является кроссплатформенной: существуют ее версии для операционных систем *Windows*, *Linux*, *Mac OS X*. На рисунке показано окно программы *Mozilla Thunderbird*:




Для общения *в реальном времени* (это значит, что собеседники одновременно находятся за компьютерами) используют программы для обмена мгновенными сообщениями (**мессенджеры**). Самые известные мессенджеры – *ICQ* (*www.icq.com*), *Mail.ru Агент* (*www.mail.ru*), *Kopete* (для *Linux*), *iChat* (для компьютеров фирмы *Apple*). В последнее время все большую популярность приобретает система мгновенного обмена сообщениями *Jabber* (*www.jabber.org*), также известная под названием *XMPP*. Она бесплатная, обеспечивает надежную защиту передаваемых данных, позволяет устанавливать связь с пользователями других систем, например, ICQ.

С помощью программы *Skype* (*skype.com*) можно установить через Интернет голосовую и видеосвязь между компьютерами. Для этого на каждом из компьютеров должен быть микрофон и веб-камера.

## ? *Контрольные вопросы*

1. Какое ПО называют прикладным?
2. Чем отличаются текстовые редакторы и текстовые процессоры?
3. Что означает формат «только текст»? В каких случаях он используется?
4. Какие программы обычно входят в офисный пакет?
5. Что такое кроссплатформенное ПО?
6. Что такое СУБД?
7. Что такое «онлайн-офис»? В чем его достоинства и недостатки?
8. Что такое «облачные вычисления»?
9. Чем отличаются растровые и векторные графические редакторы?
10. Перечислите возможности растровых редакторов.
11. Для каких целей используются векторные редакторы?
12. Что такое настольно-издательская система? Чем она отличается от текстового процессора?
13. Что такое оригинал-макет?
14. Какая система лучше всего подходит для набора математических текстов?
15. Какими возможностями обладают аудиоредакторы?
16. Перечислите возможности редакторов видео.
17. Что такое браузер?
18. Перечислите возможности почтовых программ.
19. Что такое мессенджер?

# 6.3. Системное программное обеспечение

## 6.3.1. Что такое операционная система?

Команды, которые умеет выполнять процессор представляют собой числовые коды. Чтобы он выполнил программу, нужно эту программу загрузить в память и передать процессору адрес первой команды. В принципе, это можно делать вручную, с помощью переключателей (1/0) или перфокарт, так и было на первых компьютерах. Однако в этом случае ввод программы будет значительно занимать больше времени, чем ее выполнение, поэтому процессор будет простаивать. Кроме того, для ввода и вывода данных нужно программировать внешние устройства, каждое из которых имеет собственный набор команд. В таких условиях с компьютером могут работать только высококвалифицированные специалисты. Ситуация еще более усложняется, если требуется записать данные на жесткий диск или обеспечить одновременную работу нескольких программ.

Для решения всех этих проблем программисты разработали вспомогательные программы (точнее, программные системы, состоящие из многих программ), которые называются *операционными системами*.

**Операционная система (ОС)** – это комплекс программ, обеспечивающих пользователю и прикладным программам удобный **интерфейс** (способ обмена информацией) с аппаратными средствами компьютера.

Операционная система **обеспечивает**:

- взаимодействие пользователя и аппаратных средств;
- обмен данными между прикладными программами и устройствами компьютера;
- работу файловой системы (хранение данных в виде файлов и папок);
- запуск и выполнение прикладных программ;
- обработку ошибок, контроль за работой оборудования;
- распределение ресурсов компьютера между несколькими одновременно работающими программами (время работы процессора, память, внешние устройства).

Операционные системы бывают *однозадачные* (на компьютере в любой момент выполняется только одна программа) и *многозадачными* (пользователь может запустить несколько программ, которые будут выполняться одновременно).

Первые операционные системы появились на компьютерах второго поколения и были *однозадачными*. Нередко получалось так, что большую часть времени занимали не вычисления, а операции ввода и вывода данных, тогда как процессор в это время простаивал. Чтобы полностью использовать мощность компьютера, разработали *пакетный режим*: в разные области памяти загружали несколько программ. Когда одна программа выполняла операции ввода-вывода, процессор переходил к выполнению следующей, и таким образом мог быть загружен практически на полную мощность.

На компьютерах третьего поколения часто использовался *многопользовательский режим* (режим разделения времени), при котором с большим компьютером (*мэйнфреймом*) было связано несколько *терминалов* (так называли рабочие места с клавиатурой и монитором). С каждого терминала можно было отправить задание на выполнение, таким образом, с компьютером одновременно работало несколько программистов.

Операционные системы первых персональных компьютеров были однозадачными. Самая популярная ОС в 1980-х годах – *MS DOS* (*Microsoft Disk Operating System* – дисковая операционная система фирмы *Microsoft*). Сейчас иногда на дешевые ноутбуки устанавливается ее бесплатный аналог – *FreeDOS* (*www.freedos.org*).

Все современные ОС – *многозадачные*. ОС распределяет время работы процессора между запущенными программами, выделяя каждой *кванты* (небольшие интервалы) времени, так что создается впечатление, что программы работают одновременно, даже если на компьютере установлен один процессор.

В **состав операционной системы** обычно входят:

- *начальный загрузчик* – небольшая программа, расположенная в первом секторе загрузочного диска; его задача – организовать загрузку в память *ядра* (основной части) ОС и передать ему управление;
- *система управления памятью*;
- *система ввода и вывода*, которая управляет внешними устройствами и файлами; она использует программы для обмена данными с дисководами, клавиатурой, монитором и принтером, записанные в постоянном запоминающем устройстве (ПЗУ) микросхемы BIOS[2], расположенной на материнской плате;

Микросхема BIOS

- *командный процессор* – программа, которая выполняет команды пользователя, введенные в командной строке, и *командные файлы* – текстовые файлы, содержащие списки команд и даже программы на специальном языке программирования;
- *утилиты* (лат. *utilitas* – польза) – служебные програмы для проверки и настройки компьютера.

Может ли компьютер работать без операционной системы? Да, в том случае, если он работает по одной единственной программе, которая хранится в ПЗУ или на диске, и автоматически запускается при включении питания. Например, микрокомпьютеры, встроенные в бытовые устройства, могут обходиться без операционной системы. Однако такой компьютер невозможно настраивать, поэтому во многих более сложных устройствах (игровых приставках, банковских терминалах и т.д.) используют операционные системы.

## 6.3.2. Современные операционные системы

Самые популярные современные операционные системы для персональных компьютеров – *Windows*, *Mac OS X*[3] и *Linux.* Все они используют графический интерфейс с пользователем: окна программ, управление с помощью мыши, кнопки, переключатели и т.п.

Система *Windows* разработана фирмой *Microsoft* (*www.microsoft.com*) и распространяется на коммерческой основе. По управлением *Windows* работает более 90% персональных компьютеров[4], имеющих доступ в Интернет.

Примерно 5% пользователей используют *Mac OS X*. Она устанавливается на компьютерах фирмы *Apple*, которые часто используют профессионалы в области дизайна, компьютерной графики, полиграфии, видеомонтажа.

Около 1% компьютеров работают под управлением ОС *Linux*. Ее начал разрабатывать в 1991 году финский студент *Линус Торвальдс* в качестве хобби. Сейчас в развитии *Linux* принимают участие сотни разработчиков во всём мире. В современном ядре *Linux* насчитывается более 11 млн. строк кода[5]. Система *Linux* распространяется бесплатно вместе с исходными кодами, так что каждый (при желании и умении) может ее усовершенствовать.

[2] *BIOS = Basic Input-Output System* – базовая система ввода-вывода.

[3] Последний символ в названии *Mac OS X* – это римская цифра X, что означает «версия 10».

[4] Данные сайта *www.netmarketshare.com* (2010 г.)

[5] Данные 2009 г.

На основе ядра *Linux* построено много различных *дистрибутивов* (распространяемых сборок), самые известные из них – *Ubuntu* (*ubuntu.com*), *Mandriva* (*mandriva.ru*), *OpenSUSE* (*opensuse.org*), *Slackware* (*www.slackware.com*), *Gentoo* (*www.gentoo.org*). В дистрибутивы входит не только сама операционная система, но и программное обеспечение, состав которого зависит от конкретной сборки. Существуют дистрибутивы с улучшенной поддержкой русского языка, например, *ALT Linux* (*www.altlinux.org*).

Достоинства *Linux*:

- бесплатное распространение ОС и многих программ для нее;
- высокий уровень безопасности и защиты от вирусов;
- невысокие требования к аппаратным средствам;
- возможность гибкой настройки.

Основные *сферы применения Linux*:

- личные компьютеры (не нужно платить за ПО);
- портативные компьютеры, которые закупаются организациями в большом количестве;
- серверы в локальных сетях и в Интернете (до 50%), важно быстродействие;
- суперкомпьютеры (до 80%[6]), важна возможность настройки для работы на нестандартном оборудовании;
- встроенные компьютеры в банкоматах, терминалах оплаты, стиральных машинах и даже беспилотных военных аппаратах; важна бесплатность и возможности настройки.

Среди *недостатков* этой ОС обычно отмечают:

- сложность настройки для неквалифицированного пользователя (для выполнения многих операций необходимо вводить команды в режиме командной строки);
- отсутствие драйверов для некоторых устройств и сложность их установки;
- отсутствие версий популярных профессиональных программ, например, *Adobe Photoshop*;
- отсутствие поддержки современных игр.

Появление карманных персональных компьютеров (КПК), смартфонов и коммуникаторов привело к развитию нового класса операционных систем, которые могут работать на маломощном оборудовании. Самая популярная **ОС для мобильных устройств** – *Symbian*, с ней конкурируют *Windows Mobile* (разработка фирмы *Microsoft*) и *BlackBerry*. Некоторые ОС этого класса строятся на основе ядра *Linux*: *Google Android, Palm webOS, Maemo* и другие. Портативные компьютеры фирмы *Apple* (*iPhone, iPod Touch, iPad*) используют операционную систему *iPhone OS.*

Новая операционная система компании *Google* для персональных компьютеров, названная ***Chrome OS***, тоже строится на ядре *Linux*. Она нетребовательна к аппаратным ресурсам компьютера, основная роль отводится веб-браузеру и «облачным вычислениям». Данные пользователя хранятся на серверах Интернета, для их обработки используются веб-службы, при этом на компьютер не нужно устанавливать дополнительное программное обеспечение. Недостаток этой ОС – низкая безопасность. Также она не подойдет тем, кому нужно выполнять сложную обработку графики и видео.

Существует еще один класс операционных систем, от которых требуется не просто решать задачи, а делать это за определённый промежуток времени. Такие ОС называются операционными системами **реального времени**. Они применяются в тех случаях, когда задержка может привести к аварии, катастрофе или финансовым потерям: в системах аварийной защиты, системах управления роботами и самолетами, в военных приборах. Например, робот, снимающий деталь с конвейера, должен сделать это за маленький промежуток времени. Наиболее известные системы

[6] Данные сайта *www.top500.org* (2010 г.)

реального времени – *QNX* (*www.qnx.com*), *Windows CE* (*www.microsoft.com*), *VxWorks* (*www.windriver.com*) и *LynxOS* (*www.lynuxworks.com/rtos*).

Многие современные операционные системы, включая *Linux*, *Mac OS X*, *QNX, VxWorks, LynxOS*, относятся к классу *UNIX*-подобных ОС. Это значит, что они используют общие идеи и принципы, заложенные в 1970-х годах при разработке системы *UNIX*:

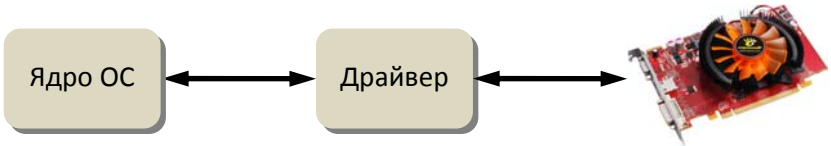
- для настройки и управления системой используются простые текстовые файлы;
- программы часто используют текстовый ввод данных и вывод результатов;
- широко применяются утилиты, запускаемые в командной строке;
- каждая утилита выполняет одну задачу; ее режимы работы можно задачать с помощью параметров командной строки;
- утилиты можно объединять в «конвейер», направляя результаты работы одной утилиты на вход следующей;
- все устройства (жесткие диски, флэш-диски, принтеры, сканеры) рассматриваются как файлы.

Сейчас система *UNIX* используется в основном для управления серверами. Все *UNIX*-подобные системы считаются очень надежными с точки зрения безопасности. Достаточно сказать, что для них практически неактуальна проблема компьютерных вирусов.

## 6.3.3. Драйверы устройств

**Драйверы** (англ. *driver* – водитель) – это программы специального типа, которые находятся в оперативной памяти и обеспечивают обмен данными между ядром ОС и внешними устройствами компьютера (звуковой картой, видеокартой, сетевой картой, принтером и т.п.). Драйверы обычно включают в подсистему ввода и вывода.

Драйвер представляет собой набор процедур, которые вызываются ядром ОС при необходимости передать данные устройству или принять от него данные. Задача драйвера – преобразовать команды ввода-вывода в команды конкретного устройства. Драйверы загружаются в память и фактически становятся частью ОС. Такая схема позволяет устанавливать и использовать устройства, которые были разработаны уже после выпуска операционной системы.




Если драйвер не установлен, устройство работать не будет, потому что неизвестно, как к нему обращаться. Драйверы наиболее популярных устройств обычно включаются в дистрибутив (установочный пакет) операционной системы. Когда ОС обнаруживает новое устройство, она пытается найти подходящий драйвер в своей базе данных. Если такого драйвера нет, его можно установить вручную с диска, который прилагается к устройству. Кроме того, любой драйвер можно бесплатно скачать из Интернета с сайта производителя.

## 6.3.4. Утилиты

Утилиты решают вспомогательные задачи, расширяя возможности ОС. К утилитам относятся

- программы для *проверки дисков* (**chkdsk** в *Windows,* **fsck** в *Linux*);
- программы для *разбивки жестких дисков*, с помощью которых можно сделать несколько разделов на одном диске (*Управление дисками* в *Windows*; *GNU Parted* в *Linux*);
- *файловые менеджеры* – программы для работы с файлами; самые известные файловые менеджеры для *Windows* – *Проводник* (входит в состав ОС), *Total Commander* (*www.ghisler.com*), *Free Commander* (*www.freecommander.com*), *Far Manager*

(*farmanager.com*); в *Mac OS X* используется программа *Finder*, а в операционной системе *Linux* – файловые менеджеры *Konqueror*, *Midnight Commander* и др.;

- *антивирусные программы*: *AVP* (*www.kaspersky.ru*), *DrWeb* (*www.drweb.com*), *Nod32* (*www.eset.com*), *McAfee* (*home.mcafee.com*) и др.;
- *архиваторы* и *программы для сжатия данных*; в ОС *Windows* чаще всего используются *WinRAR* (*www.rarlab.com*) и *WinZip* (*www.winzip.com*); в *Linux* – *Ark* (*utils.kde.org*) и *File Roller* (*fileroller.sf.net*); архиватор 7ZIP (*www.7-zip.org*) распространяется бесплатно с исходными кодами для различных операционных систем;
- программы для *шифрования данных*, например, *PGP* и ее версии для разных операционных систем (*www.pgpru.com*);
- редакторы, позволяющие менять данные на диске и в оперативной памяти; например, программы *HxD* (*mh-nexus.de/en/hxd*) и *WinHex* (*www.winhex.com*) для ОС *Windows* или *hexedit* (*rigaux.org/hexedit.html*) для *Linux*;
- сетевые утилиты для проверки связи в локальной и глобальной сетях; например, утилиты **`ping`**, **`tracert`** (**`traceroute`**), **`nslookup`** в *Windows* и *Linux*.

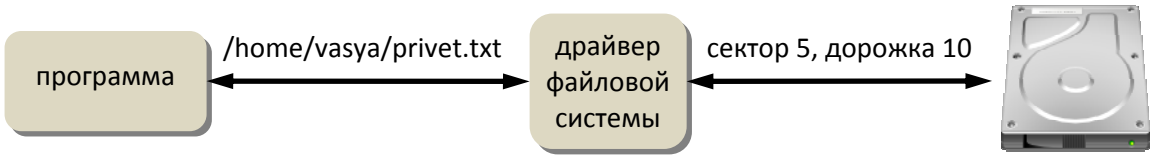
Часто к утилитам относят также

- программы для *записи CD и DVD*-дисков; в системе *Windows* наиболее популярны программы *Nero Burning ROM* (*www.nero.com*), *CDBurnerXP* (*cdburnersp.se*) и *DeepBurner* (*www.deepburner.com*); в *Linux* для этой цели используют утилиту *K3b* (k3b.org)$;
- программы для *сканирования и распознавания текста*; широко применяются коммерческая программа *ABBYY FineReader* (*www.abbyy.ru*) и бесплатная *CuneiForm* (*www.cuneiform.ru*).

## 6.3.5. Файловые системы

Вы знаете, что данные можно хранить в виде файлов на жестких дисках, CD и DVD-дисках, флэш-дисках. Однако винчестер и флэш-диск «ничего не знают» о том, что записанные на них данные в самом деле объединены в файлы и папки.

С другой стороны, когда программа сохраняет файл на диск, она «ничего не знает» о том, в какое место диска эта информация будет записана, указывается только имя файла и каталог. Поэтому между программой и носителем информации необходим «посредник», который определяет, в какое именно место диска будут записаны биты переданных данных. Эту роль выполняет *драйвер файловой системы*:




**Файловая система** – это порядок размещения, хранения и именования данных на носителе информации.

Файловые системы решают несколько **задач**:

- определяют правила построения имен файлов и каталогов;
- определяют, как именно размещаются файлы на диске;
- предоставляют программам функции для работы с файлами;
- обеспечивают защиту данных в случае сбоев и ошибок;
- обеспечивают установку прав доступа к данным для каждого пользователя;

- обеспечивают совместную работу с файлами (когда один пользователь открыл файл, для остальных устанавливается режим «только чтение»).

С точки зрения файловой системы диск делится на *кластеры* (блоки) размером от 512 байт до 64 Кбайт. Каждому файлу выделяется целое число кластеров. Файлу размером 1 байт выделяется целый кластер, остальное место считается занятым, но фактически не используется. Поэтому при большом размере кластера хранить мелкие файлы невыгодно, значительная часть места пропадает впустую. С другой стороны, при увеличении размера кластера скорость чтения и записи больших файлов повышается, кроме того, увеличивается и максимальный объем диска, который поддерживает файловая система.

В операционной системе *Linux* применяются файловые системы **ext3** и **ext4**. Они поддерживают журналирование, помогающее сохранить данные в случае сбоев. Его суть в том, что *перед* выполнением операции с файлами ОС записывает «план действий» в специальный журнал. Когда операция полностью закончена, эта запись из журнала удаляется. Если во время операции произошел сбой (например, отключение питания), по записям в журнале можно сразу определить, какие файлы могли быть затронуты. Таким образом, журналируемая файловая система устойчива к сбоям.

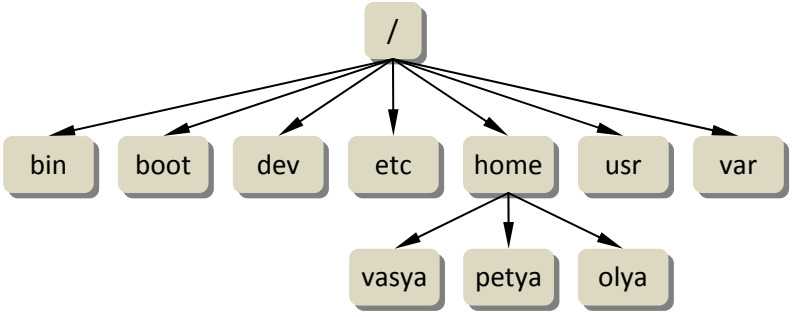
В системе *Windows* применяют файловые системы **NTFS** и **FAT32**. Хотя **FAT32** в некоторых случаях работает быстрее и требует меньше памяти, она считается устаревшей. В отличие от **FAT32**, файловая система **NTFS**

- обеспечивает защиту от сбоев с помощью журналирования (в **FAT32** журналирование не используется);
- позволяет назначить права доступа к файлам и папкам (в **FAT32** каждый пользователь может просматривать и изменять данные всех остальных);
- позволяет задать *квоту* (ограничение) на использование диска для каждого пользователя;
- позволяет использовать сжатие файлов и папок без специальных программ.

В операционной системе *Mac OS X* применяется файловая система **HFS** (англ. *Hierarchical File System*, иерархическая файловая система).

Первые файловые системы были *одноуровневыми*, то есть, все файлы хранились в одном каталоге (на дискете). С увеличением емкости дисков (и количества файлов!) это стало неудобно, поэтому разработали *иерархические* (многоуровневые) файловые системы, где файлы группируются в каталоги, а сами каталоги вложены друг в друга. Такая структура называется *деревом каталогов*.

В операционной системе *Linux* существует один корневой каталог (обозначаемый как «/»), остальные файлы и каталоги вложены в него. Любое устройство (включая жесткие диски, принтеры, сканеры и т.п.) в *Linux* рассматривается как файл, то есть входит в состав единой иерархической файловой системы.



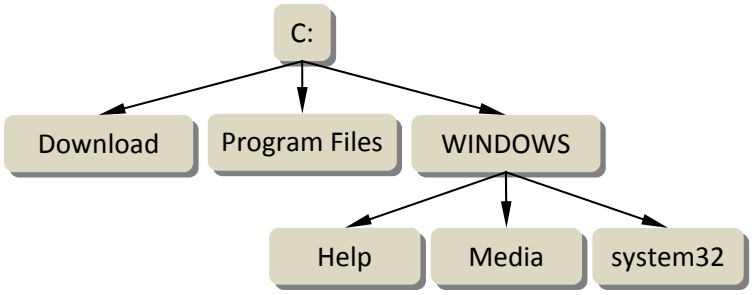

Вот что хранится в показанных каталогах:

- **bin** – команды операционной системы;

- **boot** – ядро ОС и данные для загрузки;
- **dev** – файлы устройств, подключенных к ОС;
- **etc** – файлы с настройками ОС и некоторых программ;
- **home** –домашние каталоги пользователей;
- **usr** – установленные пакеты программ;
- **var** – часто меняющиеся данные, например, журналы ОС.

Чтобы указать путь к файлу или каталогу, перечисляют (начиная от корня) все каталоги, в которых он находится, разделяя их символом «/» (он называется «слэш»). Например, адрес домашнего каталога пользователя **petya** запишется как **/home/petya**, а адрес файла **qq.txt** в этом каталоге – как **/home/petya/qq.txt**.

Дерево каталогов в операционной системе *Windows* строится отдельно для каждого диска:




В качестве разделителя при записи адреса файла или каталога (папки) используют обратный слэш «\», например, **C:\WINDOWS\System32\shell32.dll**.

Для работы с группами файлов используются *маски* или *шаблоны* (англ. *wildcards*). Кроме символов, которые допустимы в именах файлов, маска может включать два специальных символа: знак «*****» заменяет любое количество любых символов, а знак «**?**» – один любой символ. Приведем несколько примеров:

| | |
|---|---|
| ***.*** | все файлы; |
| ***.bmp** | все файлы с расширением **.bmp;** |
| **a*.?** | файлы, имя которых начинается с буквы «a», а расширение состоит из 1 символа |
| ***x*.??*** | файлы, в имени которых есть буква «x», а расширение содержит не менее 2-х символов; |
| ***z.a?** | файлы, имя которых заканчивается на букву «z», а расширение начинается с буквы «a» и состоит из 2-х символов. |

Маски чаще всего применяют для поиска файла по известной части имени или по расширению. Например, для того, чтобы найти все документы *Word*, у которых имя содержит слово «**отчет**», можно использовать маску ***отчет*.doc***. При этом будут найдены, например, такие файлы:

**отчет2010.doc**
**Самый_важный_отчет.docx**
**Новый_отчет_март_2011.docx**

В операционной системе *Windows* заглавные и строчные буквы в названиях файлов и каталогов не различаются, то есть к файлу с именем **Вася.txt** можно обращаться как **вася.txt**, **вАсЯ.txt**, **ВаСя.txt** или **ВАСЯ.txt**. В *Unix*-подобных ОС (*Linux*, *Mac OS X*) это не так, все перечисленные имена файлов – разные, и такие файлы могут быть созданы в одном каталоге.

Нужно отметить, что файловая система не обязательно напрямую связана с жестким диском или другим физическим носителем информации. Существуют, например, *сетевые файловые системы*, которые являются просто способом обращения к файлам на удалённых компьютерах.

## *Контрольные вопросы*

1. Зачем нужны операционные системы? Можно ли обойтись без них?
2. Что такое операционная система?
3. Какие задачи выполняет ОС?
4. Чем отличаются многозадачные ОС от однозадачных?
5. Как обеспечивается многозадачность на компьютерах с одним процессором?
6. Что такое пакетный режим работы?
7. Что такое многопользовательский режим?
8. Перечислите составные части ОС.
9. Что такое начальный загрузчик?
10. Что такое драйвер?
11. Что такое утилита?
12. Назовите наиболее популярные современные ОС для персональных компьютеров.
13. Какими достоинствами и недостатками обладает система *Linux*?
14. Какие ОС используются для мобильных устройств?
15. В чем состоят особенности операционной системы *Chrome OS*?
16. Что такое «операционная система реального времени»? Где применяются такие системы?
17. Что такое *UNIX*-подобные ОС? В чем их особенности?
18. Как ОС обменивается данными с внешними устройствами? В чем достоинства такой схемы?
19. Что происходит, когда ОС обнаружила новое устройство?
20. Какие программы относятся к утилитам?
21. Что такое файловая система? Зачем она нужна?
22. Какие задачи решает файловая система?
23. Что такое *кластер*?
24. Почему невыгодно хранить мелкие файлы в файловой системе с большим размером кластера?
25. Что улучшается при увеличении размера кластера?
26. Какие файловые системы используются в ОС *Linux, Windows* и *Mac OS X*?
27. Что такое журналируемая файловая система? Какие журналируемые системы вы знаете?
28. Почему файловая система **FAT32** считается устаревшей?
29. Что такое одноуровневая и многоуровневая файловые системы?
30. Что такое корневой каталог?
31. В чем отличие файловых систем в ОС *Linux* и *Windows*?
32. Где расположены каталоги пользователей в ОС *Linux*?
33. Какие символы используются как разделители при записи адреса файла в ОС *Linux* и *Windows*?
34. Что такое сетевая файловая система?

## *Задачи*

1. Определите, какое из указанных имен файлов удовлетворяет маске: **`?hel*lo.c?*`**

   а) **`hello.c`** б) **`hello.cpp`** в) **`hhelolo.cpp`** г) **`hhelolo.c`**
   д) **`hello.cc`** е) а**`hello.cpp`** ж) а**`helolo.c`** з) а**`helolo.cp`**

   (Ответ: в, е, з)

2. Пользователь **vasya** работает в ОС *Linux*. Перемещаясь из одного каталога в другой, он последовательно посетил каталоги **math**, **lections**, **professor** и оказался в своем домашнем каталоге. Каково полное имя каталога, из которого начал перемещение пользователь?
   (Ответ: **/home/vasya/professor/lections/math**)
3. Пользователь ОС *Windows*, перемещаясь из одного каталога в другой, последовательно посетил каталоги **Математика**, **Задания**, **c:**, **Классы**, 10**-А**. Каково полное имя каталога, из которого начал перемещение пользователь?
   (Ответ: **c:\Задания\Математика**)

# 6.4. Системы программирования

## 6.4.1. Зачем нужны системы программирования?

Процессор, выполняющий всю обработку данных, понимает только машинные команды (числовые коды). Чаще всего их записывают в шестнадцатеричном виде, например, так:

**B82301052500**

Чтобы понять, что делает этот код, нужно взять таблицу команд процессора и посмотреть, что означает каждая пара шестнадцатеричных цифр (байт).

Программы для первых компьютеров составляли именно в машинных кодах. Программирование было доступно только специалистам, отладка программы занимала очень много времени. Человек плохо воспринимает коды, поэтому для каждой машинной команды придумали символические обозначения. Например, приведенная выше программа – это две машинные команды для процессоров фирмы *Intel*, их можно записать так:

```
MOV AX,0123h
ADD AX,25h
```

Здесь **AX** – это имя *регистра* (ячейки памяти) процессора, команда **MOV** записывает в регистр новое значение, а команда **ADD** – добавляет число к содержимому ячейки. Буква «h» после числа означает, что оно записано в шестнадцатеричной системе счисления.

Чтобы переводить программу с такого языка в машинные коды, используют программы-*ассемблеры* (англ. *assembler* — рабочий-сборщик), а сам язык называется *языком ассемблера.* Этот язык – **машинно-ориентированный**, потому что он определяется набором команд конкретного процессора (ориентирован на машину).

Очевидно, что программировать на языке ассемблера тоже не очень удобно – нужно хорошо знать команды процессора, организацию памяти и т.п. Кроме того, каждый процессор имеет свою систему команд и свой язык ассемблера. Это значит, что программы на языке ассемблера *непереносимы* – программа, написанная для одного процессора, не будет работать на другом.

Людям хочется (в идеале) разговаривать с компьютером на естественном языке, не думая о том, какой процессор он использует. К сожалению, пока это невозможно. Сейчас для программирования чаще всего используют компромиссный вариант – *языки программирования высокого уровня* или *алгоритмические языки*. Это *формальные* языки, созданные специально для разработки программ. Команды строятся из слов естественного (чаще всего, английского) языка, каждая команда воспринимается однозначно в соответствии с установленными правилами.

Вспомним, что процессор может выполнить только программу, написанную в машинных кодах. Поэтому возникает задача: перевести программу, написанную на языке высокого уровня, в машинные коды. Для этого используют специальные программы-*трансляторы* (англ. *translator* – переводчик). Кроме трансляторов, в системы программирования входят и другие программы, о которых будет рассказано далее.

**Системы программирования** – это программные средства для создания и отладки новых программ.

## 6.4.2. Языки программирования

К 2010 году в мире было разработано более 8500 языков программирования[7]. Первой программисткой в мире считается Ада Лавлейс, которая в 1843 году написала программу для аналитической машины Чарльза Бэббиджа. В 1979 году в США был разработан язык программирования *Ада*, названный в её честь. Один из первых алгоритмических языков – *Фортран* – создан в 1957 году, однако он и сейчас применяется для научных вычислений.

Как вы уже знаете, языки программирования можно разделить на *языки низкого уровня* (машинно-ориентированнные, языки ассемблера) и *языки высокого уровня* (алгоритмические языки). По области применения можно выделить несколько групп современных языков программирования:

- профессиональные языки общего назначения[8]: *Java, C, C++, C#, Visual Basic, Delphi*;
- языки для программирования Интернет-сайтов: *PHP, Javascript, Perl, ASP*;
- языки для решения задач искусственного интеллекта: *Лисп, Пролог*;
- языки для обучения программированию: *Бейсик, Паскаль, Лого, Рапира*.

## 6.4.3. Трансляторы

Основа любой системы программирования – транслятор.

**Транслятор** – это программа, которая переводит в машинные коды тексты программ, написанных на языке высокого уровня.

Существует два типа трансляторов – *интерпретаторы* и *компиляторы*.

**Интерпретатор** анализирует текст программы по частям. Разобрав очередной фрагмент, он немедленно выполняет описанные в нем действия и переходит к обработке следующего фрагмента. *Достоинства* интерпретаторов:

- программы *переносимы* (программа будет работать в любой системе, где установлена программа-интерпретатор);
- удобно отлаживать программу.

Есть и существенные *недостатки*:

- программу невозможно выполнить, если не установлен интерпретатор;
- программы выполняются медленно (в цикле из 100 шагов каждая строчка 100 раз «разбирается» интерпретатором);
- в тех частях программы, которые не выполнялись во время отладки, могут оставаться синтаксические ошибки.

Второй тип трансляторов – **компиляторы**. Они, в отличие от интерпретаторов, сразу переводят всю программу в машинный код и строят исполняемый файл, готовый к запуску. *Достоинства* компиляторов:

- чтобы запустить программу, не нужно устанавливать транслятор;
- поскольку программа уже переведена в машинные коды, она выполняется значительно быстрее, чем при использовании интерпретатора.

*Недостатки* тоже есть:

[7] http://hopl.murdoch.edu.au

[8] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

- при любом изменении нужно ждать окончания компиляции (перевода в коды); это несколько затрудняет отладку;
- готовая программа будет выполняться только в той операционной системе, для которой она была создана[9].

Чтобы как-то совместить достоинства интерпретаторов и компиляторов, была предложена идея компиляции программы в некоторый промежуточный исполняемый код (*псевдокод*, *P-код*), а не сразу в команды конкретного процессора. Для выполнения такого псевдокода нужна специальная среда – *виртуальная машина*, которую в принципе можно разработать для любого процессора и любой операционной системы.

Программа сначала обрабатывается *компилятором*, который строит псевдокод, а потом этот псевдокод выполняется *интерпретатором*. Таким образом,

- при компиляции в псевдокод проверяются все синтаксические ошибки, поэтому при выполнении такую проверку делать не нужно; это значительно ускоряет работу программ в сравнении с интерпретацией;
- обеспечивается *переносимость* программ – можно выполнять программу (псевдокод) на любом компьютере, где есть виртуальная машина.

*Байт-код* – это разновидность псевдокода, в котором команда занимает 1 байт, а далее следуют ее аргументы (или их адреса). Современные версии интерпретируемых языков *Perl*, *PHP*, *Python* используют компиляцию в байт-код для ускорения выполнения программы.

Готовые программы на *Java* распространяются в виде байт-кода, поэтому для их выполнения необходимо установить виртуальную *Java*-машину. При этом для ускорения работы часто используется *JIT*-компиляция (англ. *JIT = just-in-time* – «в это самое время»), при которой байт-код «на лету» преобразуется в команды конкретного процессора. Тогда при повторном выполнении команды трансляция уже не нужна.

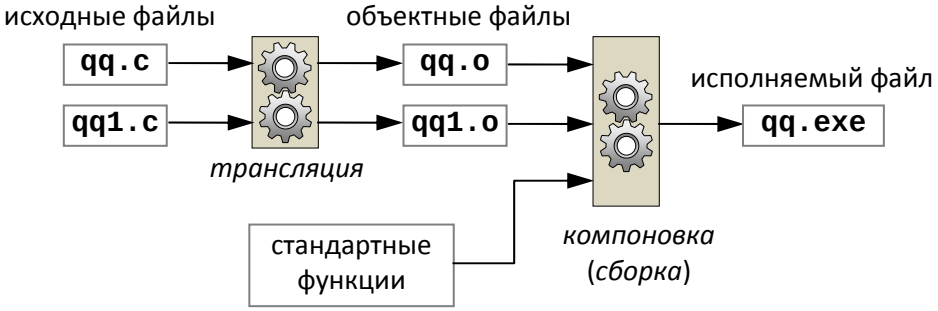
Аналогичный подход применяется в среде *.NET*, которую разработала фирма *Microsoft*. Одна из основных идей среды *.NET* – объединение программ, написанных на разных языках. В частности, разные части программы могут быть написаны на *C#, J#, VB.NET, Delphi.NET*, все они в конечном счете транслируются в байт-код на промежуточном языке *IL* (англ. *Intermediate Language*), который потом выполняется виртуальной машиной.

## 6.4.4. Состав системы программирования

Кроме трансляторов, в **состав системы программирования** обычно входят

- *компоновщик* (редактор связей, сборщик, англ. *linker*) – программа, которая собирает разные части (модули) создаваемой программы и функции из стандартных библиотек в единый исполняемый файл; вот, например, как собирается программа на языке Си, состоящая из двух модулей (исходные файлы **`qq.c`** и **`qq1.c`**):

[9] Многие программы, разработанные для ОС *Windows*, могут быть запущены в *Linux* с помощью программы-оболочки *Wine*.




- *отладчик* (англ. *debugger*[10]) – программа для поиска ошибок в других программах; отладчик позволяет:
  - выполнять программу в пошаговом режиме (по одной строчке);
  - выполнять программу до строчки, где установлен курсор;
  - устанавливать точки останова (англ. *breakpoints*);
  - просматривать и изменять значения переменных в памяти.
- *профилировщик* (англ. *profiler*) – программа, позволяющая оценить время работы каждой процедуры и функции («профиль» времени выполнения программы); используется для того, чтобы выяснить, какую именно процедуру нужно оптимизировать в первую очередь.

Любая система программирования включает *библиотеки стандартных подпрограмм*. Это набор готовых процедур и функций, которые можно вызывать из своей программы. Например, в большинстве языков программирования есть стандартные функции для вычисления синуса и косинуса. Они подключаются к программе на этапе сборки, это делает компоновщик.

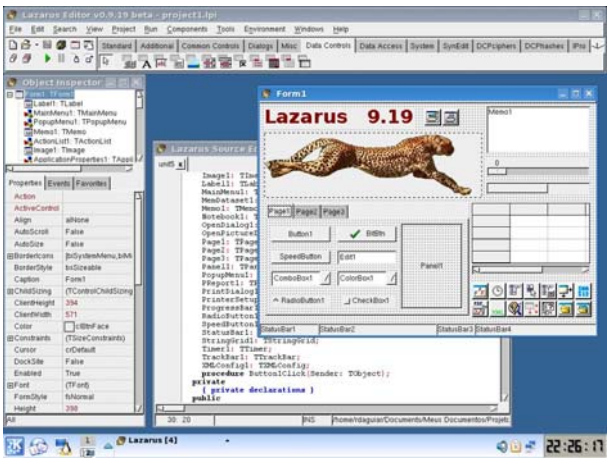
Многие программы используют одни и те же достаточно сложные системные функции (например, операции с окнами в графической среде). Если включать эти функции в код каждой программы, размеры исполняемых файлов намного увеличатся, из-за этого жесткий диск и память будут расходоваться неэффективно. Поэтому библиотеки таких функций хранятся на диске в виде отдельных файлов – *динамически подключаемых библиотек*, в системе *Linux* они имеют расширение **`.so`** (от англ. *shared objects* – разделяемые объекты), а в *Windows* – расширение **`.dll`**. Когда программа вызывает функцию из такой библиотеки, библиотека загружается в память и управление передается вызванной функции. Несколько программ могут обращаться к одной и той же копии библиотеки в памяти.

Набор стандартных структур данных и функций операционной системы, которые программисты могут использовать в прикладных программах, называется *интерфейсом прикладного программирования* (англ. *API = Application Programming Interface*). В ОС *Windows* применяется *Windows API*, а в *Unix*-подобных операционных системах – стандарт *POSIX* (англ. *Portable Operating System Interface for Unix* – переносимый интерфейс операционных систем *Unix*).

Сейчас для разработки программ чаще всего используются **интегрированные среды** (англ. *IDE = Integrated Development Environment*). В такую оболочку обычно входит текстовый редактор для набора текста программ, транслятор, компоновщик, отладчик и профилировщик.

Многие современные интегрированные среды позволяют строить интерфейс программы (расположение элементов в окне) с помощью мыши. Они называются *средами быстрой разработки приложений* (англ. *RAD = Rapid Application Development*) или средами визуального программирования. На рисунке показано окно RAD-среды *Lazarus* для программирования на объектном Паскале:

[10] Согласно одной из версий, это название связано с жучком, который попал между контактов реле компьютера Mark II в 1947 году. Дословно «*debug*» – «удаление жучков».




Среди профессиональных *RAD*-сред нужно, в первую очередь, назвать *Microsoft Visual Studio* (*msdn.microsoft.com/vstudio*). Её профессиональная версия – коммерческая, но все желающие могут бесплатно скачать и использовать ограниченную версию (*Express*) для учебных целей.

Большой популярностью пользуется также среды *Dev-C++* (*wxdsgn.sourceforge.net*) и *Delphi* (*embarcadero.com*). Кроссплатформенная среда *Code::Blocks* (*www.codeblocks.org*) распространяется бесплатно, существуют версии для *Windows*, *Mac OS X* и *Linux.*

## *Контрольные вопросы*

1. Что такое машинный код?
2. Зачем нужны системы программирования? Можно ли обходиться без них?
3. Что такое язык ассемблера? Почему он называется машинно-ориентированным?
4. Что такое язык программирования высокого уровня?
5. Как можно разделить языки программирования по области применения?
6. Зачем нужен транслятор?
7. Какие два типа трансляторов вы знаете? В чем их достоинства и недостатки?
8. Какие программы входят в системы программирования?
9. Зачем нужен компоновщик?
10. Что такое отладчик? Перечислите возможности отладчиков.
11. Что такое профилировщик? Зачем он нужен?
12. Что такое интегрированная среда разработки?
13. Что такое среда быстрой разработки (*RAD*-среда)?

# 6.5. Инсталляция программ

Большинство современных программ требуется устанавливать (*инсталлировать*, от англ. *install* – установить). Это связано с тем, что

- необходимо проверить, соответствует ли компьютер требованиям (к процессору, оперативной памяти, операционной системе и т.д.), которые обязательны для работы программы;
- программы содержат множество файлов, которые должны быть записаны на диск определенным образом;
- у пользователя должна быть возможность выбора нужных ему компонентов программы (остальные не устанавливаются);
- необходимо записать некоторые файлы в каталоги операционной системы (например, при установке драйверов устройств);
- необходимо настроить режимы работы программы с учетом особенностей компьютера;

- при установке коммерческих программ необходимо вводить *ключ* (серийный номер копии программы).

**Инсталляция** – это установка и настройка программы на компьютере пользователя.

Пользователь получает программу в виде *дистрибутива* (установочного пакета, от англ. *distribute* – распространять). Дистрибутив – это несколько файлов на CD- или DVD-диске, или один файл (который часто можно загрузить из Интернета). В таком виде программу невозможно использовать по прямому назначению. Данные в дистрибутиве обычно сжаты, распаковка происходит во время установки. Чтобы установить или удалить ПО, пользователь должен иметь права администратора компьютера.

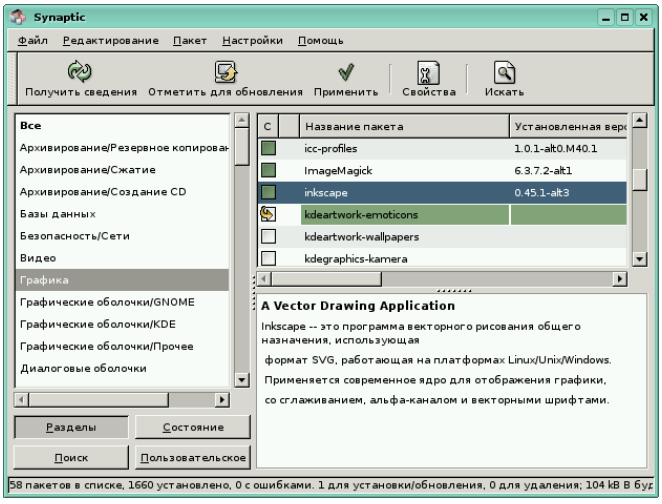
Иногда программное обеспечение может поставляться в виде исходного кода. Чтобы преобразовать его в готовую для выполнения программу, нужно использовать одну из систем программирования.

Обычно установка включает несколько этапов (некоторые из них могут отсутствовать):

- выбор компонентов программы, которые пользователь хочет установить;
- просмотр лицензионного соглашения (договора о возможности использования программы);
- ввод ключа (серийного номера) программы;
- определение каталога, в котором нужно разместить файлы программы;
- распаковка и копирование файлов на жесткий диск компьютера;
- настройка программы с помощью файлов конфигурации (или запись настроек в системный реестр в ОС *Windows*);
- создание ярлыков для запуска программы в меню и/или на *Рабочем столе*.

В операционной системе *Linux* программы чаще всего распространяются в виде *пакетов* (файлы с расширениями **.rpm** или **.deb**, в зависимости от сборки) или в исходных кодах. Для установки пакетов используются утилиты **apt-rpm** и **apt-get**. Они позволяют проверить *зависимости* пакетов: устанавливается не только указанный пакет, но и другие пакеты, необходимые ему для работы.

Начинающим пользователям *Linux* проще всего использовать графические программы для работы с пакетами (*менеджеры пакетов*), например, *Aptitude* или *Synaptic*. В них можно мышкой отметить пакеты, которые требуется установить, обновить или удалить.




В ОС *Windows* для установки программ используется служба *Windows Installer*, которая работает с установочными пакетами – файлами в формате **.msi**. Дистрибутив также может пред-

ставлять собой программу (файл с расширением **.exe**), которая содержит все необходимые данные и при запуске «ведёт» пользователя через все этапы установки.

Программы для операционной системы *Mac OS X* распространяются в виде пакетов (файлов с расширением **.pkg**). Для работы с ними в ОС включена программа *Installer*.

Системные администраторы, которым приходится устанавливать ПО на большое количество компьютеров, нередко используют автоматическую установку (без участия человека) или удаленную установку (через сеть).

Установка ПО – это дополнительное неудобство для пользователей. Поэтому особой популярностью пользуются *переносимые программы* (англ. *portable applications*). Их не нужно устанавливать, они могут быть просто скопированы на жесткий диск компьютера или запущены прямо с CD-, DVD- или флэш-диска. К этой группе относятся многие небольшие бесплатные программы для ОС *Windows* и *Mac OS X*.

Часто делают специальные переносимые версии «обычных» программ (которые необходимо устанавливать); их можно найти, например, на сайте *portableapps.com*. Они очень полезны для тех, кто часто работает на разных компьютерах и хочет использовать привычный набор программ. Пользователь может запустить программу со своего флэш-диска, причем все настройки (например, закладки в браузере *Opera Portable*) сохраняются в папке программы.

Существуют ознакомительные версии операционных систем, которые загружаются прямо с CD- или DVD-диска (англ. *live disc* – «живой диск») или флэш-диска. Они не меняют данные на жестком диске, а все файлы, необходимые для работы, размещаются в оперативной памяти компьютера. Многие «живые диски» позволяют работать с файлами на винчестере, поэтому с их помощью можно сохранить данные в случае отказа установленной на компьютере операционной системы. Большинство *live*-систем строится на базе *Linux*; часто с «живого диска» можно сразу же выполнить полную установку ОС на компьютер.

### *Контрольные вопросы*

1. Что такое инсталляция? Почему она необходима для многих современных программ?
2. Что происходит во время инсталляции?
3. Что такое дистрибутив? В чем его отличие от установленной программы?
4. Что такое менеджер пакетов?
5. Как устанавливаются программы в разных операционных системах?
6. Какие методы используются для массовой установки ПО?
7. Что такое переносимая программа?
8. Как можно познакомиться с операционной системой, не устанавливая ее на жесткий диск?

## 6.6. Правовая охрана программ и данных

### 6.6.1. Авторские права

По законам большинства стран компьютерные программы и данные охраняются *авторским правом*. Это значит, что автор (или правообладатель, например, фирма, в которой работает автор) могут ограничивать распространение и использование программы.

В Конституции Российской Федерации записано, что «интеллектуальная собственность охраняется законом» (ст. 41 ч. 1). Интеллектуальная собственность – это права на результаты творческой деятельности человека. Эти права детально определены в Гражданском кодексе РФ (часть IV, «Права на результаты интеллектуальной деятельности и средства индивидуализации»).

Авторские права **распространяются** на

- *программы для компьютеров* (включая подготовительные материалы, а также звук, графику и видео, которые получаются с помощью программы);
- *базы данных* (массивы данных, специально организованные для поиска и обработки с помощью компьютеров).

**Не охраняются** авторским правом

- *алгоритмы и языки программирования*;
- *идеи и принципы*, лежащие в основе программ, баз данных, интерфейса;
- *официальные документы*.

Важно, что охраняется форма, а не содержание. Это значит, авторские права получает не тот, кто придумал *метод* решения задачи, а тот, кто написал *программу*, которая решает задачу на основе предложенного алгоритма.

Согласно российским законам об авторском праве, автор – это физическое лицо (не организация). Авторское право

- возникает «в силу создания» продукта и не требует формальной регистрации, хотя при желании автор может зарегистрировать программу в государственных органах;
- обозначается знаком ©, после которого записывается фамилия автора и год первого выпуска программы, например, © *Иванов, 2008*;
- действует в течение жизни и 70 лет после смерти автора;
- передается по наследству.

Автор получает **личные права**:

- право авторства (право считаться автором);
- право на имя (право выпускать программу под своим именем, псевдонимом или анонимно);
- право на неприкосновенность программы и ее названия;

и **имущественные права**: осуществлять или разрешать

- выпуск программы в свет;
- копирование в любой форме;
- распространение;
- изменение (в т.ч. перевод на другой язык).

Серьезные нарушения авторских прав могут попасть под действие Уголовного кодекса РФ (ст. 146, «Нарушение авторских и смежных прав»). Уголовная ответственность наступает при крупно ущербе (более 50 000 руб.). Присвоение авторства (*плагиат*) наказывается лишением свободы на срок до 6 месяцев. В случаях незаконного использования, а также приобретения и хранения объектов авторского права (например, дисков с нелицензионными программами) в целях сбыта срок лишения свободы может достигать 5 лет (при особо крупном ущербе).

## 6.6.2. Типы лицензий на использование ПО

Право на использование программы дает документ (договор), который называют **лицензией** (лат. *litentia*) или *лицензионным соглашением*. Это соглашение между правообладателем и пользователем, где четко определены права и обязанности сторон. Как правило, в соответствии с лицензией пользователь без дополнительного разрешения автора может

- установить программу на 1 компьютер (или так, как указано в договоре);
- вносить изменения, необходимые для работы программы на компьютере пользователя; исправлять явные ошибки;
- изготовить копию, чтобы можно было восстановить программу в случае сбоя;
- передать программу другому лицу вместе с лицензией.

Программы, которые получены и используются в соответствии с законом, называют *лицензионными*. Если же при создании копии были нарушены авторские права, ее называют *контрафактной* или *пиратской*.

По **типу лицензий** можно разделить ПО на 4 типа:

- коммерческое;
- условно-бесплатное (англ. *shareware*);
- бесплатное (англ. *freeware*);
- свободное ПО (англ. *open source* – ПО с открытым кодом).

Значительная часть ПО является **коммерческим**, поскольку создается с целью получения прибыли путем продажи экземпляров. За каждую копию коммерческого ПО нужно платить (покупать лицензию); исходный код, как правило, не распространяется. Обычно фирмы предусматривают скидки при закупке большого количества лицензий (лицензии на организацию) и скидки для образовательных учреждений. Зарегистрированные пользователи программ имеют право на бесплатную техническую поддержку – консультации по телефону или электронной почте. Типичный пример коммерческого ПО – операционная система *Windows*.

Часто разработчики дают возможность бесплатно скачать пробную (англ. *trial*) версию программы из Интернета и попробовать, как она работает (англ. «*try before you buy*» – «попробуй, прежде чем купить»). Такие программы называют **условно-бесплатными** (англ. *shareware*). Пробные версии всегда имеют какие-то ограничения, например:

- ограниченный срок работы (обычно 30 дней);
- ограниченное количество запусков;
- ограничение функций (например, невозможно сохранить результаты на диск);
- встроенный рекламный блок;
- всплывающие сообщения с призывом заплатить автору деньги за программу.

Обычно в лицензионном соглашении указывается, что пробная версия не может быть использована для коммерческих целей и профессиональной работы.

К условно-бесплатным можно отнести многие популярные программы, у которых есть пробные версии. Например, на ноутбуки часто устанавливается пробная версия пакета *Microsoft Office*, которую можно бесплатно использовать 60 дней. В Интернете можно скачать пробные версии векторного редактора *Corel Draw,* почтовой программы *TheBat*, всех программ фирмы *Adobe*.

Для решения большинства задач можно найти **бесплатные** программы (англ. *freeware*). Это значит, что лицензионное соглашение не требует никаких выплат автору. Бесплатные программы можно свободно скачать из Интернета и использовать, однако чаще всего коммерческое использование и изменение запрещается, исходные коды не распространяются. Иногда фирмы бесплатно распространяют ограниченные версии коммерческих программ. К бесплатным программам относятся браузеры *Opera, Chrome* и *Safari*, программа для записи CD и DVD-дисков *CDBurber XP*, множество небольших утилит.

Наибольшие возможности предоставляет **свободное ПО** (англ. *open source* – открытый исходный код). Авторы свободных программ передают пользователю не только исполняемую программу, но и ее исходный код, и предоставляют:

- право использовать программу в любых целях;
- право изучать исходный код и изменять его для своих целей;
- право свободно распространять программу;
- право улучшать программу и распространять измененные версии на тех же условиях.

Программное обеспечение, которое не удовлетворяет этим критериям, называется собственническим или *проприетарным* (англ. *proprietary* — частное).

В первые годы на свободное ПО никакие документы не оформлялись, но возникла необходимость защищать права авторов юридически. Поэтому сейчас свободное ПО чаще всего распространяется под лицензией *GPL* (англ. *General Public Licence* – «генеральная общественная лицензия»). В ней перечислены приведенные выше права и установлено одно ограничение: разрешается распространять измененную версию только как свободное ПО (запрещается делать его несвободным).

К свободному программному обеспечению относится операционная система *Linux*, браузер *Mozilla Firefox*, почтовая программа *Mozilla Thunderbird,* графические редакторы *Gimp* и *Inkscape*, архиватор 7ZIP (*www.7-zip.org*), среда для быстрой разработки программ *Code::Blocks* и многие другие программы.

Как ни странно, свободное ПО может приносить прибыль. Например, некоторые фирмы оказывают платную техническую поддержку по развертыванию и настройке системы *Linux*. Второй вариант – предоставление коммерческой лицензии в том случае, если открытый исходный код используется в коммерческих программах.

## ? *Контрольные вопросы*

1. Что обозначает термин «авторские права»?
2. Что такое интеллектуальная собственность?
3. Какие законы РФ регулируют вопросы, связанные с авторскими правами?
4. На какие творческие результаты распространяются авторские права?
5. Что не охраняется авторским правом?
6. Что значит положение «охраняется форма, а не содержание»?
7. Нужно ли регистрировать авторское право?
8. Кто может быть правообладателем согласно российским законам?
9. Как обозначается авторское право в документе?
10. Как действует авторское право после смерти автора?
11. Какие личные и имущественные права имеет автор?
12. Какие наказания предусмотрены за нарушение авторских прав?
13. Василий Пупкин разработал программу *SuperPuper* в 2010 году. Как он должен правильно обозначить в тексте программы своё авторское право?
14. Что такое лицензионное соглашение?
15. Что обычно может делать пользователь программы, не спрашивая дополнительного разрешения автора?
16. Что такое лицензионная программа?
17. Что такое контрафактная программа?
18. Какие типы лицензий на ПО сейчас существуют?
19. Что такое условно-бесплатная программа? Какие ограничения она может иметь?
20. Какие программы относятся к бесплатным (*freeware*)?
21. Что такое свободное ПО? Почему оно распространяется по лицензии?
22. Какие свободы предоставляет пользователю лицензия GPL?
23. Какие ограничения предусматривает лицензия GPL?
24. Что такое собственническое (проприетарное) ПО? Чем оно отличается от коммерческого?
25. Как можно сделать разработку свободного ПО коммерчески выгодным?
26. Какие типы ПО можно законно загружать из Интернета?
27. Можно ли, не спрашивая автора (правообладателя)
    a) скопировать картинку с веб-страницы на свой компьютер?

б) послать скопированную картинку другу?

в) разместить на своем сайте отсканированную книгу?

г) привести на сайте цитату из книги с указанием источника?

д) разместить на своем сайта картинку с другого сайта?

(Ответ: можно а, б, г)

28. Можно ли размещать в Интернете, не спрашивая авторов:

а) произведения А.С. Пушкина?

б) записи популярных исполнителей?

в) документы, принятые Государственной Думой?

г) описание алгоритма решения квадратного уравнения?

д) базу данных сотовых телефонов?

(Ответ: можно а, в, г)

# Глава 7. Компьютерные сети

## 7.1. Основные понятия

### 7.1.1. Что такое компьютерная сеть?

**Компьютерная сеть** – это группа компьютеров, соединенных линиями связи.

Для передачи данных между компьютерами могут использоваться:

- телефонная линия;
- специальные электрические кабели;
- оптоволокно (нить из стекла или пластика, по которой идет свет);
- радиоволны (в беспроводных сетях).

Объединяя компьютеры в сеть, мы получаем следующие **преимущества**:

- быстрый *обмен данными* между компьютерами без использования сменных носителей (CD и DVD-дисков, флэш-дисков);
- *совместное использование ресурсов*:
  - общих данных, которые могут быть размещены на одном компьютере;
  - программ, которые могут запускаться с другого компьютера;
  - внешних устройств (например, все компьютеры в сети могут использовать один принтер);
- электронную почту и другие способы сетевого общения (чаты, форумы и т.п.).

В то же время существуют и **недостатки**:

- необходимы *денежные затраты* на сетевое оборудование (кабели, вспомогательные устройства) и программное обеспечение (например, операционную систему специального типа);
- *снижается безопасность* данных, поэтому компьютеры, на которых ведутся секретные разработки, не должны быть подключены к сети;
- для настройки сети и обеспечения ее работы необходим высококвалифицированный специалист – *системный администратор*.

*Системный администратор* (его также называют «сисадмин» или «админ») обычно решает следующие задачи:

- устанавливает и настраивает программное обеспечение;
- устанавливает права доступа пользователей к ресурсам сети;
- обеспечивает защиту информации;
- предотвращает потерю данных в случае сбоя электропитания;
- периодически делает резервные копии данных на DVD-дисках, внешних жестких дисках или магнитных лентах;
- устраняет неисправности в сети.

### 7.1.2. Какие бывают сети?

По «радиусу охвата» обычно выделяют следующие типы компьютерных сетей:

- *персональные* сети (англ. *PAN = Personal Area Network*), объединяющие устройства одного человека (сотовые телефоны, карманные компьютеры, смартфоны, ноутбук и т.п.) в радиусе не более 30 м; самый известный стандарт таких сетей – *Bluetooth*;
- *локальные сети* (англ. *LAN = Local Area Network*), объединяющие, как правило, компьютеры в пределах одного или нескольких соседних зданий;

- *корпоративные сети* (англ. *Corporate network*) – сети компьютеров одной организации (возможно, находящие в разных районах города или даже в разных городах);
- *городские сети* (англ. *MAN = Metropolitan Area Network*), объединяющие компьютеры в пределах города;
- *глобальные сети* (англ. *WAN = Wide Area Network*), объединяющие компьютеры в разных странах; типичный пример глобальной сети – *Интернет*.

### 7.1.3. Серверы и клиенты

В любой сети одни компьютеры используют ресурсы других. Для описания роли компьютеров в обмене данными вводят два термина: *сервер* и *клиент*.

**Сервер –** это компьютер, предоставляющий свои ресурсы (файлы, программы, внешние устройства и т.д.) в общее использование.

**Клиент** – это компьютер, использующий ресурсы сервера.

Обычно *серверы* – это специально выделенные мощные компьютеры, которые используются только для обработки запросов большого числа клиентских компьютеров (*рабочих станций*) и, как правило, включены постоянно. Как правило, они находятся в отдельных помещениях, куда пользователи не имеют доступа; это повышает защищенность данных.

В крупных локальных сетях используют несколько серверов, каждый из которых решает свою задачу:

- *файловый сервер* хранит данные и обеспечивает доступ к ним;
- *сервер печати* обеспечивает доступ к общему принтеру;
- *почтовый сервер* управляет электронной почтой;
- *серверы приложений* (например, серверы баз данных) выполняют обработку информации по запросам клиента.

Сервер получает запросы от клиентов, ставит их в очередь, и после выполнения посылает каждому клиенту ответ с результатами выполнения запроса. Задача клиента – послать серверу запрос в определенном формате, и после получения ответа вывести ответ на монитор пользователя. Такая технология называется «**клиент-сервер**». Ее используют, например, все веб-сайты в Интернете: программа-браузер (клиент) посылает запрос серверу и выводит его ответ (веб-страницу) на экран.

Часто понятия «сервер» и «клиент» относятся не к компьютерам, а к программам. Например, на одном и том же компьютере может работать веб-сервер (программа, которая отправляет веб-страницы по запросу пользователей) и почтовый клиент (программа, которая обращается к почтовому серверу на другом компьютере для отправки и получения сообщений электронной почты).

### 7.1.4. Обмен данными

Для того, чтобы люди могли полноценно общаться, нужно, чтобы они говорили на одном языке. Эта аналогия действует и для компьютерных систем, где вместо слова «язык» используется понятие *протокол*.

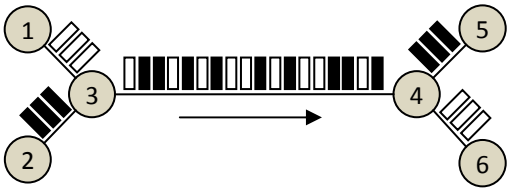
**Протокол –** это набор правил и соглашений, определяющих порядок обмена данными.

Можно объединить в одну сеть устройства, которые используют разные протоколы обмена данными. Для этого нужно устройство-«переводчик», которое называют *шлюзом*. Задача шлюза – «перевести» принятые данные в формат другого протокола. Шлюзы часто используются для связи

между промышленными сетями (измерительной аппаратурой, датчиками) и сетями персональных компьютеров.

В современных сетях пересылаемые данные делятся на *пакеты*. Дело в том, что чаще всего одна линия связи используется для обмена данными между несколькими узлами. Если передавать большие файлы целиком, то получится, что сеть будет заблокирована, пока не закончится передача файла. Кроме того, в этом случае при сбое весь файл нужно передавать заново, это увеличивает нагрузку на сеть.

Если передавать отдельные пакеты, время ожидания сокращается до времени передачи одного пакета (это доли секунды), нагрузка на линию связи становится более равномерной. По сети одновременно передаются пакеты, принадлежащие нескольким файлам. На рисунке по одной линии связи (между узлами 3 и 4) одновременно выполняется передача данных от узла 2 к узлу 5 (пакеты обозначены черными прямоугольниками) и от узла 1 к узлу 6 (белые прямоугольники).




Вместе с каждым пакетом передается его *контрольная сумма* – число, найденное по специальному алгоритму и зависящее от всех данных пакета. Узел-приемник рассчитывает контрольную сумму полученного блока данных и, если она не сходится с контрольной суммой, указанной в пакете, фиксируется ошибка и этот пакет (а не весь файл!) передается, как правило, еще раз.

Казалось бы, чем меньше размер пакета, тем лучше. Однако это не так, потому что любой пакет кроме «полезных» данных содержит служебную информацию: адреса отправителя и получателя, контрольную сумму. Поэтому в каждом случае есть некоторый оптимальный размер пакета, который зависит от многих условий (например, от уровня помех, количества компьютеров в сети, передаваемых данных и т.д.). Чаще всего для обмена данными в локальных сетях и в Интернете используются пакеты размером не более 1,5 Кб.

### ? *Контрольные вопросы*

1. Что такое компьютерная сеть?
2. Какие каналы связи могут использоваться в сетях?
3. Какие преимущества дает объединение компьютеров в сеть? Что ухудшается?
4. Что входит в обязанности системного администратора?
5. Как разделяются сети по области действия?
6. Что такое персональные сети?
7. Что такое сервер и клиент?
8. Может ли один компьютер выполнять роли сервера и клиента?
9. Что такое протокол? Зачем нужны протоколы?
10. Что такое шлюз?
11. Зачем данные, передаваемые по сети, делятся на пакеты?
12. Почему размер пакета не должен быть очень маленьким?

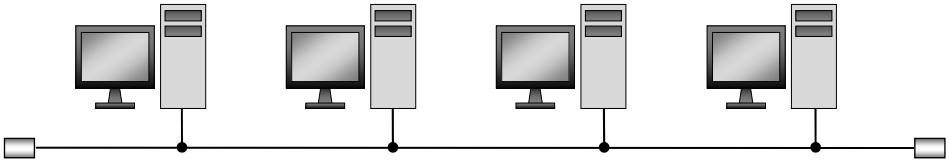
## 7.2. Структура (топология) сети

Для обмена данными в сети очень важно, как именно связаны компьютеры линиями связи. В этом разделе мы кратко рассмотрим три основные структуры (или *топологии*) сетей с разными

схемами соединений между компьютерами: «общую шину», «звезду» и «кольцо». Каждая из них обладает своими достоинствами и недостатками.

## 7.2.1. Общая шина

**Шина** – это линия связи, которую несколько устройств используют для обмена данными. В схеме «общая шина» компьютеры (рабочие станции) подключены к одному кабелю с помощью специальных разъемов. Чаще всего такая сеть – одноранговая (хотя это не обязательно).



Чтобы сигнал не отражался от концов кабеля (и не шел в обратную сторону), их закрывают заглушками (*терминаторами*).

Так как существует всего одна линия связи, компьютеры передают данные по очереди. Сигнал, который идет по шине, получают все компьютеры, но каждый из них обрабатывает только те данные, которые ему предназначены.

*Достоинства* схемы «общая шина»:

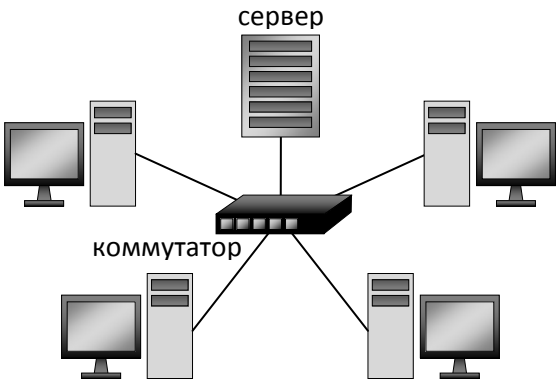
- самая простая и дешевая схема;
- небольшой расход кабеля;
- легко подключать новые рабочие станции;
- при выходе из строя любого компьютера сеть работает.

*Недостатки*:

- при разрыве кабеля или выходе из строя терминатора вся сеть не работает;
- низкий уровень безопасности (каждая рабочая станция имеет доступ ко всем данным, которые идут по сети);
- один канал связи на всех (при увеличении числа компьютеров падает скорость передачи; возможны конфликты, когда две рабочие станции хотят передать данные одновременно);
- сложно обнаруживать неисправности (неясно, где проблема);
- ограничение размера (не более 185 м, при большей длине нужны усилители сигнала).

## 7.2.2. Звезда

В схеме «звезда» есть центральное устройство, через которое идет весь обмен информацией. На практике чаще всего в центре находится *коммутатор* (его часто называют «свитч», от англ. *switch* – переключать). Коммутатор передает принятый пакет только адресату, а не всем компьютерам в сети.




*Достоинства* схемы «звезда»:

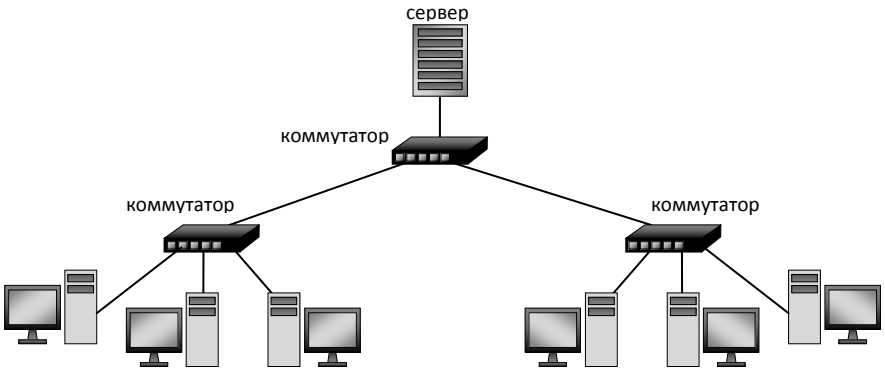
- при выходе из строя любой рабочей станции сеть работает;
- высокий уровень безопасности (каждая рабочая станция получает только «свои» данные, а не все, что передается по сети);
- простой поиск неисправностей и обрывов (сразу ясно, с каким компьютером нет связи).

*Недостатки*:

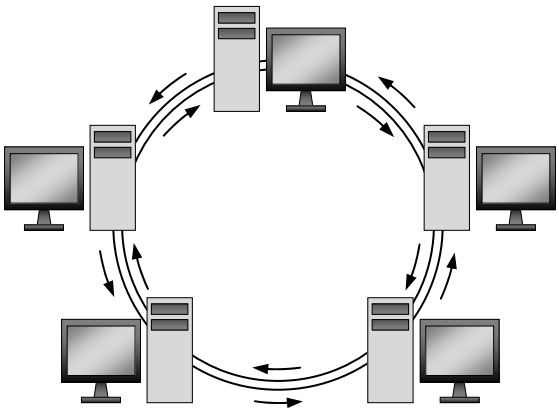
- большой расход кабеля, высокая стоимость;
- при выходе из строя коммутатора вся сеть не работает;

- количество рабочих станций ограничено количеством портов коммутатора.

Иерархическая структура (*дерево*) – это фактически многоуровневая схема «звезда». Она нередко применяется в крупных сетях, которые состоят из сотен компьютеров.




## 7.2.3. Кольцо



В схеме «кольцо» каждый компьютер соединении с двумя соседними, причем от одного он только получает данные, а другому только передает. Таким образом, пакеты движутся по кольцу в одном направлении. Для повышения надежности обычно используют «двойное кольцо», в котором каждая линия связи дублируется. По второму кольцу данные могут передаваться в обратном направлении.

Каждый компьютер участвует в передаче сигнала и усиливает его, поэтому размер сети может быть очень велик, ограничено лишь расстояние между соседними узлами (для оптоволоконных сетей – до 2 км).

*Достоинства* схемы «кольцо»:

- большой размер сети (до 20 км);
- надежная работа при большом потоке данных, конфликты практически невозможны;
- не нужно дополнительное оборудование (коммутаторы).

*Недостатки*:

- для подключения нового узла нужно останавливать сеть;
- низкая безопасность (все данные проходят через каждый компьютер);
- сложность настройки и поиска неисправностей.

В современных сетях кольцевая схема чаще всего используется в сочетании со «звездой»: компьютеры соединяются с коммутатором по схеме «звезда», а коммутаторы между собой объединяются в кольцо.

## ? *Контрольные вопросы*

1. Что такое «топология сети»?
2. Опишите структуру, достоинства и недостатки сетей типа «общая шина», «звезда» и «кольцо».
3. Какую структуру вы предложили бы использовать для школьной сети (рассмотрите разные ситуации)?

# 7.3. Локальные сети

## 7.3.1. Типы локальных сетей

**Локальная сеть** объединяет компьютеры в одном или нескольких соседних зданиях.

Для того, чтобы организовать локальную сеть, нужно использовать сетевую операционную систему, которая поддерживает:

- сетевое оборудование (например, сетевые карты);
- сетевые протоколы обмена данными;
- доступ к удалённым ресурсам (папкам, принтерам и т. п.).

Эти возможности существуют почти во всех современных ОС (*Windows, Linux, Mac OS X* и др.).

В небольших организациях часто используют **одноранговые** сети (на 10-15 компьютеров), в которых все компьютеры равноправны, каждый может выступать как в роли клиента, так и в роли сервера. Пользователь может открыть *общий доступ* к некоторым ресурсам своего компьютера (папкам, принтерам), то есть предоставить их в совместное использование. На каждый общий ресурс можно установить пароль для чтения и/или записи.

*Достоинства* одноранговых сетей:

- низкая стоимость;
- простота настройки и обслуживания;
- независимость компьютеров друг от друга;
- сеть может работать при любом количестве подключенных компьютеров;
- не нужно сложное программное обеспечение.

*Недостатки*:

- сложность управления и настройки прав доступа (нет единого центра, нужно создавать учетные записи для всех пользователей на каждом компьютере);
- низкая защищенность данных;
- резервное копирование данных нужно выполнять на каждом компьютере.

С увеличением количества компьютеров становится очень сложно управлять одноранговыми сетями, а также обеспечивать безопасность данных. Поэтому в крупных организациях используют **сети с выделенными серверами**, в которых один или несколько мощных компьютеров играют роль серверов (пользователи на них не работают), а остальные (клиенты, рабочие станции) используют их ресурсы. Такие сети обладают серьезными *достоинствами*:

- основная обработка данных выполняется на серверах;
- через сеть передаются только нужные данные;
- упрощается модернизация системы: достаточно переоборудовать сервера;
- повышенный уровень безопасности: права на доступ к данным устанавливаются на сервере;
- клиенты могут использовать различное оборудование и операционные системы;
- резервное копирование данных нужно выполнять только на серверах.

В то же время есть *недостатки*:

- высокая стоимость серверного оборудования;
- сложность настройки и обслуживания сервера.
- при выходе сервера из строя служба, которую он обеспечивал, не работает (например, недоступны хранящиеся на нем данные).

Для поддержки сервера во многих случаях требуется специальная *серверная операционная система* (*Windows Server, Linux, FreeBSD, Solaris*). В таких ОС основное внимание уделяется стабильной и надежной работе с большим количеством клиентов, а не пользовательскому интерфей-

су. Они содержат развитые средства для поддержки совместной работы пользователей, веб-узла, электронной почты, систем управления базами данных и т.п. Важная возможность сетевых ОС – *терминальный доступ*, при котором пользователь со своей рабочей станции запускает программу на сервере и получает на своем экране результаты ее работы.

## 7.3.2. Беспроводные сети

Беспроводные сети используются там, где создание кабельной сети невозможно или невыгодно, например, за пределами зданий, в исторических помещениях и т.п. Кроме того, с их помощью мобильные компьютеры (ноутбуки, карманные компьютеры) могут легко подключаться к сети и получать выход в Интернет. Для обмена данными применяются радиоволны сверхвысокой частоты.

Самый популярный стандарт для беспроводных **персональных** сетей – *Bluetooth*, обеспечивающий обмен данными между 8 устройствами. Это могут быть карманный и обычный компьютер, мобильный телефон, ноутбук, принтер, цифровой фотоаппарат, мышь, клавиатура, наушники. Радиус действия такой сети обычно[1] не более 20 м, он зависит от мощности передатчиков, а также от преград и помех. Максимальная скорость обмена данными – около 700 кбит/с. Ожидается, что в будущем с помощью *Bluetooth* можно будет связывать любые электронные устройства, включая холодильники и стиральные машины. Среди достоинств *Bluetooth* – низкая стоимость, удобство и простота в использовании, высокая надежность. Для обеспечения защиты данных от перехвата приемник и передатчик 1600 раз в секунду одновременно меняют частоту сигнала.

В **локальных** беспроводных сетях применяют стандарт *WiFi* (от англ. *Wireless Fidelity* – «беспроводная точность»). Для объединения компьютеров в беспроводную сеть чаще всего используют специальное устройство – *точку доступа* (англ. *WAP = Wireless Access Point*, точка беспроводного доступа). К одной точке доступа обычно подключаются не более 15 компьютеров (при увеличении этого количества падает скорость передачи данных). Часто главная задача точки доступа – обеспечить мобильным компьютерам доступ к кабельной сети и выход в Интернет.


Современные операционные системы (*Windows, Mac OS X, Linux*) поддерживают технологию и устройства *WiFi*.

Согласно стандарту, скорость передачи данных может быть от 0,1 Мбит/с до 480 Мбит/с. Обычно радиус действия сети *WiFi* в помещениях не превышает 45 м, а вне зданий – 450 м.

Технология *WiFi* широко используется как в офисах, так и в домашних сетях. Бесплатный выход в Интернет через *WiFi* предоставляют многие библиотеки, университеты, кафе (для привлечения посетителей), гостиницы. Такие зоны доступа называют «*хот-спот*» (от англ. *hot spot* – «горячая точка»). В некоторых гостиницах и аэропортах эта услуга платная.

Сети *WiFi* работают в радиоэфире, так что любое приемное устройство, настроенное на нужную частоту, может перехватить сигнал. Поэтому в беспроводных сетях важно обеспечить за-

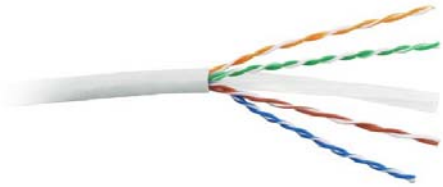
[1] Стандарт предусматривает радиус действия до 100 м.

щиту данных. Для этого используют специальные алгоритмы кодирования сигналов, шифрование и другие методы.

## 7.3.3. Сетевое оборудование

В современных локальных сетях используется технология пакетной передачи данных, которая называется *Ethernet* (от лат. *aether* — эфир). Для связи компьютеров могут использоваться электрические кабели или оптоволокно, в существующем стандарте определяются скорости передачи данных 10 Мбит/с, 100 Мбит/с, 1 Гбит/с и 10 Гбит/с.

Для того, чтобы подключить компьютер к кабельной сети, он должен иметь *сетевую карту* (сетевой адаптер, англ. *network interface card*). В современных материнских платах настольных компьютеров и в ноутбуках обычно уже есть встроенная сетевая карта, поддерживающая стандарт *Ethernet* со скоростью до 1 Гбит/с.



Сетевой кабель «витая пара»

Разъем RJ-45

Сетевая карта

Для локальных сетей чаще всего используется восьмижильный кабель «витая пара», который представляет собой четыре пары скрученных проводов. Восьмиконтактный разъем с защелкой часто называют RJ-45. Сейчас наиболее распространены сети со скорость 100 Мбит/с, построенные с помощью кабеля «витая пара» по схеме «звезда» (с коммутатором в центре).

Для передачи данных на большие расстояния применяют *оптоволоконные кабели,* в которых информация передается с помощью светового луча. Свет идет внутри кабеля, отражаясь от стенок стеклянного или пластикового цилиндра-световода.

*Коммутаторы* или *свитчи* используются для объединения компьютеров в единую сеть по схеме «звезда», которая чаще всего применяется на практике:

Коммутаторы

В отличие от концентраторов («хабов», от англ. *hub*), коммутаторы передают пакет только тому узлу, которому он предназначен, а не дублируют на все выходы. Компьютер соединяют с коммутатором отрезком кабеля с двумя разъемами RJ-45, который называется «*патч-корд*» (от англ. *patching cord* — соединительный шнур).

Обычно все компьютеры, входящие в локальную сеть, получают доступ в Интернет через один канал связи. Для связи локальной сети с Интернетом необходим *маршрутизатор* или *роутер* (англ. *router*). Задача маршрутизатора – определить дальнейший маршрут движения пакета и направить его на нужный выход (порт). Для этого используются *таблицы маршрутизации*, в которых записано, куда направлять пакеты в зависимости от адреса назначения. Роль маршрутизатора в локальной сети может выполнять обычный компьютер с несколькими сетевыми картами. С точки зрения компьютеров локальной сети, маршрутизатор является *шлюзом* – устройством, которое связывает две сети, локальную и глобальную.

При создании беспроводных сетей компьютеры должны иметь *адаптеры WiFi,* они обычно встроены в современные портативные компьютеры. Если встроенного адаптера нет, можно использовать дополнительный адаптер, который подключается к USB-порту. Для связи компьютеров в беспроводной сети и для обеспечения доступа в Интернет используют точки доступа и беспроводные маршрутизаторы.

USB-адаптер WiFi

Точка доступа

Маршрутизатор

## ? *Контрольные вопросы*

1. Что такое локальная сеть?
2. Какими возможностями должны обладать сетевые операционные системы?
3. Когда лучше использовать сеть с выделенными серверами?
4. Какие задачи решают компьютеры-серверы?
5. Чем отличаются серверные ОС от клиентских?
6. Что такое терминальный доступ?
7. В каких случаях применяются стандарты беспроводных сетей *Bluetooth* и *WiFi*?
8. Как обеспечивается защита данных в беспроводных сетях?
9. Что такое точка доступа? зона доступа *WiFi*?
10. Назовите преимущества и недостатки беспроводных сетей.
11. Какое сетевое оборудование необходимо для кабельных сетей?
12. Что такое коммутатор?
13. Что такое патч-корд?
14. Что такое маршрутизатор? Какую роль он выполняет в локальной сети?
15. Какие оборудование необходимо для создания беспроводной сети?

# 7.4. Сеть Интернет

## 7.4.1. Что такое Интернет?

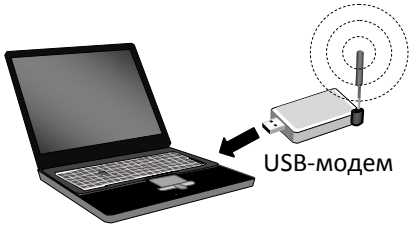
Слово *Интернет* (англ. *Internet*), обозначающее глобальную компьютерную сеть, возникло как сокращение *Inter*connected *Net*works — объединённые сети или «сеть сетей». В отличие от локальных сетей, ее «элементы» – не отдельные компьютеры, а сети.

Информация в Интернете хранится на *серверах*, связанных скоростными линиями связи (оптоволоконными, спутниковыми). Практически все услуги Интернета основаны на использовании технологии «клиент-сервер»: программа-клиент на компьютере пользователя запрашивает информацию, сервер возвращает ответ.

Пользователь получает доступ к глобальной сети через *провайдера* – фирму, локальная сеть которой непосредственно связана с Интернетом. Существует несколько **способов подключения** к провайдеру:

- с помощью *модема* по обычной телефонной линии; скорость обмена данными не превышает 56 кбит/с, поэтому такой способ уже практически не используется;

- с помощью *ADSL-модема*, который также использует телефонную линию, но позволяет одновременно разговаривать по телефону и работать в Интернете; скорость передачи данных из Интернета к пользователю может достигать 25 Мбит/с, однако на телефонной станции необходимо устанавливать дополнительное оборудование (*сплиттер*, отделяющий низкочастотный телефонный сигнал от высокочастотного сигнала, передающего цифровые данные);
- через *локальную сеть* провайдера (если она существует в вашем доме); в этом случае телефонная линия не задействована;
- с помощью *беспроводных модемов* (USB-модемов), которые используют сети сотовых операторов и работают везде, где доступна мобильная связь; скорость передачи данных для сетей 3-го поколения (англ. *3G = 3rd generation*) достигает 10 Мбит/с, а в сетях 4-го поколения (4G) – до 1 Гбит/с.




## 7.4.2. Краткая история

В 1960-е годы в министерстве обороны США начали разработку компьютерной системы передачи данных, которая получила название *ARPANET* (англ. *Advanced Research Projects Agency Network*, сеть агентства передовых исследований). В основу этого проекта были положены следующие идеи:

- сеть объединяет компьютеры, имеющие разное аппаратное и программное обеспечение;
- при подключении новой сети не требуется переделка существующей части;
- нет единого центра (такая сеть называется *распределенной*), это обеспечивает живучесть в случае выхода из строя любого узла;
- пакетная передача данных: передаваемые данные разбиваются на пакеты небольшого размера, одна линия связи используется для одновременной передачи нескольких блоков данных.

В 1969 году состоялся первый обмен данными по сети между компьютерами, установленными в Калифорнийском университете и Стэнфордском исследовательском центре. В 1971 году была создана программа для работы с электронной почтой, которая сразу стала очень популярной. Начиная с 1973 года, к новой сети подключаются университеты и колледжи не только США, но и Европы. В 1983 году сеть разделяется на две части: военную сеть *MilNet* и общедоступную сеть, которая получила название Интернет.

История российского Интернета начинается с 1990 года, когда была организована почтовая сеть «*Релком*» – первый провайдер в Советском Союзе.

В 1991 году британский ученый Тим Бернес-Ли разработал систему обмена данными в виде *гипертекста* – текста с активными ссылками на другие документы. Сейчас она называется Всемирной паутиной (англ. *WWW = World Wide Web*) и является самой мощной службой Интернета. Многие ошибочно считают, что Интернет и Всемирная паутина – это одно и то же. В самом деле это не так, потому что в Интернете есть и другие службы – электронная почта, обмен файлами, чаты, форумы и т.д.


Т. Бернес-Ли

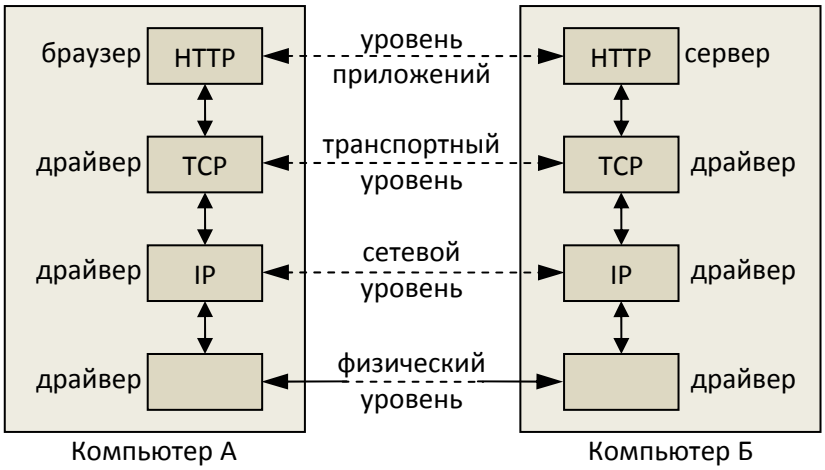
## 7.4.3. Протоколы

Вы уже знаете, что для передачи информации источник и приемник должны использовать один и тот же протокол – набор правил и соглашений, определяющих порядок обмена данными в сети. В Интернете в качестве стандарта принят протокол TCP/IP, разработанный в 1974 году. Во-

обще говоря, это не один протокол, а целое семейство, название которого происходит от двух самых важных протоколов – TCP (англ. *Transfer Control Protocol* – протокол управления передачей) и IP (англ. *Internet Protocol* – межсетевой протокол).

Попробуем разобраться, почему для работы в Интернете нужно использовать несколько протоколов. Предположим, что браузер на компьютере А запрашивает веб-страницу с сервера, который находится на компьютере Б. «Разговор» между браузером и сервером идет с помощью протокола HTTP (англ. *HyperText Transfer Protocol* – протокол передачи гипертекста). Браузер и веб-сервер не могут связаться напрямую. Чтобы послать запрос серверу, браузер передает адрес сервера и текст запроса операционной системе, которая вызывает драйвер протокола TCP.

Задача драйвера TCP – установить соединение с удаленным компьютером и обеспечить доставку данных. Передаваемый блок данных разбивается на пакеты (размер пакета обычно не превышает 1,5 Кбайта), и каждый пакет передается на следующий уровень – драйверу протокола IP, который посылает его в сеть по указанному адресу.




Обычно при работе в Интернете компьютеры А и Б напрямую не связаны, поэтому задача протокола IP – определить узел-*маршрутизатор*[2], на который нужно отправить пакет, чтобы он дошел до компьютера Б. Когда маршрут определен, пакет (с добавленной служебной информацией) передается на физический уровень (например, в сетевую карту), где передается просто как цепочка байт. Протоколы физического уровня могут быть любыми, они не определены в стандарте.

Протокол IP не гарантирует доставку пакетов, поэтому драйвер TCP должен (с помощью установленного соединения) проверить, что данные получены, и в случае сбоя передать пакет повторно. На другом конце соединения драйвер TCP «собирает» пакеты в единый блок данных и передает на уровень приложения (запрос дошел до сервера).

Таким образом, в Интернете используется четырехуровневая система протоколов, каждый из которых «занимается своим делом»:

1) *уровень приложений* – формат запросов и ответов, которыми обмениваются программы;
2) *транспортный уровень* (TCP) – правила пакетной передачи блоков данных без учета их содержания;
3) *сетевой уровень* (IP) – правила выбора маршрута для отдельных пакетов без гарантии их доставки;

[2] Маршрутизаторы обмениваются информацией друг с другом, сообщая о выходе из строя или подключении каких-то участков сети. Таблицы маршрутизации обновляются автоматически, так что при выборе маршрута пакетов учитывается фактическая структура сети в данный момент.

4) *физический уровень* – правила передачи отдельных байтов по кабельной, оптоволоконной или другой линии связи.

На уровне приложений (который находится «ближе всего» к пользователю) чаще всего применяются протоколы:

HTTP – для передачи веб-страниц;

FTP – для передачи файлов;

SMTP – для передачи на сервер сообщений электронной почты;

POP3 или IMAP – для приема сообщений электронной почты с сервера.

Существуют и другие протоколы (для чатов, новостных групп и т.п.), но все они используют TCP и IP соответственно на транспортном и сетевом уровнях.

### *Контрольные вопросы*

1. Как в Интернете используется технология «клиент-сервер»?
2. Что такое провайдер?
3. Расскажите, как можно получить доступ в Интернет. В чем достоинства недостатки разных способов?
4. Какие идеи были положены в основу глобальной компьютерной сети?
5. Как вы думаете, почему в 1983 году сеть разделили?
6. Что такое гипертекст?
7. Чем отличаются понятия «Интернет» и «Всемирная паутина»?
8. Какое семейство протоколов используется в сети Интернет?
9. Объясните, почему применяются несколько уровней протоколов. Расскажите о роли протоколов разных уровней.
10. Какова роль узлов-маршрутизаторов?
11. Как обеспечивается гарантированная доставка сообщений в Интернете?
12. Назовите наиболее известные протоколы уровня приложений. Где они применяются?

## 7.5. Адреса в Интернете

### 7.5.1. IP-адреса

В Интернете любые два компьютера могут связаться друг с другом. Для этого каждый из них должен иметь уникальный адрес. С «точки зрения» компьютеров, удобнее работать с числовыми адресами, каждый из которых занимает одинаковое место в памяти. Такие адреса (их называют *IP-адресами*) состоят из четырех чисел в интервале от 0 до 255, например,

**192.168.104.115**

В этих числах закодированы номер сети и номер компьютера в сети. Для того, чтобы выделить эти две части из IP-адреса, используют маски-шаблоны. Маска – это тоже четыре числа в интервале 0-255, но она строится особым образом, по принципу «*n* единиц, потом – нули» в двоичном коде. Например, маска 255.255.255.0 в двоичном виде запишется так:

**11111111.11111111.11111111.00000000**

В ней сначала идут 24 единицы, а потом – нули. Это значит, что первые 24 бита адреса – номер сети (192.168.104.0), а оставшиеся 8 бит – номер компьютера (узла) в этой сети (115). Можно использовать другую запись, которая значит то же самое («/24» говорит о том, что в маске 24 единицы):

**192.168.104.115/24**

В такой сети может быть 254 узла, а не 256, как можно было бы ожидать. Дело в том, что младший адрес (192.168.104.0) используется для обозначения всей сети, а старший (192.168.104.255) – для так называемой широковещательной рассылки (сообщение отправляется всем компьютерам данной сети). Все узлы с адресами 192.168.104.* (здесь * – любое число от 1 до 254), находятся в той же сети, что и данный компьютер.

Тот же самый адрес с другой маской имеет совершенно иной смысл. Например, маска 255.255.255.248 в двоичной системе содержит 29 единиц и 3 нуля. На рисунке область, отведенная номеру сети, выделена штриховой рамкой.

| | | |
|---|---|---|
| **адрес** | **192 . 168 . 104 . 115** | |
| | **11000000.10101000.01101000.01110**011$_2$ | |
| **маска** | **11111111.11111111.11111111.11111**000$_2$ | |
| | **255 . 255 . 255 . 248** | |

Узел с адресом 192.168.104.115/29 – это узел номер 3 ($011_2$) в сети 192.168.104.112:

**192.168.104.112 = 11000000.10101000.01101000.01110000$_2$**

Существуют так называемые «серые» адреса, которые не используются в «большом» Интернете. Например, диапазоны адресов:

**192.168.0.0–192.168.255.255 (192.168.0.0/16)**
**172.16.0.0–172.31.255.255 (172.16.0.0/12)**
**10.0.0.0–10.255.255.255 (10.0.0.0/8)**

предназначены только для локальных сетей. Адреса 127.0.0.0 – 127.255.255.255 служат для обращения к своему компьютеру (обычно для этой цели применяют адрес 127.0.0.1).

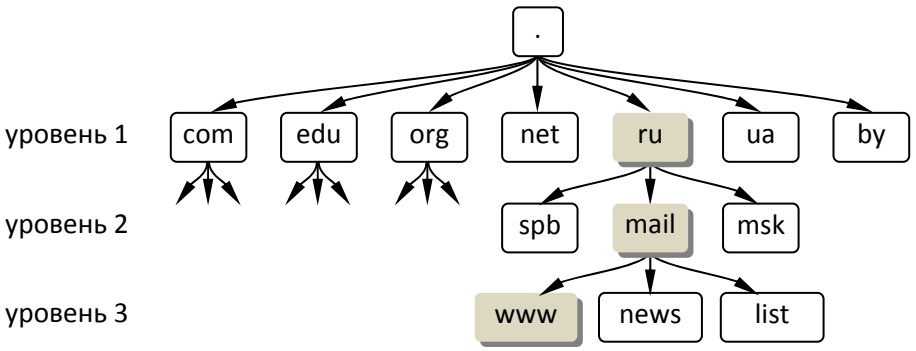
В связи с бурным развитием Интернета адресов, которые можно использовать при таком кодировании, скоро не будет хватать для всех желающих. Поэтому предполагается, что произойдет постепенный переход на новую (шестую) версию протокола IP, которая обозначается как IPv6. В ней на каждый адрес отводится 128 бит, а не 32. Уже сейчас существует более 1600 сетей, где применяется IPv6; он поддерживается всеми современными операционными системами и производителями оборудования. Полный переход на IPv6 займет несколько лет, он потребует больших денежных затрат и замены всех устаревших устройств.

Вообще говоря, IP-адрес присваивается не компьютеру, а *интерфейсу* – каналу передачи данных (сетевой карте, модему). Поэтому один компьютер может иметь несколько IP-адресов, например, если на нем установлено две сетевых карты (или сетевая карта и модем).

## 7.5.2. Доменные имена

В отличие от компьютеров, человеку неудобно работать с числовыми адресами. Они плохо запоминаются, при вводе IP-адреса легко сделать ошибку, а заметить ее иногда достаточно сложно. Поэтому в 1984 году была разработана *система доменных имен* (англ. *DNS = Domain Name System*), которая позволила использовать символьные имена сайтов, например, **www.mail.ru**.

Домен (англ. *domain* – область, район) – это группа символьных адресов в Интернете. Домены образуют многоуровневую структуру (*иерархию*, *дерево*), вкладываются друг в друга, как матрешки:




Чем-то такая система напоминает почтовый адрес, в котором указывается страна, город, улица, дом, квартира.

Точка в корне дерева – это *корневой домен*. Домены первого уровня (они называются *доменные зоны*) могут обозначать тип организации, например[3]:

**.com** – коммерческие организации;
**.edu** – образовательные организации (университеты, колледжи);
**.gov** – правительство США;
**.biz** – бизнес;
**.info** – информационные сайты;
**.name** – личные сайты;
**.museum** – музеи;
**.net** – сетевые организации;
**.org** – разные организации.

Кроме того, каждая страна имеет свой двухбуквенный домен первого уровня.

Распределением IP-адресов и доменов первого уровня занимается международная организация ICANN (англ. *Internet Corporation for Assigned Names and Numbers*). Российский домен **.ru** был зарегистрирован в 1994 году.

Свободный домен второго уровня может зарегистрировать любой желающий за небольшую плату. Такие услуги оказывают специальные организации – *регистраторы доменных имен*, например, *RU-Center* (*nic.ru*). Домены третьего уровня часто можно получить бесплатно. Например, сайт *narod.yandex.ru* предоставляет всем желающим место под сайт и домен третьего уровня вида **ivanov.narod.ru**.

Раньше в доменных именах было разрешено использовать только латинские буквы, цифры и дефис. Сейчас можно регистрировать домены, содержащие другие знаки, входящие в кодировку *UNICODE*, например, буквы русского алфавита. За Россией закреплен домен **.рф**, в котором все желающие могут зарегистрировать домены второго уровня.

Таким образом, сейчас в Интернете используется две системы адресов: IP-адреса и доменные имена. Чтобы установить соответствие между ними, на специальных серверах, которые называются *DNS-серверами*, хранятся таблицы, состоящие из пар «IP-адрес – доменное имя». Их задача – по запросу компьютера-клиента вернуть IP-адрес для заданного доменного имени (или наоборот).

Для того, чтобы компьютер смог установить связь с сетью, в настройках сетевой карты (или модема) указывается IP-адрес, маска сети и адрес DNS-сервера. Иногда эти данные определяются автоматически при подключении к сети провайдера.

Когда вы вводите адрес сайта (доменное имя) в адресной строке браузера, сначала отправляется запрос на DNS-сервер, цель которого – определить IP-адрес сервера. Если это удалось, на-

[3] Здесь перечислены не все тематические домены первого уровня.

правляется запрос на получение веб-страницы, причем драйвер протокола IP использует полученный IP-адрес, а не доменное имя.

Заметим, что одному доменному имени может соответствовать несколько IP-адресов. такой прием применяется для распределения нагрузки на сайты с большим количеством посетителей (например, *www.yandex.ru, www.google.com*). Таким образом, соответствие между доменными именами и IP-адресами можно описать как «многие ко многим»: с одним IP-адресом может быть связано несколько доменных имен и наоборот.

## 7.5.3. Адрес ресурса (URL)

Точный адрес имеет не только каждый компьютер в Интернете, но и каждый документ. Для такого адреса чаще всего используется английское сокращение *URL = Uniform Resource Locator* – универсальный указатель ресурса. Типичный URL-адрес состоит из четырех частей: протокола, имени сервера (или его IP-адреса), каталога и имени документа (файла). Такую систему записи придумал в 1990 году создатель Всемирной паутины Т. Бернес-Ли. Например, адрес

**`http://example.com/doc/new/vasya-new.htm`**

включает

1) протокол HTTP – протокол для обмена гипертекстовыми документами (это веб-страница);
2) доменное имя сервера **`example.com`**;
3) каталог на сервере **`/doc/new`**;
4) имя файла **`vasya-new.htm`**.

Иначе говоря, для обращения к документу **`vasya-new.htm`**, который находится в каталоге **`/doc/new`** на сервере **`example.com`** нужно использовать протокол HTTP.

Иногда каталог и имя файла не указывают, например, **`http://example.com`**. Это означает, что мы обращаемся к главной странице сайта. Она может иметь разные имена, в зависимости от настроек сервера (чаще всего – **`index.htm`**, **`index.html`**, **`index.php`**).

Для скачивания и загрузки файлов часто используется протокол FTP, тогда адрес документа выглядит примерно так:

**`ftp://files.example.com/pub/new/vasya-new.zip`**

## 7.5.4. Тестирование сети

При работе с сетью возникает несколько характерных задач, связанных с проверкой доступности компьютеров и правильности работы службы DNS. Для этой цели администраторы используют утилиты, работающие из командной строки. В *Linux* для работы в командной строке нужно запустить *Терминал* (*Konsole*), а в *Windows* – командный процессор *cmd*.

Во-первых, определим IP-адрес и настройки своего компьютера. Для этого в *Windows* используется команда **`ipconfig`**, результат работы которой может быть, например, таким:

```
Подключение по локальной сети - Ethernet адаптер:
  IP-адрес:      192.168.45.48
  Маска подсети: 255.255.255.0
  Основной шлюз: 192.168.45.5
```

Последняя строка показывает адрес *шлюза* – узла, на который отправляются все пакеты, в которых указан IP-адрес получателя, не входящий в локальную сеть (в данном случае – в сеть 192.168.45.0/24). В операционной системе *Linux* (и других *Unix*-подобных системах) для той же цели используется команда **`ifconfig`**[4].

[4] В некоторых версиях, например, в *AltLinux*, ее нужно вызывать как **`/sbin/ifconfig`**.

Команда **ping** посылает на указанный узел пакеты и ждет ответных пакетов (по протоколу ICMP). По команде

```
ping 192.168.45.5
```

мы можем получить, например, такой результат

```
Обмен пакетами с 192.168.45.5 по 32 байт:
Ответ от 192.168.45.5: число байт=32 время=5мс
Ответ от 192.168.45.5: число байт=32 время<1мс
Превышен интервал ожидания для запроса.
Ответ от 192.168.45.5: число байт=32 время<1мс
```

Для каждого пакета указано время получения отклика. В данном случае связь есть, но третий пакет был потерян. Если пакеты не доходят, связи с узлом нет или администратор запретил отвечать на запросы по протоколу ICMP.

Теперь проверим, как работает DNS-сервер. Определим IP-адрес сервера *www.altlinux.org* с помощью команды **nslookup**:

```
nslookup www.altlinux.org
```

Ответ может быть таким:

```
Server:  UnKnown
Address: 172.16.172.19
Name:    www.altlinux.org
Address: 194.107.17.79
```

Это значит, что в настройках сетевого соединения установлен DNS-сервер 172.16.172.19, который не имеет доменного имени (англ. *UnKnown* – неизвестный). Как следует из ответа этого DNS-сервера, узел *www.altlinux.org* имеет IP-адрес 194.107.17.79.

Если DNS-сервер доступен, в команде **ping** можно указывать не только IP-адрес, но и доменное имя, например,

```
ping www.google.ru
```

Утилита **tracert** (в *Linux* – **traceroute**) показывает, по какому маршруту идут пакеты к заданному сайту. Например, результат выполнения команды

```
tracert www.yandex.ru
```

может выглядеть примерно так:

```
Трассировка маршрута к www.yandex.ru [87.250.251.3]
с максимальным числом прыжков 30:
  1 <1 мс  <1 мс  <1 мс  192.168.45.5
  2  3 мс   2 мс   3 мс  193.85.124.15
  3 10 ms  12 ms  11 ms  aurora-spb-ix.yandex.net [194.85.177.90]
  4 16 ms  10 ms  12 ms  aluminium-vlan934.yandex.net [213.180.208.12]
  5 19 ms  23 ms  12 ms  silicon-vlan901.yandex.net [77.88.56.125]
  6 30 ms  32 ms  31 ms  l3link-iva1-ugr1.yandex.net [213.180.213.4]
  7 18 ms  21 ms  24 ms  www.yandex.ru [87.250.251.3]
Трассировка завершена.
```

Эти данные говорят о том, что пакет достигает узла *www.yandex.ru* за 7 прыжков («хопов»), то есть проходит 6 промежуточных узлов-маршрутизаторов. Каждому узлу посылается 3 пакета, в ответе указано время прохождения каждого из них. Если узел имеет доменное имя, оно записывается слева от IP-адреса. С помощью утилиты **tracert** (**traceroute**) можно определить, где именно нарушена связь.

## ? *Контрольные вопросы*

1. Сколько места в памяти занимает IP-адрес?
2. Что такое маска? Как она строится?

3. Как вы думаете, могут ли два компьютера иметь одинаковый IP-адрес? Ответ обоснуйте.
4. Какие IP-адреса используются для локальных сетей?
5. Какие IP-адреса используют для обращения к своему компьютеру?
6. Почему становится необходимым переход на протокол IPv6?
7. Может ли компьютер иметь несколько IP-адресов? В каких случаях?
8. Зачем нужны доменные адреса?
9. Что такое домен?
10. В виде какой структуры можно представить доменную систему имен?
11. Что такое корневой домен?
12. Что такое доменные зоны? Какие они бывают?
13. Какие домены вы можете зарегистрировать (если они свободны)?
14. Как вы думаете, будут ли пользоваться популярностью домены с русскими буквами? Ответ обоснуйте.
15. Что такое DNS-сервер? Какие функции он выполняет?
16. Что такое URL? Из каких частей он обычно состоит?
17. Приведите примеры URL для веб-страниц, рисунков, файлов на FTP-серверах.
18. Определите IP-адрес своего компьютера и маску подсети. Сколько компьютеров может быть в такой сети?
19. Что такое шлюз? Какие пакеты направляются на шлюз?

## *Задачи*

1. Какие из приведенных последовательностей могут быть масками:
   а) 255.255.255.128 д) 255.255.255.192
   б) 255.255.128.64 е) 255.255.224.0
   в) 255.255.128.128 ж) 255.255.224.192
   г) 255.255.128.0 з) 255.255.248.0

   (Ответ: а, г, д, е, з)
2. Почему в сети с маской /24 может быть только 254 узла, а не 256?
3. Для каждого приведенного адреса определите номер сети, номер узла, наибольшее возможное количество компьютеров в сети:
   а) 192.168.104.109/30 д) 92.60.65.180/26
   б) 172.16.12.12/29 е) 118.212.123.1/24
   в) 193.25.5.136/28 ж) 85.16.172.127/23
   г) 10.10.40.15/27 з) 134.5.169.172/22

   (Ответы: а – 192.168.104.108/30, 1, 2; б – 172.16.12.8/29, 4, 6; в – 193.25.5.128/28, 8, 14; г – 10.10.40.0/27, 15, 30; д – 92.60.65.128/26, 52, 62; е – 118.212.123.0/24, 1, 255; ж – 85.16.172.0/23, 127, 510; з – 134.5.168.0/22, 428, 1022)
4. Определите маску сети минимального размера, в которую входит N компьютеров для значений N = 16, 25, 32, 112. (Ответ: /27, /27, /26, /25)
5. Проверьте, есть ли у вашего компьютера связь с узлом www.ya.ru. Определите среднее время отклика.
6. С помощью утилиты **`nslookup`** определите IP-адрес сервера *www.google.ru*. Что особенного вы обнаружили?
7. Определите маршрут, по которому идут пакеты с вашего компьютера на сайт *kremlin.ru*. Сколько «прыжков» составляет этот маршрут?

# 7.6. Всемирная паутина

## 7.6.1. Что такое всемирная паутина?

Всемирная паутина или «*веб*» (англ. *WWW = World Wide Web*) – это служба для доступа к гипертекстовым документам (веб-страницам), хранящимся на серверах. Сейчас *WWW* – наиболее популярная служба Интернета.

**Гипертекст** – это текст, в котором есть активные ссылки (*гиперссылки*) на другие документы.

Гиперссылки обычно подчеркиваются и выделяются цветом (по умолчанию – синим). Если щелкнуть левой кнопкой мыши на гиперссылке, в окно браузера загружается документ, на который указывает ссылка.

На современных веб-страницах встречается не только текст, но и графика, звук, видео, причем каждый элемент может быть гиперссылкой. Такие документ называются *гипермедиа*.

**Сайт (веб-сайт)** – это группа веб-страниц, которые расположены на одном сервере, объединены общей идеей и связаны с помощью гиперссылок.

Чтобы сайт стал доступен другим компьютерам, на сервере должна быть запущена специальная программа – веб-сервер. Наиболее популярные веб-серверы:

-  *Apache* (*httpd.apache.org*), свободный веб-сервер для различных операционных систем, включая *Windows, Linux, Mac OS*;
- *IIS* (*www.iis.net*) – коммерческий веб-сервер для *Windows*;
- *nginx* (*sysoev.ru/nginx*) – бесплатный веб-сервер и почтовый сервер для крупных сайтов (есть версии для *Windows* и *UNIX*-подобных систем).

Для просмотра веб-страниц на экране используются программы-*браузеры* (*Internet Explorer, Mozilla Firefox, Safari, Chrome, Opera*). Браузер отправляет веб-серверу запрос, содержащий URL-адрес документа (веб-страницы, рисунка, файла и т.п.), а сервер в ответ передает запрошенные данные. Обмен происходит по протоколу HTTP.

Особенность современной веба – привлечение пользователей к наполнению сайтов информацией и ее корректировке. Это привело к появлению термина «*Web 2.0*», которым иногда обозначают современный этап развития Всемирной паутины.

Сайты, использующие технологии *Web 2.0*, как правило, требуют регистрации пользователей, для этого необходим действующий адрес электронной почты. Любой желающий может создать «личную зону» с собственными настройками и хранить там файлы, фотографии, видео, заметки. Другие могут комментировать эти материалы.

Пользователи объединяются в группы (сообщества) для того, чтобы вместе обсуждать интересующие их вопросы. Часто участники могут оценивать сообщения друг друга, таким образом, изменяется «репутация» (или «карма») участников, появляется некоторое соперничество.

*Социальные сети*:  *ВКонтакте* (*vkontakte.ru*),  *Одноклассники* (*www.odnolkassniki.ru*),  *Facebook* (*www.facebook.com*) для многих стали местом общения с друзьями и одноклассниками.

Появились специальные сайты, где пользователи могут вести *блоги* – сетевые дневники (  *www.livejournal.com*,  *www.blogspot.com*). Влияние блогов настолько возросло, что их стали приравнивать к средствам массовой информации.

Активно развиваются *вики-системы* (англ. *wiki*) – веб-сайты, структуру и содержимое которых пользователи могут изменять с помощью инструментов, которые есть на самом сайте. Самый известный вики-сайт – это свободная энциклопедия *Википедия* (русская версия размещена на сайте *ru.wikipedia.org*).

С одной стороны, *Web 2.0* расширяет возможности пользователей. С другой стороны, нужно понимать, что размещенные данные хранятся где-то на серверах, куда в принципе может получить доступ злоумышленник. Известны случаи массовых взломов учетных записей в социальных сетях и блогах. Поэтому не следует размещать в Интернете информацию, опубликование которой как-то может вам повредить, даже теоретически.

Фактически на таких сайтах пользователи сами заполняют базу данных о себе, своих друзьях, карьере и даже личной жизни. Изучая и анализируя эти данные, владельцы сайтов и спецслужб получают возможность манипулировать людьми, используя полученную информацию в своих целях, например, для рекламы товаров. Очень часто социальные сети используются для распространения вредоносных программ и рекламных сообщений (*спама*).

В начале XXI века Т. Бернес-Ли (автор Всемирной паутины) предложил развивать веб в направлении создания «семантической паутины» (*Web 3.0*), в которой все документы связаны по ключевым словам, как в базе данных. Это потребует переделки всех сайтов (добавления специальных смысловых «ярлыков» – *тэгов*), что обеспечит возможность полностью автоматического поиска и обработки информации. Вместо ручного поиска человек будет использовать программу-агент, которая подберет возможные ответы на вопрос и даст ему право окончательного выбора. С другой стороны, поиск нового типа позволит автоматически собирать всю информацию о личности (или организации), так что область его «частного пространства», «личной тайны» значительно уменьшится.

## 7.6.2. Поиск информации в Интернете

В Интернете сейчас содержится огромное количество данных, при этом найти нужную информацию иногда оказывается достаточно сложно.

**Поисковая система** – это веб-сайт, который предназначен для поиска информации в Интернете.

В начале развития Интернета, когда сайтов было немного, веб-мастера (создатели сайтов) составляли списки ссылок на интересные сайты. Когда ссылок стало много, их начали объединять в группы по темам. В результате развития этой идеи появились *каталоги*.

**Каталог** (англ. *web directory*) – это разбитый по темам список ссылок на сайты с их кратким описанием.

В каталогах обычно используют многоуровневую группировку ссылок (*дерево*): в каждой из крупных тем (*Новости, Наука, Образование* и др.) есть разделы, в разделах – подразделы и т.д.

Первым крупным сайтом-каталогом стал *Yahoo* (*www.yahoo.com*), созданный в 1995 году. За рубежом очень популярен также *Открытый каталог* (*www.dmoz.org*), который поддерживается международным сообществом редакторов. Самые крупные из российских каталогов – *Яндекс-каталог* (*yaca.yandex.ru*) и *Каталог@Mail.ru* (*list.mail.ru*).

Каталоги заполняются вручную людьми-экспертами (редакторами каталога), каждый из которых отвечает за определенный раздел. Кроме того, веб-мастера могут предложить редакторам свои сайты для включения в каталог (бесплатно или платно).

Ссылки в каталогах, как правило, точно соответствуют разделу, в котором они размещены. Однако редакторы физически не могут посетить и проверить все новые сайты, которые ежедневно появляются в Интернете, поэтому часто случается, что нужный вам сайт не включен в каталог. Поэтому возникла естественная идея – заставить компьютерную программу искать новые сайты и автоматически анализировать информацию на их страницах. Так появились *поисковые машины*.

**Поисковая машина** – это автоматическая система, которая хранит информацию обо всех известных ей веб-страницах и выдает по запросу адреса тех из них, где встречаются введенные поль-

зователем ключевые слова.

Робот-браузер поисковой машины (его часто называют «паук», англ. *crawler*) выкачивает с сайтов веб-страницы, переходя по всем встречающимся на них ссылкам[5].



Затем другая программа (*индексный робот*) удаляет из текста страницы всю служебную информацию (например, команды оформления) и строит *индекс*, похожий на книжный (см. рисунок справа) – алфавитный список слов, для каждого из которых хранится адрес веб-страницы и номер (или номера) этого слова на странице.

Пользователь вводит в запросе *ключевые слова*, которые его интересуют.

**Ключевые слова** – это набор слов и выражений, которые отражают требуемую информацию.

Поисковый робот с помощью индекса находит те страницы, где встречаются эти слова.

Каждая поисковая машина имеет свой язык, который позволяет составлять сложные запросы, например, исключать некоторые ключевые слова из поиска или искать одно из заданного набора слов. Во многих системах для обозначения логической операции «ИЛИ» (нужно одно из указанных слов) используется символ |, а для логической операции «И» (нужны оба слова) – символ &. Если нужно найти словосочетание, в запросе его берут в кавычки.

Обычно поисковый робот находит тысячи страниц, соответствующих запросу. Они выдаются пользователю в том порядке, который определяется разработчиками. Чаще всего учитывается *цитируемость* – число ссылок с других сайтов на эту страницу; чем ссылок больше, чем выше «ранг» данной страницы и тем выше она расположена в результатах поиска.

Самая крупная международная поисковая машина – *Google* (*www.google.com*). В России лидирующие позиции занимает *Яндекс* (*www.yandex.ru*). Эти системы умеют искать не только текст, но также картинки и видео (правда, при поиске изображений используется текстовая информация рядом с ними). Поисковая система *TinEye* (*tinneye.com*) позволяет находить изображения, похожие на образец.

## ? *Контрольные вопросы*

1. Что такое гипертекст? гиперссылка?
2. Чем отличаются понятия «гипертекст» и «гипермедиа»?
3. Что такое веб-сервер? браузер?
4. Как работает технология «клент-сервер» при просмотре веб-страниц?
5. Что такое *Web 2.0*? Расскажите о достоинствах и недостатках этой технологии.
6. Что такое сообщество?
7. Что такое блог? Чем, на ваш взгляд, объясняется популярность блогов?
8. Что такое вики-сайт?
9. Что такое «семантическая паутина»? Каковы, на ваш взгляд, достоинства и недостатки этой идеи?
10. Что такое поисковая система? Какие типы поисковых систем вы знаете?
11. Назовите известные вам каталоги.
12. В чем достоинства и недостатки каталогов?
13. Что такое поисковая машина? Как она работает?
14. Что такое «ключевые слова»?
15. Какую информацию, кроме текста, умеют искать поисковые машины?

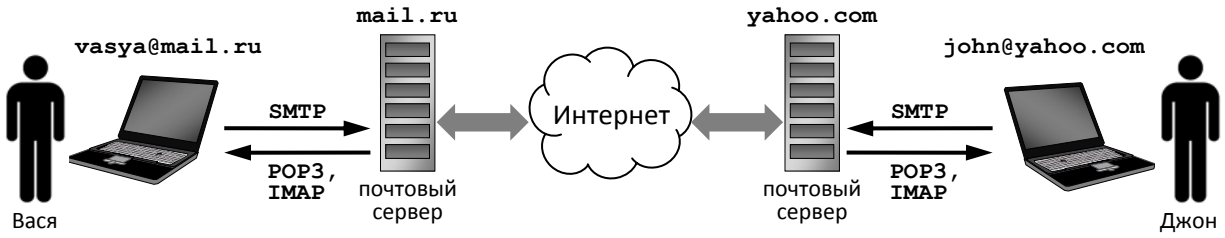
[5] Начальный список страниц обычно задают разработчики

### *Задачи*

1. Расположите запросы в порядке возрастания количества найденных страниц:
   **а) принтеры & сканеры & продажа**
   **б) принтеры | продажа**
   **в) принтеры | сканеры | продажа**
   **г) принтеры & продажа**
   (Ответ: а, г, б, в)
2. Расположите запросы к поисковому серверу в порядке возрастания количества найденных страниц:
   **а) Америка | путешественники | Колумб**
   **б) Америка | путешественники | Колумб | открытие**
   **в) Америка | Колумб**
   **г) Америка & путешественники & Колумб**
   (Ответ: г, в, а, б)
3. Расположите запросы к поисковому серверу в порядке возрастания количества найденных страниц:
   **а) семена | помидоры | огурцы**
   **б) семена | (огурцы & семена)**
   **в) семена & помидоры**
   **г) семена | огурцы**
   (Ответ: в, б, г, а)
4. Расположите запросы к поисковому серверу в порядке возрастания количества найденных страниц:
   **а) ананасы | (груши & лимоны)**
   **б) ананасы | груши**
   **в) (груши & лимоны) | (ананасы & мандарины)**
   **г) ананасы | лимоны | груши**
   (Ответ: в, а, б, г)

## 7.7. Электронная почта

Электронная почта – один из первых сервисов Интернета, но и сейчас она играет в сети огромную роль. Для того чтобы отправлять и принимать сообщения, пользователь должен зарегистрировать почтовый ящик на одном из почтовых серверов в Интернете. Многие из них бесплатны и имеют *веб-интерфейс*, то есть позволяют работать с почтой в браузере, не устанавливая на компьютер почтовые программы. Примеры таких серверов – *mail.ru* и *yahoo.com*.




Электронный адрес состоит из двух частей – названия почтового ящика и имени сервера; они отделяются символом @, который в России называют «собакой» (его официальное название – «коммерческое *at*»). Адрес **vasya@mail.ru** означает «почтовый ящик **vasya** на сервере **mail.ru**».

Для отправки сообщения компьютер пользователя (Васи) должен обменяться данными с почтовым сервером по протоколу SMTP (англ. *Simple Mail Transfer Protocol* – простой протокол передачи почты). Затем электронное письмо передается на сервер, где зарегистрирован почтовый ящик адресата (на рисунке – это сервер *yahoo.com*). Письмо сохраняется на сервере до тех пор, пока адресат (Джон) со своего компьютера не примет пришедшую ему почту, используя протокол POP3 (англ. *Post Office Protocol* – почтовый протокол) или протокол IMAP (англ. *Internet Message Access Protocol* – протокол доступа к сообщениям в Интернете).

Чтобы получить свои сообщения с сервера, Джону необходимо ввести пароль, однако для отправки сообщения (по протоколу SMTP) это не нужно. Поэтому можно послать сообщение с любого адреса (без ведома его владельца), и очень сложно разобраться, кто же в самом деле автор письма. Эта ситуация привела к появлению массовых рассылок рекламы с чужих адресов – *спаму*. Для борьбы со спамом многие сервера требуют ввести пароль (выполнить *авторизацию*) даже при отправке сообщений.

Сообщение электронной почты состоит из заголовка, текста письма и вложенных файлов.

**Заголовок** содержит служебную информацию, необходимую для пересылки. Вот пример информации в заголовке (приведены русские и английские обозначения):

| | |
|---|---|
| **Кому (To):** | **john@yahoo.com** |
| **От кого (From):** | **vasya@mail.ru** |
| **Ответить (Reply To):** | **vasya-home@mail.ru** |
| **Копия (CC):** | **boss@mail.ru** |
| **Скрытая копия (BCC):** | **john2@yahoo.com** |
| **Тема (Subject):** | **О покупке слона** |

Поле «Ответить» используется тогда, когда сообщение посылается с одного адреса (например, с рабочего), а отвечать нужно на другой (домашний). По всем адресам, указанным в поле «Копия», отправляются копии письма. Копии отправляются еще и по всем адресам, которые указаны в поле «Скрытая копия», но остальные получатели об это не узнают.

Считается дурным тоном не заполнять поле «Тема**»** осмысленным текстом. Во-первых, часто сообщения без темы удаляются сразу как спам. Во-вторых, многие люди получают десятки сообщений в день, и им удобно сразу сортировать их по темам, а потом уже читать. В-третьих, искать нужное сообщение среди десятков других тоже удобнее всего по полю «Тема».

**Основная часть** письма тоже строится по некоторым правилам, похожим на правила составления бумажных писем. Сначала идет приветствие, затем суть сообщения, в конце – фамилия и имя автора, а если это официальное письмо – его должность и сведения об организации. Например:

```
Здравствуй, Джон!

Нет ли у тебя желания купить слона?

С уважением, Василий Пупкин,
генеральный директор ООО «Рога и копыта»,
г. Роговск, ул. Копытная, 2
тел. +7 (1812) 111-22-33, факс +7 (1812) 111-22-34
http://rogakopyta.ru
```

Вместе с письмом можно отправить любые небольшие файлы (ограничения на максимально допустимый размер файла устанавливается администратором сервера). Многие почтовые серверы запрещают пересылку исполняемых файлов (с расширением **.exe**), потому что так могут распространяться вирусы и вредоносные программы.

### *Контрольные вопросы*

1. Что необходимо для отправки сообщений по электронной почте?
2. Из каких частей строится адрес электронной почты?
3. Что такое почтовый сервер?
4. Какие протоколы используются для работы с электронной почтой?
5. Что такое «веб-интерфейс»? Какие преимущества и недостатки он имеет в сравнении с использованием почтовых программ?
6. Объясните, как передается электронное сообщение от отправителя к получателю.
7. Почему на многих почтовых серверах нужно вводить пароль для отправки сообщений?
8. Что такое спам? Как вы думаете, почему спам – это плохо?
9. Какая информация включается в заголовок электронного письма?
10. Почему рекомендуется не оставлять поле «Тема» пустым?
11. Как обычно строится основная часть электронного письма?
12. Какие файлы можно отправлять вместе с электронным сообщением?
13. Почему многие почтовые сервера не разрешают пересылку исполняемых файлов?

# 7.8. Другие службы Интернета

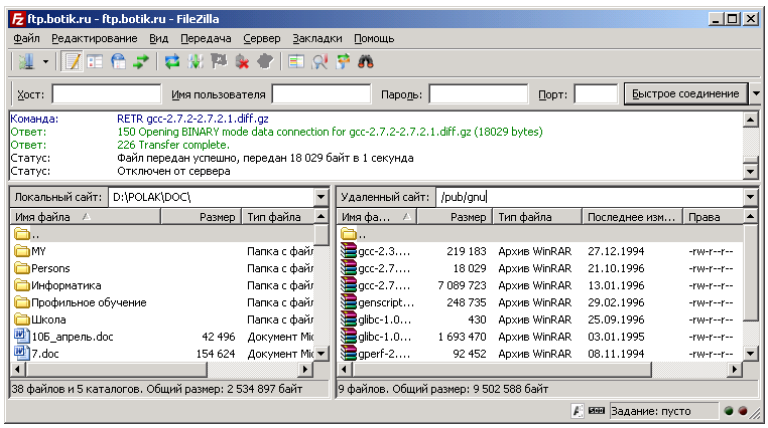
## 7.8.1. Обмен файлами (FTP)

Для обмена файлами используется протокол FTP, позволяющий скачивать файлы с сервера на компьютер пользователя (англ. *download*) и загружать файлы на сервер (англ. *upload*). Это возможно только тогда, когда на компьютере-сервере работает специальная программа – FTP-сервер, которая принимает запросы клиентов и отвечает на них по протоколу FTP.

FTP-сервера используются для распространения бесплатного программного обеспечения (свободные, бесплатные и условно-бесплатные программы, обновления антивирусных баз и т.п.) и загрузки файлов на веб-сайт.

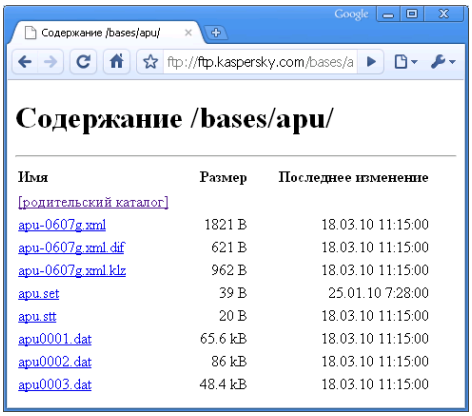
Для работы с FTP-сервером необходимо зарегистрироваться под своим кодовым именем (англ. *login*) и ввести пароль (англ. *password*). Однако многие FTP-серверы поддерживают анонимный вход: вместо имени нужно ввести «*anonymous*» (анонимный), а вместо пароля – любую последовательность символов.

Для работы с FTP-серверами используют программы, которые называются FTP-клиентами. Одна из самых популярных программ этого типа – свободный кроссплатформенный клиент *FileZilla* (*filezilla-project.org*). В одном окне программы показан каталог на компьютере пользователя, а в другом – каталог на FTP-сервере. Для работы с файлами можно использовать мышь.



Встроенные FTP-клиенты есть во многих других программах, например, в *Far Manager*.

Браузеры тоже умеют работать по протоколу FTP. Зайдя на FTP-сервер, вместо красочных веб-страниц вы увидите просто список файлов и каталогов. Файл начинает скачиваться, если щелкнуть на его имени, которое является ссылкой.




По умолчанию браузеры используют анонимный вход. Если нужно указать кодовое имя и пароль, их включают в адрес, который набирается в адресной строке:

**`ftp://user:asd@files.example.com`**

Здесь вход на FTP-сервер **`files.example.com`** выполняется под именем **`user`** с паролем **`asd`**.

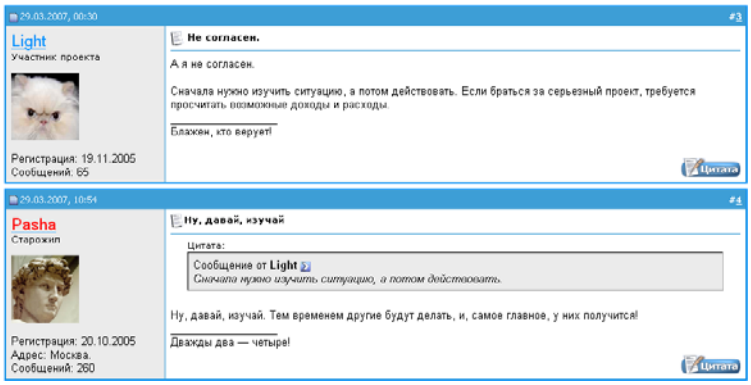
Чтобы понять, какая информация содержится в файлах, можно поискать файлы с именами **`index`**, **`dirinfo`**, **`readme`** – обычно они содержат описание файлов текущего каталога.

Если вы точно знаете имя файла, с помощью специальных поисковых систем (например, *www.filesearch.ru*), можно найти FTP-сервер, где он находится.

## 7.8.2. Форумы

*Форумы* – это специальные веб-сайты, предназначенные для общения посетителей в форме обмена сообщениями. Сообщения хранятся на серверах в Интернете и поэтому доступны всем участникам в любой момент.

Администратор создает несколько разделов форума, отличающихся по тематике. Пользователи создают в этих разделах *темы* для обсуждения (иногда тему называют «топик», от англ. *topic*, или «тред», от англ. *thread* – нить). Участники могут отвечать на любые сообщения в теме, комментировать их. Для изучения общественного мнения автор первого сообщения темы («топик-стартер», от англ. *topic starter* – тот, кто начал тему) может добавить к ней опрос (голосование).




Большинство форумов могут просматривать все желающие. Для того, чтобы отправить свое сообщение, обычно требуется регистрация. Могут быть и «закрытые» разделы, для доступа в которые кроме регистрации нужно специальное разрешения администратора.

При регистрации пользователь выбирает *ник* (от англ. *nickname* – псевдоним); на сервере создается *профиль* участника форума – страница, где он может оставить информацию о себе, загрузить *аватар* – свою фотографию или любую другую картинку, которая его может характеризовать. Иногда используют также слова «*аватарка*» или «*юзерпик*» (от англ. *user picture* – картинка пользователя), имеющие то же значение, что и аватар.

На большинстве форумов работает система личных сообщений – внутренняя электронная почта форума.

За порядком следят ответственные участники – администратор форума и назначенные им *модераторы*. Они могут изменять, перемещать и удалять любые сообщения, удалять профили пользователей и ограничивать им доступ – «банить», то есть запрещать отправлять сообщения (от англ. *ban* – запрет). Обычно в форумах наказывается

- отклонение от темы, «оффтопик» (от англ. *off-topic*, «вне темы»);
- оскорбление участников, нецензурная брань;
- реклама.

В некоторых форумах есть список часто задаваемых вопросов и ответов на них («ЧаВо»), в английском варианте – *FAQ = Frequently Asked* Question. Перед тем, как создать новую тему, нужно попытаться найти ответ на вопрос самостоятельно, прочитав этот документ (иначе вас могут «забанить»).

Производители аппаратуры и программного обеспечения часто создают на своих сайтах форумы, где их представители помогают пользователям решать возникающие вопросы.

Раньше роль форумов выполняли так называемые *группы новостей* (англ. *newsgroup*), для работы с которыми использовались специальные клиентские программы, работающие по протоколу NNTP (англ. *Network News Transfer Protocol* – сетевой протокол передачи новостей). Иногда такие клиенты встраивают в почтовые программы, например, в *Mozilla Thunderbird*. Для участия в современных группах новостей достаточно использовать любой браузер (см., например, *groups.google.com*).

## 7.8.3. Общение в реальном времени

Выражение «в реальном времени» (англ. *on-line* – «на линии») означает, что все участники в момент обмена информацией находятся за компьютерами.

Простейший вариант общения – обмен текстовыми сообщениями. *Чаты* (англ. *chat* – болтовня) позволяют «разговаривать» группе людей, которые как бы находятся в одной комнате. Для личного общения используют программы для обмена мгновенными сообщениями (*мессенджеры*), например, *ICQ* (*www.icq.com*), *Mail.ru Агент* (*www.mail.ru*), *Kopete* (для *Linux*), *iChat* (для компьютеров фирмы *Apple*). Возможно, в недалеком будущем их заменит бесплатная система мгновенного обмена сообщениями *Jabber* (*www.jabber.org*). С помощью мессенджеров можно передавать файлы напрямую с компьютера на компьютер или через сервер.

Особой популярностью пользуется бесплатная программа *Skype* (*skype.com*), которая позволяет

- организовывать личные и групповые чаты;
- передавать сообщения и файлы с компьютера на компьютер;
- устанавливать голосовую и видеосвязь (для этого необходимы наушники, микрофон и веб-камера);
- звонить на стационарные и мобильные телефоны;
- принимать звонки с телефонов на специальный номер;
- отправлять SMS-сообщения;
- организовывать конференции (совещания через Интернет).

Часть из этих функций (например, звонки на телефоны и отправка SMS) платные.

*Skype* использует технологию *VoIP* (англ. *Voice over Internet Protocol* – передача голоса через интернет-протокол), все передаваемые данные шифруются. Звонок через *Skype* в другой город или в другую страну обходится значительно дешевле, чем обычная телефонная связь.

*Skype* – это кроссплатформенная программа, существуют ее версии для операционных систем *Windows, Linux, Mac OS X*, а также для смартфонов. Центральный сервер используется только для установки связи, а дальше компьютеры обмениваются данными напрямую.

Протокол *VoIP*, по которому работает *Skype*, закрыт, так же как и исходный код программы. Все данные многократно шифруются, и их не могут анализировать антивирусы. Поэтому специалисты по компьютерной безопасности предупреждают о возможности использования *Skype* для распространения вредоносных программ и шпионажа.

## 7.8.4. Информационные системы

Информационные системы в Интернете позволяют быстро находить нужную в данный момент информацию. Информационная система состоит из базы данных и программного обеспечения для поиска информации, размещенного на сайте.

На многих сайтах доступны прогнозы погоды на разные сроки (*pogoda.ru, gismeteo.ru*, *weather.yandex.ru*). С помощью сервиса *rasp.yandex.ru* можно узнать расписание электричек, поездов дальнего следования и самолетов по всей России. На сайтах аэропортов постоянно обновляется табло фактического прибытия и отправления самолетов с учетом задержки рейсов.

Заказывать билеты на поезда и самолеты удобно на специальных сайтах, которые связаны с системой продажи билетов железной дороги и авиакомпаний. Здесь же можно получить полную информацию о расписании, возможных вариантах проезда (перелета), стоимости и наличии билетов. Оплатить билеты можно прямо на сайте с помощью банковской карты или платежных систем, а также через терминалы приема платежей.

Часто используются так называемые *электронные билеты* (англ. *e-ticket*) на поезда и самолеты. Электронный билет – это информация о заказе, сделанном через веб-сайт, которая внесена в базу данных. По номеру заказа в кассе вокзала можно получить бумажный билет, а в аэропортах для регистрации по электронному билету достаточно просто предъявить паспорт.

Сервисы веб-картографии *Google Maps* (*maps.google.ru*), *Яндекс.Карты* (*maps.yandex.ru*), *Карты@Mail.ru* (*maps.mail.ru*) позволяют найти на карте любой адрес, проложить маршрут и оценить его длину. На этих сайтах доступны не только карты (хранящиеся в векторном формате), но также снимки многих районов из космоса.

### ? *Контрольные вопросы*

1. Зачем используются FTP-серверы?
2. Как и в каком случае можно зайти на FTP-сервер, не имея своей учетной записи?
3. Что такое FTP-клиент?
4. Как работать с FTP-серверами с помощью браузера?
5. Как узнать, что находится в файлах, не скачивая их? Всегда ли это возможно?
6. Как найти сервер, откуда можно загрузить интересующий вас файл?
7. Что такое форумы?
8. Что такое «топик», «ник», «аватар», «модератор», «бан», «оффтопик»?
9. Как поддерживается порядок в форуме?
10. Что такое *ЧаВо* (FAQ)?
11. Как вы думаете, зачем производители аппаратуры организуют форумы на своих сайтах?
12. Что такое «общение в реальном времени»? Относятся ли форумы к «онлайн-общению»?
13. Расскажите о достоинствах и недостатках программы *Skype*.
14. Какие информационные системы существуют в Интернете?
15. Что такое электронный билет? Чем он удобнее бумажного?

# 7.9. Электронная коммерция

## 7.9.1. Что такое электронная коммерция?

В начале своего развития Интернет был полностью некоммерческим – в его развитии участвовали инженеры, программисты, ученые и военные. Однако к началу 1990-х годов стало ясно, что глобальная сеть может быть источником огромной прибыли.

**Электронная коммерция** (англ. *e-commerce*) – это покупка и продажа товаров и услуг с помощью электронных систем, например, через Интернет.

Развитие электронной коммерции в Интернете началось в 1994 году, когда на сайте американской сети ресторанов *Pizza Hut* появилась возможность заказать пиццу с доставкой на дом. В том же году открылись сайты некоторых банков в Интернете, и пользователи получили возможность управления своими счетами через сеть. В 1995 году был создан первый книжный интернет-магазин *Amazon* (*www.amazon.com*), который и сейчас остается самым крупным в мире.

Электронная коммерция включает в себя

- исследование рынка;
- обмен данными и документами в электронном виде;
- денежные операции в электронной форме;
- продажу товаров, услуг и информации;
- поддержку покупателей после продажи.

Сейчас многие покупатели начинают поиски нужного товара в Интернете, а не в обычных магазинах. Чтобы привлечь внимание клиентов, фирмы размещают подробную информацию о товарах на своих сайтах, делают рассылки по электронной почте, создают сообщества в социальных сетях, организуют дискуссии на форумах. Посетителям сайтов предлагается оставить отзыв о товарах на специальной веб-странице, чтобы остальные смогли его прочитать.

Через Интернет удобно приобретать книги и электронику, авиа- и железнодорожные билеты, оплачивать услуги операторов сотовой связи.

Для пользователя всегда важно знать, что после покупки он не окажется один на один со своими проблемами и сможет получить консультацию. Поэтому поддержка покупателей – тоже важный элемент электронной коммерции. На сайтах производителей оборудования (винчестеров, принтеров, сканеров и т.п.) всегда можно найти всю документацию по настройке и обслуживанию устройств и бесплатно скачать самые новые драйверы. На форумах фирм-изготовителей пользователи получают консультацию службы технической поддержки и делятся друг с другом решениями возникших проблем.

Бизнес в Интернете предоставляет дополнительные возможности **для компаний**:

- расширение *сферы влияния*: фирмы любого размера могут принимать заказы со всего мира;
- увеличение *конкурентоспособности*: быстрая реакция на претензии клиентов и изменение ситуации на рынке;
- *индивидуальный подход* (система собирает информацию о клиенте и пытается предложить именно те товары, которые он чаще покупает);
- *уменьшение затрат*: не нужно арендовать помещение для магазина и нанимать много сотрудников.

**Покупатели** тоже получают серьезные преимущества:

- *выбор* товаров и услуг из большого количества вариантов;
- легко *сравнить* разные предложения;
- можно узнать *отзывы* других пользователей;

- можно заказывать и оплачивать товары *в удобное время*;
- *цены* на товары и услуги обычно ниже, чем в обычных магазинах.

## 7.9.2. Интернет-магазины

Все больше людей используют **интернет-магазины** – веб-сайты, которые рекламируют товары или услуги, принимают заказы на покупку, предлагают варианты оплаты и получения заказа. Выбрав товары в интернет-магазине, пользователь может «положить их в корзину», то есть, включить в список для приобретения. Когда состав покупки полностью определен, оформляется заказ: покупатель указывает свои данные, способ оплаты и доставки. На некоторых сайтах возможен заказ товара по телефону.

**Оплата** выполняется разными способами:

- через банковскую карту, пригодную для оплаты в Интернете (*Visa, MasterCard*);
- банковским переводом (например, через *Сбербанк*);
- почтовым переводом;
- с помощью электронных платежных систем (электронными деньгами);
- наличными при получении товара;
- через отправку платного SMS-сообщения (для небольших покупок).

Для «вещественных» товаров существует три способа **доставки**:

- курьерская доставка по указанному адресу;
- почтовая доставка;
- «самовывоз» с пунктов выдачи товара (в некоторых городах).

Товары в электронной форме (программы, коды доступа, тексты, статьи, фотографии и т.п.) обычно высылаются по электронной почте, или покупателю предоставляется персональная ссылка на нужный файл на сервере фирмы-продавца.

Для управления интернет-магазином используют специальное программное обеспечение, например, кроссплатформенные системы *1С-Битрикс* (*www.1c-bitrix.ru*) и *osCommerce* (*www.oscommerce.com*).

Еще один вид электронной коммерции – **интернет-аукционы** («онлайновые аукционы»), в которых фирма-организатор – это просто посредник между продавцом и покупателем. Любой желающий может делать ставки, используя веб-сайт аукциона. Когда интернет-аукцион завершен, покупатель, сделавший наибольшую ставку, должен перевести деньги продавцу, а продавец обязан выслать товар покупателю по почте. Крупнейший интернет-аукцион – *eBay* (*www.ebay.com*).

## 7.9.3. Электронные платежные системы

Электронная торговля не была бы столь удобной, если бы не было возможности оплачивать покупку прямо в Интернете, не выходя из дома. Для этого должны были появиться *электронные деньги*, которые можно свободно купить и продать. Системы расчетов электронными деньгами называются *электронными платежными системами*. Среди российских платежных систем наиболее известны *WebMoney* (*www.webmoney.ru*) и *Яндекс.Деньги* (*money.yandex.ru*).

В платежной системе можно завести электронный кошелек, который пополняется с помощью специальных карт оплаты или через терминалы. Из такого кошелька можно оплачивать коммунальные услуги, сотовую связь и Интернет, покупать авиа- и железнодорожные билеты, товары в интернет-магазинах. По условиям пользовательского соглашения кошелек системы *Яндекс.Деньги* нельзя использовать для коммерческой деятельности (например, для получения оплаты за выполненную работу), иначе его может заблокировать служба безопасности.

Пользователи могут переводить электронные деньги не только на счета интернет-магазинов, но и на кошельки других пользователей. Чтобы убедиться в том, что вы не ошиблись при вводе номера кошелька и переводите деньги именно тому, кому нужно, применяют перевод с *кодом протекции*. Код протекции – это некоторое число, которое должен ввести получатель для получения перевода на свой счет. Этот код можно сообщить по электронной почте или по телефону. Если получатель вводит неверный код несколько раз подряд или истекает срок действия перевода, средства возвращаются на кошелек отправителя.

При необходимости можно вывести средства из электронного кошелька, то есть получить их в виде наличных денег в банке (за вычетом небольшой комиссии).

С электронными кошельками можно работать в браузере (через веб-интерфейс сайта) или с помощью специальной программы, которая устанавливается на компьютер пользователя.

### ? *Контрольные вопросы*

1. Что такое электронная коммерция?
2. Какие направления включает электронная коммерция?
3. Какие преимущества предоставляет электронная торговля для продавцов и покупателей?
4. Что такое интернет-магазин?
5. Как можно оплатить покупку в интернет-магазине? Как доставляются покупки?
6. Что такое интернет-аукцион?
7. Что такое электронные деньги? Чем они, на ваш взгляд, отличаются от «обычных»?
8. Зачем используется код протекции при переводе электронных денег?

## 7.10. Интернет и право

Как только в сети появилась купля-продажа товаров и услуг, возникла необходимость регулировать эти отношения с помощью закона, защищать интересы пользователей и предотвращать мошенничество.

Интернет – это не организация, он не принадлежит ни одной стране, развивается во многом стихийно и не может быть юридическим лицом. В связи этим возникает множество проблем, с которыми юристы раньше не сталкивались. Например, не вполне ясно

- несет ли провайдер ответственность за действия пользователей, которым он предоставляет доступ в Интернет (в том числе за нарушения авторских прав)?
- можно ли признавать доказательствами цифровые документы (сообщения электронной почты, рисунки, звукозаписи, видео)?
- как доказать условия заключенной сделки, если фирма может в любой момент изменить условия договора на сайте?
- какую ответственность несут платежные системы перед государством и пользователями?

Соблюдение *авторских прав*, то есть, прав автора на результаты своего интеллектуального труда – один из самых актуальных правовых вопросов Интернета. Практически вся информация на сайтах защищается авторским правом: программное обеспечение, тексты, рисунки и фотографии, музыка и видеофильмы. Часто на веб-страницах публикуются условия использования материала (англ. *terms of use*), где указывается, можно ли сохранять и копировать материал, вставлять его в другие документы, распечатывать. Если такой информации нет, следует получить разрешение на использование материала, отправив сообщение веб-мастеру по электронной почте.

На своем веб-сайте **можно без разрешения**:

- размещать гиперссылки на другие сайты;
- использовать бесплатную графику.

Без разрешения **нельзя**:

- копировать содержание других сайтов;
- объединять информацию из разных источников для создания «собственного» документа;
- изменять чужой текст или изображение;
- размещать любые изображения с других сайтов, о которых явно не написано, что они бесплатные.

В Гражданском кодексе Российской Федерации (ГК РФ) определяется, что можно без согласия автора использовать его произведения в научных, учебных или культурных целях (статья 1274). При этом обязательно указать имя автора и источник (книгу, статью, сайт). Например, разрешается

- *цитировать* произведения (для того, чтобы подтвердить или опровергнуть какую-то мысль автора) в объеме, оправданном целью цитирования;
- использовать произведения и их отрывки в *учебных* материалах в объеме, оправданном поставленной целью;
- использовать произведения для создания пародии или карикатуры.

*Использование* чужого произведения (например, текста, музыки или видеозаписи) как составной части в своих работах является нарушением авторского права независимо от объемов (например, нельзя включить в свой фильм даже 1 секунду из другого фильма или звукозаписи). При этом не имеет значения, что вы не получили от этого коммерческой выгоды (это влияет только на размер штрафа).

Переработка чужого материала (например, наложение текста, графики, звука, монтаж видео) – это серьезное нарушение права автора на неприкосновенность произведения (статьи 1255 и 1266 ГК РФ), за это в законе предусмотрены значительные штрафы.

Незаконный доступ к информации – это тоже уголовное преступление, за которое в Уголовном кодексе РФ предусмотрен штраф или лишение свободы на срок до двух лет (статья 272). Если ваш сайт (или учетную запись на сайте) взломали, или вы стали жертвой интернет-мошенничества, нужно обращаться в отдел по борьбе с компьютерными преступлениями (отдел «К») милиции.

### ? *Контрольные вопросы*

1. Какие юридические проблемы возникают в связи с куплей-продажей услуг в Интернете?
2. В каком случае можно размещать на своем веб-сайте чужой материал?
3. В каком случае можно бесплатно использовать произведения без согласия автора? Какие ограничения нужно при этом соблюдать?
4. Расскажите о наиболее распространенных нарушениях авторских прав в Интернете.
5. Является ли взлом персональной странички в социальной сети преступлением? Почему?
6. Что можно сделать, если вас обманули мошенники в Интернете?

## 7.11. Нетикет

На ранних этапах развития, когда к Интернету были подключены только научные центры и университеты, общение основывалось на взаимном уважении и доверии пользователей. В результате сложились неофициальные правилами общения, которые называются сетевым этикетом или *нетикетом* (фр. *netiquette*).

Несмотря на то, что при общении в Интернете вы, как правило, не видите собеседника, нужно вести себя так, как будто вы говорите с человеком лично: не пишите то, что вы не смогли бы сказать ему в лицо; не оскорбляйте собеседника, не ругайтесь. Перед тем, как послать сообщение, прочтите его внимательно еще раз и поставьте себя на место получателя.

Передавая информацию по открытым каналам, помните, что копии всех ваших сообщений могут храниться, например, у провайдера. Не посылайте информацию, доступ к которой ограничен. Уважайте авторские права, не используйте чужой материал без разрешения. Уважайте тайну переписки: если вы хотите опубликовать личное сообщение, нужно спросить разрешения у автора.

В сообщениях **электронной почты** и **форумов**

- цените время других людей, пишите кратко и точно;
- не пишите всеми заглавными буквами (это воспринимается как крик или визг);
- не используйте слэнг, пишите грамотно, не пропускайте пробелы и знаки препинания;
- для передачи тона письма используйте *смайлики* (англ. *smiley*) – значки для обозначения эмоций, например,

  **:-)** или **:)** – улыбка;

  :-( или **:(** – несчастное лицо, сожаление или разочарование;

  **;-)** или **;)** – подмигивающее лицо, слова не следует понимать слишком серьезно;
- цитируйте те высказывания, на которые отвечаете (собеседник может забыть содержание предыдущего письма);
- не распространяйте *спам* – нежелательную рекламу.

Посылая сообщения по **электронной почте,**

- обязательно пишите тему сообщения, отражающую содержание письма;
- ставьте подпись (имя и фамилию) в конце письма;
- не посылайте большие файлы без согласия получателя.

Участвуя в **форумах,**

- сначала почитайте список часто задаваемых вопросов и прошлые сообщения, возможно, вы найдете там ответ на свой вопрос;
- отправляя сообщение, осознайте, что многие увидят ваш текст;
- не отклоняйтесь от темы обсуждения;
- не участвуйте во *флейме* (от англ. *flame* – огонь, пламя) – так называется «спор ради спора», словесная война; обычно за флейм наказывают модераторы форума;
- не разжигайте *холивары* (от англ. *holy war* – «священная война») – так называется спор о двух идеях, каждая из которых имеет своих сторонников (например, что лучше, «*Windows* или *Linux*», «*Паскаль* или *Си*» и т.п.).

Общаясь в **чатах**,

- не перебивайте собеседника;
- не обижайтесь, если незнакомые люди не хотят с вами разговаривать;
- не пытайтесь выведывать личную информацию;
- уважайте анонимность, не разглашайте реальное имя другого участника без разрешения, если вы его знаете;
- будьте снисходительны к ошибкам других;
- не обижайтесь, если собеседник неожиданно покинул чат.

## ? *Контрольные вопросы*

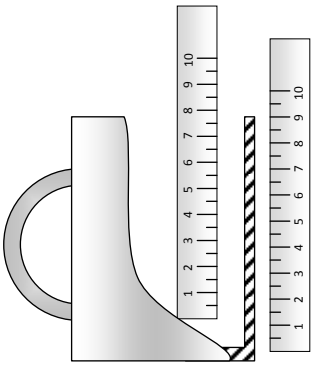
1. Что такое нетикет?
2. Можно ли опубликовать на форуме личное сообщение?
3. Какие правила рекомендуется соблюдать в сообщениях электронной почты?
4. Как нужно вести себя в форумах и чатах?

5. Как в электронном письме можно передать свое настроение?
6. Почему при пересылке больших файлов нужно спросить согласия получателя?
7. Что такое «флейм»? «холивар»?

# Глава 9. Решение вычислительных задач на компьютере

## 9.1. Точность вычислений

«Недостатки математического образования с наибольшей отчетливостью проявляются в чрезмерной точности численных расчетов» – писал немецкий математик Карл Фридрих Гаусс.

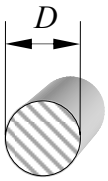


Все практические расчеты выполняются неточно, с некоторой ошибкой. В первую очередь, это связано с тем, что неточно известны исходные данные. Пусть нужно найти толщину дна кружки, используя только линейку с делениями через 0,5 см. Линейкой можно измерить высоту кружки снаружи и глубину внутри, причем при этом точность измерений будет (в лучшем случае) около 0,1 см. Например, если измеренная высота кружки оказалась примерно 9,3 см, в самом деле она может быть от 9,2 до 9,4 см. Если измеренная глубина равна 8,9 см, фактическая может быть от 8,8 до 9,0 см. Используя данные измерений, можно найти толщину дна как разность

$$9{,}3 - 8{,}9 = 0{,}4 \text{ см}.$$

Это не означает, что толщина дна действительно такая. Действительно, с учетом ошибок измерений она может быть равна как 9,2 – 9,0 = 0,2 см, так и 9,4 – 8,8 = 0,6 см. Таким образом, реальная толщина может быть от 0,2 до 0,6 см (разница в 3 раза!), и в полученном ответе (0,4 см) нет ни одной верной значащей цифры! Обычно в этом случае пишут ответ в виде 0,4±0,2 см.

> Ошибка результата вычислений не может быть меньше, чем ошибка в исходных данных.




Рассмотрим другой пример: нужно вычислить площадь сечения цилиндра, диаметр которого ($D$ = 1,2 см) известен с точностью 0,1 см. По известной формуле площади круга получаем (например, на калькуляторе):

$$S = \frac{\pi \cdot D^2}{4} = 1{,}130973355292325565846551617 9806...$$

Значит ли это, что мы нашли площадь с такой точностью? Конечно, нет. Вспомним, что диаметр был измерен с точностью 0,1 см, то есть он мог быть, в самом деле, равен как 1,1 см, так и 1,3 см. В этих «крайних» случаях получаем площадь

$$S_{\min} = \frac{\pi \cdot 1{,}1^2}{4} = 0{,}950... \quad \text{и} \quad S_{\max} = \frac{\pi \cdot 1{,}3^2}{4} = 1{,}327...$$

Таким образом, следует записать ответ в виде $S \approx 1{,}1 \pm 0{,}2$ см$^2$.

Теперь вернемся к расчетам с помощью компьютера. Как известно, данные записываются в память в двоичном коде ограниченной длины, при этом практически все вещественные числа хранятся с некоторой ошибкой. При выполнении вычислений ошибки накапливаются, поэтому при сложных расчетах может получиться совершенно неверный ответ. Классический пример – катастрофическая потеря точности при решении на компьютере больших систем линейных уравнений методом Гаусса. Оказывается, метод Гаусса *вычислительно неустойчив*: это значит, что малые ошибки в исходных данных могут привести к огромным ошибкам в решении. Например, с точки зрения точности очень плохо, если ответ – это небольшое (по модулю) число, которое вычисляется как разность двух больших чисел (вспомните пример с кружкой!).

Подводя итог, можно выделить несколько источников ошибок при компьютерных вычислениях:

- неточность исходных данных;

- ошибки при записи вещественных чисел в двоичном коде конечной длины;
- ошибки при приближенном вычислении некоторых стандартных функций (например, $\sin(x)$ или $\cos(x)$);
- накопление ошибок при арифметических действиях с неточными данными;
- собственная погрешность используемого метода (для приближенных методов, рассматриваемых в следующем параграфе).

Проблемы, возникающие при вычислениях с конечной точностью, изучает *вычислительная математика*, задача которой – разработать *вычислительно устойчивые* методы решения задач, при которых небольшие ошибки исходных данных мало влияют на результат. Иногда этого удается добиться просто изменением порядка действий или преобразованием формул.

### *Контрольные вопросы*

1. Как вы понимаете приведенное высказывание К.Ф. Гаусса?
2. Что такое вычислительно неустойчивый метод?
3. В каких случаях проявляется вычислительная неустойчивость метода Гаусса?
4. Перечислите источники ошибок при компьютерных вычислениях.
5. Какие задачи изучает вычислительная математика?

### *Задачи*

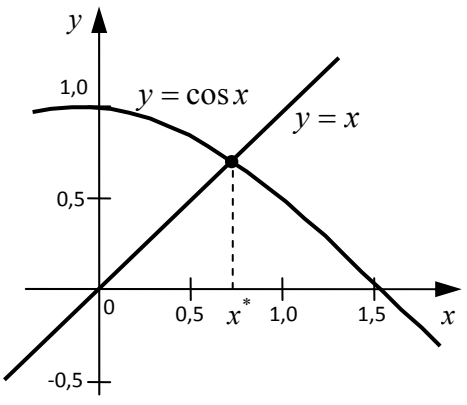
1. Радиус шарика $R$ измерили с точностью 0,1 см и получили 1,2 см. Затем рассчитали его объем по формуле $V = \dfrac{4 \cdot \pi \cdot R^3}{3} = 7{,}23822947387088...$ Запишите ответ в этой задаче, оставив нужно количество значащих цифр. (Ответ: $V \approx 7 \pm 2$ см$^3$)
2. С помощью рулетки размеры бруса (220 см × 11 см × 10 см) измерили с точностью 1 см. Определите его объем. Запишите ответ с учетом точности полученного результата. (Ответ: $V \approx 24000 \pm 5000$ см$^3$)

# 9.2. Решение уравнений

## 9.2.1. Приближенные методы

На уроках математики вас учили искать решение уравнения в виде формулы, выражающей неизвестную величину через известные. Например, решение уравнения $ax + b = 1$ при $a \neq 0$ можно записать в виде $x = (1-b)/a$. Такое решение называется *аналитическим*, оно может быть использовано для теоретического исследования (изучения влияния исходных данных на результат).

Однако не все уравнения можно (на современном уровне развития математики) решить аналитически. Иногда решение есть, но очень сложное. Например, уравнение $x = \cos x$ так просто не решается. В этом случае приходится использовать другие методы решения, например, графический: построить по точкам графики функций, стоящих в левой и правой частях равенства, и посмотреть, где они пересекаются. Затем решение можно уточнять, уменьшая шаг при построении графика до получения требуемой точности.



Если нужна высокая точность, графический метод требует очень большого объема вычислений, который имеет смысл поручить компьютеру. Однако, нужно как-то учесть, что компьютер не способен «посмотреть» на график, и для уточнения решения может использовать только числовые данные. Компьютерный алгоритм решения уравнения может выглядеть примерно так:

1) выбрать интервал $[a_0, b_0]$ для поиска решения (обычно предполагается, что на этом интервале решение есть, и притом только одно);
2) с помощью некоторого алгоритма уточнить решение, перейдя к меньшему интервалу $[a, b]$;
3) повторять шаг 2, пока ширина интервала, в который мы «загнали» решение, не станет достаточно мала.

Здесь не совсем ясно, что значит «пока ширина интервала не станет достаточно мала». Обычно задается нужная точность $\varepsilon$: это означает, что отклонение полученного решения от истинного $x^*$ не должно быть больше $\varepsilon$. Если корень уравнения находится в интервале $[a, b]$, то в качестве решения лучше всего взять его середину $(a+b)/2$. В этом случае ошибка будет минимальной: не больше половины ширины интервала. Поэтому цикл нужно остановить, когда ширина интервала станет меньше, чем $2\varepsilon$.

Часто используется другой вариант, когда интервал не нужен, а требуется знать только одну точку вблизи решения:

1) выбрать *начальное приближение* $x_0$ около решения;
2) с помощью некоторого алгоритма перейти к следующему приближению $x$, которое находится ближе к точному решению $x^*$;
3) повторять шаг 2, пока на очередном шаге решение не изменится на величину, меньшую, чем допустимая ошибка $\varepsilon$.

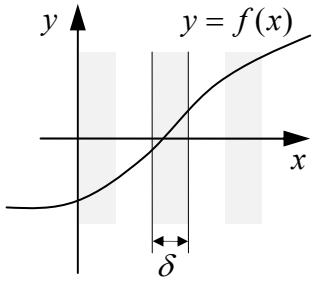
Подобные методы решения уравнений называются *приближенными*. Их суть в том, что решение последовательно уточняется до тех пор, пока не будет найдено с требуемой точностью. Поскольку при каждом уточнении выполняются одинаковые действия, приближенные методы называют *итерационными* (от лат. *iteration* – повторение).

Приближенные методы имеют ряд **недостатков**:

- получается *приближенное* решение, а не точное; это значит, что нельзя написать $x^* = 1{,}2345$, нужно использовать знак приближенного равенства: $x^* \approx 1{,}2345$;
- мы получаем не формулу, а число, по которому невозможно оценить, как меняется решение при изменении исходных данных;
- объем вычислений может быть слишком велик, часто это не позволяет использовать приближенные методы в системах реального времени;
- не всегда можно оценить ошибку (отклонение найденного решения от точного).

Однако в некоторых практических случаях приближенные методы более полезны, чем аналитические. Во-первых, часто получение аналитического решения невозможно или требует значительных усилий, тогда как приближенные методы позволяют достаточно быстро решить задачу с заданной точностью. Во-вторых, при компьютерных расчетах (с конечной точностью) вычисления по «точным» аналитическим формулам часто могут давать очень неточный результат из-за вычислительной неустойчивости метода. Нередко для таких задач (например, для решения систем линейных уравнений) известны приближенные методы, которые дают значительно более точное решение.

## 9.2.2. Метод перебора




Как принято в вычислительной математике, далее мы будем рассматривать уравнение общего вида $f(x)=0$, к которому можно привести любое заданное уравнение. Например, для уравнения $x=\cos x$ получаем $f(x)=x-\cos x$.

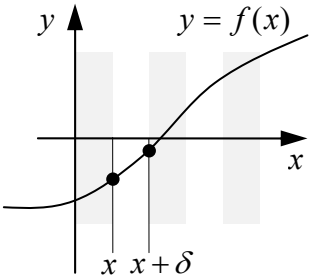
Предположим, что нужно найти решение уравнения $f(x)=0$ с точностью $\varepsilon$, причем известно (например, видно на графике), что решение находится справа от точки $x=a$. В этом случае можно разбить всю область, где может быть решение, на узкие полоски шириной $\delta=2\varepsilon$, и выбрать такую полоску, где график функции пересекает ось OX.

Для того, чтобы поручить решение этой задачи компьютеру, нужно ответить на два вопроса:

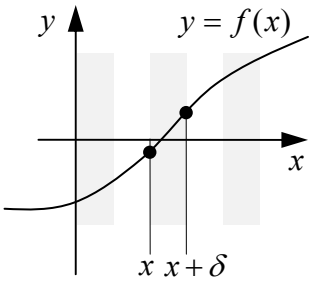
1) как с помощью математических операций определить, что в полосе $[x,x+\delta]$ есть решение?
2) что считать решением уравнения, когда такая полоса определена?

Проще всего ответить на второй вопрос: лучше всего взять в качестве решения середину полосы, то есть, точку $x_*=x+\varepsilon$ (в этом случае ошибка гарантированно не будет больше, чем $\varepsilon$).

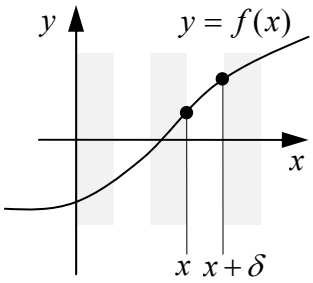
Для того, чтобы определить, есть ли решение на интервале $[x,x+\delta]$, сравним значения функции на концах этого интервала. Рассмотрим три случая, показанные на рисунках:




$f(x)<0,\ f(x+\delta)<0$




$f(x)<0,\ f(x+\delta)>0$




$f(x)>0,\ f(x+\delta)>0$

Несложно сообразить, что если график пересекает ось OX, на концах интервала функция имеет разные знаки, то есть $f(x)\cdot f(x+\delta)<0$. При этом важно, чтобы график не имел разрывов.

> Если непрерывная функция имеет разные знаки на концах интервала $[x_1,x_2]$, то в некоторой точке внутри этого интервала она равна нулю.

Таким образом, нужно найти интервал шириной $2\varepsilon$, на концах которого функция имеет разные знаки, и взять в качестве решения его середину. Решение этой задачи на алгоритмическом языке и языке Паскаль может выглядеть, например, так:

```
алг Перебор
нач
  вещ eps, x, delta;
  eps:= 0.001;
  x:= 0;
  delta:= 2*eps;
  нц пока f(x)*f(x+delta) > 0
    x:= x + delta;
  кц
  вывод 'Решение: ', x+eps;
кон
```

```
const eps=0.001;
var x, delta: real;
function f(x: real):real;
begin
  f := x - cos(x);
end;
begin
  x:= 0;
  delta:= 2*eps;
  while f(x)*f(x+delta) > 0 do
    x:= x + delta;
```

```
алг вещ f( вещ x )
нач
  знач:= x - cos(x);
кон
```

```
  writeln('Решение: ', (x+eps):6:3);
end.
```

Здесь заданная точность $\varepsilon$ хранится в виде константы **eps**, а вычисление функции $f(x)$ оформлено в виде подпрограммы-функции **f**. Поиск решения здесь начинается с нуля (в других задачах начальное значение может быть другое). Цикл останавливается, когда для очередного интервала получаем $f(x)\cdot f(x+\delta)<0$.

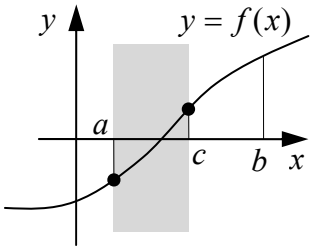
Обратите внимание, что в этом простейшем варианте программа зациклится, если справа от нуля решения нет. Подумайте, как изменить программу так, чтобы зацикливания не было.

Главный недостаток метода перебора – большое количество операций. Например, для решения уравнения с точностью 0,001 может понадобиться несколько тысяч вычислений значения функции $f(x)$. Поэтому сначала можно использовать перебор с достаточно большим шагом, чтобы *отделить корни*, то есть найти интервалы, в каждом из которых есть только один корень. После этого выполняется *уточнение корней* – перебор внутри каждого из таких интервалов с шагом $\delta = 2\varepsilon$, достаточным для определения решения с заданной точностью.

## 9.2.3. Метод деления отрезка пополам

Есть старая задача-шутка: «как поймать льва в Африке?» Предлагается перегородить забором всю Африку, разбив ее на две равные части, и ждать, где появится лев. Затем часть, в которой есть лев, разделить забором на две равные области и т.д. В конце концов, лев окажется в маленькой клетке. В самом деле, эта шутка иллюстрирует метод *деления отрезка пополам* (или метод *бисекции*), с помощью которого можно найти решение уравнения на некотором интервале $[a,b]$:

1) найти середину интервала $c=\dfrac{a+b}{2}$;
2) если на отрезке $[a,c]$ есть решение, присвоить **b:=c**, иначе присвоить **a:=c**;
3) повторять шаги 1-2 до тех пор, пока $b-a>2\varepsilon$.




В п. 2 нам нужно ответить на вопрос, если решение на интервале $[a,c]$. Мы уже имеем это делать: решение есть, если $f(a)\cdot f(c)<0$. Таким образом, цикл, уточняющий решение, запишется в виде:

```
delta:= 2*eps;
нц пока b - a > delta
  c:= (a + b) / 2;
  если f(a)*f(c) < 0 то
      b:= c;
    иначе
      a:= c;
  все
кц
вывод 'Решение: ', (a+b)/2;
```

```
delta:= 2*eps;
while b - a > delta do begin
  c:= (a + b) / 2;
  if f(a)*f(c) < 0 then
       b:= c
  else a:= c;
end;
writeln('Решение: ', (a+b)/2:6:3);
```

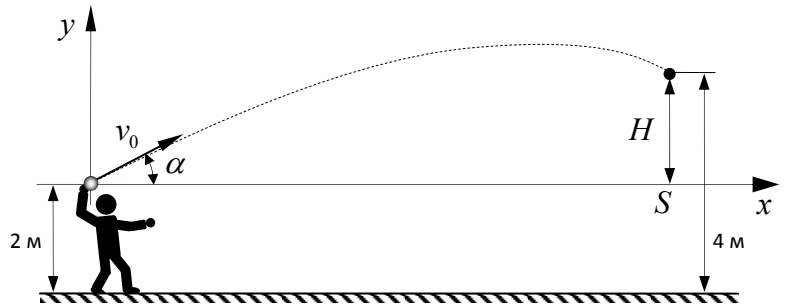
На каждом шаге этого цикла ширина интервала уменьшается в 2 раза, за $n$ шагов она уменьшится в $2^n$ раз. Таким образом, при выборе единичного начального интервала для получения решения с точностью 0,001 достаточно 10 шагов цикла.

Метод деления отрезка пополам очень прост и надежен, позволяет найти решение с заданной точностью (в пределах точности машинных вычислений). Однако, для его применения нужно заранее *отделить корни уравнения*, то есть найти интервалы, каждый из которых содержит только один корень. Для отделения корней можно использовать построенный график функции или метод перебора с большим шагом. Таким образом, решение уравнения проводится в два этапа – отделение корней и уточнение корней.

К сожалению, метод деления отрезка пополам неприменим для решения систем уравнений с несколькими неизвестными.

## 9.2.4. Пример: полет мяча

Для иллюстрации рассмотрим такую задачу: Вася Пупкин бросает мяч со скоростью 12 м/с. Под каким углом к горизонту ему нужно бросить мяч, чтобы попасть в мишень на высоте 4 м на расстоянии 10 м от Васи? В момент броска мяч находится на высоте 2 м.




Примем за начало координат точку, откуда вылетает мяч. Обозначим через $v_0$ начальную скорость мяча, через $H$ – разницу высот ($H$ = 4 – 2 = 2 м), а через $S$ – расстояние до мишени ($S$ = 10 м). Будем считать шарик материальной точкой; поскольку его скорость невысока, сопротивлением воздуха можно пренебречь. Известные из физики уравнения движения запишутся в виде:

$$x = v_0 \cdot t \cdot \cos\alpha, \quad y = v_0 \cdot t \cdot \sin\alpha - \frac{gt^2}{2},$$

где $g \approx 9{,}81$ м/с$^2$ – ускорение свободного падения. Задача сводится к тому, чтобы найти два неизвестных, $t$ и $\alpha$, при которых $x = S$ и $y = H$, то есть

$$S = v_0 \cdot t \cdot \cos\alpha, \quad H = v_0 \cdot t \cdot \sin\alpha - \frac{gt^2}{2}.$$

Время $t$ можно сразу выразить из первого уравнения:

$$t = \frac{S}{v_0 \cdot \cos\alpha}.$$

Подставляя этот результат во второе уравнение, получаем уравнение с одним неизвестным $\alpha$, которое можно привести к стандартному виду $f(x) = 0$, где:

$$f(x) = S \cdot \mathrm{tg}\,\alpha - \frac{g \cdot S^2}{2v_0^2 \cos^2\alpha} - H.$$

Решать его аналитически достаточно сложно[1], поэтому мы найдем приближенное решение. При вычислении тригонометрических функций угол измеряется в радианах, поэтому нужно искать решение в диапазоне углов $\alpha$ от 0 до $\pi/2$.

[1] Хотя можно, например, использовав замену $z = \cos^2\alpha$.

Мы не знаем, сколько решений имеет уравнение, поэтому изменим метод перебора так, чтобы найти все решения. Цикл **while** не будет останавливаться на первом найденном решении, а будет продолжаться, пока угол не станет больше $\pi/2$. Если в какой-то полосе есть решение, вычисляем угол в градусах и выводим его на экран. Приведем основные части программ на алгоритмическом языке

```
pi:= 3.1415926;
x:= 0;
delta := 2*eps;
нц пока x < pi/2
  если f(x)*f(x+delta) < 0 то
    вывод 'Угол: ', (x+eps)*180/pi, ' градусов', нс;
  все
  x:= x + delta;
кц
```

и на Паскале

```
x:= 0;
delta:= 2*eps;
while x < pi/2 do begin
  if f(x)*f(x+delta) < 0 then begin
    alpha:= (x+eps)*180/pi;
    writeln('Угол: ', alpha:4:1, ' градусов');
  end;
  x:= x + delta;
end;
```

В этой программе в переменной **x** хранится угол в радианах, а в переменной **alpha** – угол в градусах. Если запустить эту программу, мы увидим, что уравнение имеет два решение – углы примерно равны 35,6° и 65,8°.

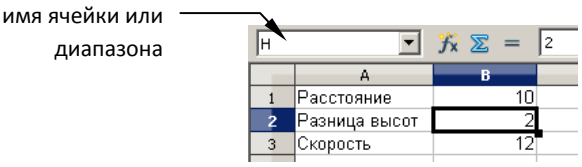
Попробуйте применить в этой же задаче метод деления отрезка пополам. Подумайте, с какими проблемами мы здесь сталкиваемся, и почему они возникают.

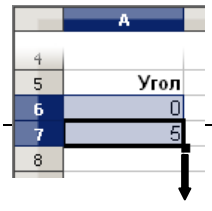
Повторите вычисления для начальных скоростей 10 м/с и 20 м/с и объясните полученные результаты.

## 9.2.5. Использование табличных процессоров

Для решения уравнений можно использовать табличный процессор, например, *OpenOffice Calc* или *Microsoft Excel*. Обычно сначала строится график функции $f(x)$, который позволяет определить количество решений уравнения и их примерное расположение; затем используется модуль «Поиск решения». Далее мы будем рассматривать программу *OpenOffice Calc,* указывая на незначительные отличия *Excel*.

Введем исходные данные, как показано на рисунке. Для того, чтобы формулы выглядели более привычно, дадим ячейкам B1, B2 и B3 имена **S**, **H** и **v** (их можно ввести в левом верхнем углу таблицы).



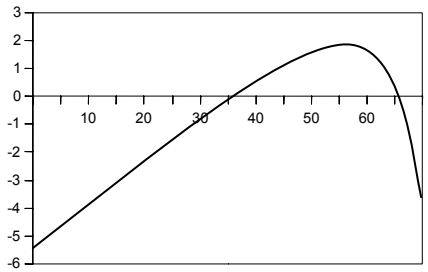



В столбце А заполним ряд значений углов от 0° до 85° с шагом 5°. Для этого введем два первых значения, выделим эти ячейки и «растянем» за маркер заполнения (квадратик в правом нижнем углу выделенной части).

Добавим столбцы, в которых для каждого угла будут вычисляться его значение в радианах (с помощью стандартной функции RADIANS[2]), время полета, координату $y$ и значение функции $f(x)$:

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Расстояние** | 10 | | | |
| 2 | **Разница высот** | 2 | | | |
| 3 | **Скорость** | 12 | | | |
| 4 | | | | | |
| 5 | **Угол** | **Угол (рад)** | **Время** | **y** | **f(x)** |
| 6 | 0 | =RADIANS(A6) | =S/v/COS(B6) | =v*SIN(B6)*C6-9,81*C6^2/2 | =D6-H |
| 7 | 5 | | | | |
| 8 | 10 | | | | |

Обратите внимание, что в формулах мы используем имена ячеек **S**, **H** и **v**. Это абсолютные ссылки, не меняющиеся при копировании; например, вместо имени **L** можно было бы написать адрес **$B$1**, но это было бы менее понятно. Эти формулы можно просто «растянуть» (скопировать) вниз за маркер заполнения.

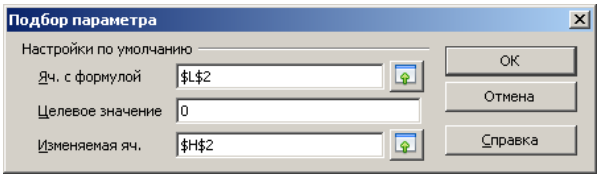



Теперь построим график функции $f(x)$. Сначала нужно выделить данные в столбцах А и Е, это можно сделать, если удерживать нажатой клавишу *Ctrl*. Затем строим диаграмму типа «Диаграмма XY[3]». График функции пересекает ось OX в двух точках, то есть уравнение $f(x)=0$ имеет два решения, одно около 35°, второе – около 65°.

Теперь уточним решение, используя возможности табличного процессора, в котором реализован один из приближенных методов решения уравнений. Для этого нужно знать *начальное приближение* $x_0$ – значение неизвестной величины, достаточно близкое к решению. По графику мы определили, что первый раз график пересекает ось OX для значения угла около 35°, поэтому можно взять $x_0 = 35°$. Запишем это значение в свободную ячейку, например, в H2, и добавим недостающие формулы так, чтобы получить значение функции $f(x)$ в ячейке L2:

| | H | I | J | K | L |
|---|---|---|---|---|---|
| 1 | **Угол** | **Угол (рад)** | **Время** | **y** | **f(x)** |
| 2 | 35 | | | | |

Задача подбора параметра формулируется так: «установить в ячейке ... значение ..., изменяя значение ячейки ...». Например, в нашем случае нужно установить в ячейке L2 значение 0, изменяя H2. Ячейка L2 называется *целевой*, потому что наша цель – получить в ней нуль. Ячейка H2 – это изменяемая ячейка. В верхнем меню выбираем пункт «Сервис – Подбор параметра» и вводим эти данные[4]:




После нажатия на кнопку ОК найденное решение уравнения будет записано в ячейку H2.

Как же найти второе решение? Для этого нужно выбрать другое начальное приближение, например, $x_0 = 70°$, в остальном порядок действий не меняется. Сделайте это самостоятельно.

[2] В *Excel* – функция РАДИАНЫ.

[3] В *Excel* – диаграмма «Точечная».

[4] В *Excel-2007* пункт «Подбор параметра» находится в группе «Анализ что-если» на вкладке «Данные».

Проверьте, что будет происходить при изменении начальной скорости до 10 м/с и до 20 м/с. Объясните эти результаты с точки зрения физики.

### *Контрольные вопросы*

1. Какие методы называются приближенными? В каких случаях они используются?
2. Что такое итерационный метод?
3. Сравните приближенные и аналитические методы решения уравнений. В чем достоинства и недостатки каждого подхода?
4. Объясните, в чем заключается метод перебора. В чем его недостатки?
5. Как с помощью математических операций определяют, есть ли решение уравнения на заданном интервале? В каких случаях такой подход не сработает?
6. Как избежать зацикливания в методе перебора?
7. Объясните, почему при ширине полосы $\delta = 2\varepsilon$ методом перебора можно найти решение с точностью $\varepsilon$.
8. Что такое отделение корней и уточнение корней?
9. Объясните изменения, сделанные в первоначальной программе для решения уравнения методом перебора, которые позволили в одном цикле найти все решения на заданном интервале.
10. Объясните, как работает метод деления отрезка пополам. Сравните его с методом перебора.

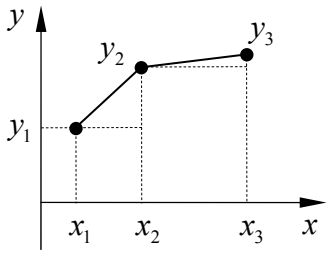
### *Задачи*

1. *Попытайтесь улучшить метод перебора, используя значение функции, вычисленное на предыдущем шаге цикла.
2. Решите уравнение $x^2 = 5\cos(x-1)$ методом перебора и методом деления отрезка пополам. Сравните количество шагов цикла при использовании каждого метода. (Ответ: -0,517; 1,833)
3. Решите уравнение $x^2 = 5\cos(x-1)$, используя табличный процессор.

## 9.3. Дискретизация

### 9.3.1. Вычисление длины кривой

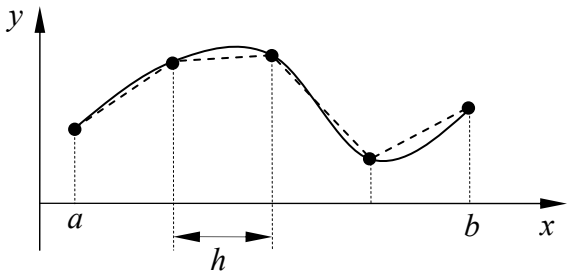
Определим длину траектории $L$, по которой летит шарик, брошенный под углом к горизонту (см. задачу в предыдущем параграфе). Это оказывается не так просто, потому что траектория – кривая линия.




Постараемся как-то свести задачу к более простой, которую мы умеем решать. Если бы линия состояла из отдельных отрезков, ее длину можно было бы точно вычислить с помощью теоремы Пифагора. Например, длина ломаной на рисунке справа равна

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} + \sqrt{(x_3 - x_2)^2 + (y_3 - y_2)^2}\ .$$

Используя эту идею, разделим кривую линию на небольшие участки и заменим каждый участок отрезком так, чтобы получилась ломаная (штриховая линия на рисунке):




Обычно разбивают исходный отрезок $[a,b]$ (на котором нужно определить длину кривой) на равные участки длиной $h$. Конечно, длина ломаной отличается от длины кривой, но естественно предположить, что при уменьшении шага разбиения $h$ эта разница будет уменьшаться.

Обратим внимание, что мы фактически выполнили *дискретизацию* – представили кривую в виде набора точек, при этом информация о поведении функции между этими точками была потеряна (вспомните оцифровку звука!). Величина $h$ называется *шагом дискретизации*.

Подведём итоги:

- дискретизация позволяет представить задачу в виде, пригодном для компьютерных расчетов;
- при дискретизации часть информации теряется, поэтом все методы, основанные на дискретизации – приближенные, они решают задачу с некоторой ошибкой;
- чтобы уменьшить ошибку, нужно уменьшать шаг дискретизации (увеличивать количество точек), но при этом возрастет объем (и время) расчетов.

Теперь можно составить программу. Будем считать, что константы (или переменные) **a**, **b** и **h** задают границы интервала и шаг дискретизации. Тогда основная часть программы может выглядеть так:

```
x:= a; L:= 0;
нц пока x < b
  y1:= f(x);
  y2:= f(x+h);
  L:= L + sqrt(h*h + (y1-y2)*(y1-y2));
  x:= x + h;
кц
вывод 'Длина кривой ', L;
```

или так:

```
L := 0;
x := a;
while x < b do begin
  y1 := f(x);
  y2 := f(x+h);
  L := L + sqrt(h*h + (y2-y1)*(y2-y1));
  x := x + h;
end;
writeln('Длина кривой ', L:10:3);
```

Возвращаясь к задаче с шариком, вспомним, что его движение описывается уравнениями

$$x = v_0 \cdot t \cdot \cos\alpha, \quad y = v_0 \cdot t \cdot \sin\alpha - \frac{gt^2}{2}.$$

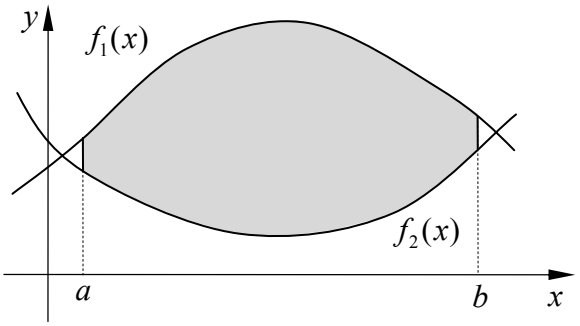
Если выразить время из первого уравнения и подставить во второе, получается

$$y = f(x) = x \cdot \mathrm{tg}\,\alpha - \frac{g \cdot x^2}{2v_0^2 \cos^2\alpha}.$$

Это и есть функция, график которой нас интересует. Написать полную программу вы уже можете самостоятельно. Проверьте ее работу для разных значений исходных данных (скорости и угла вылета, расстояния).

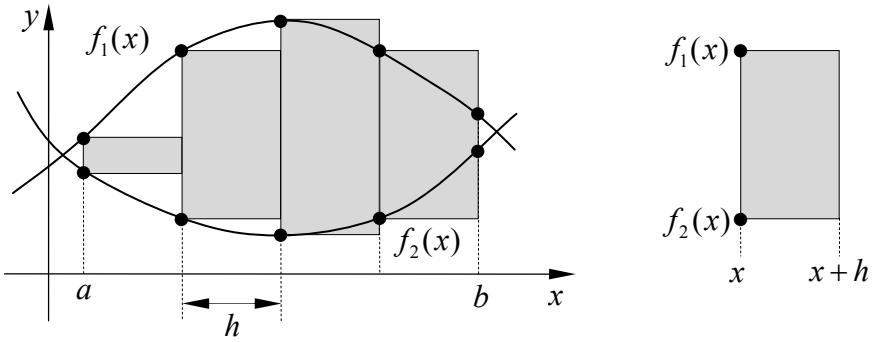
## 9.3.2. Вычисление площадей фигур

Применим метод дискретизации в другой достаточно сложной задаче – для вычисления площади сложной фигуры. Пусть фигура, площадь которой нас интересует, ограничена графиками функций $y = f_1(x)$ и $y = f_2(x)$:




В некоторых простых случаях (но далеко не всегда!) площадь такой фигуры можно вычислить аналитически с помощью методов высшей математики. Мы же будем использовать приближенные методы, применять которые значительно проще.

Как и при вычислении длины кривой, применим *дискретизацию* – разделим фигуру на отдельные полоски, и заменим каждую полоску на другую фигуру, для которой мы можем легко найти площадь, например, на прямоугольник.




Понятно, что площадь исходной фигуры не совпадает с суммой площадей получившихся прямоугольников. Так и должно было быть, ведь при дискретизации информация о поведении функций между точками отсчета была утеряна. Однако, при уменьшении шага дискретизации $h$ эта разница будет уменьшаться.

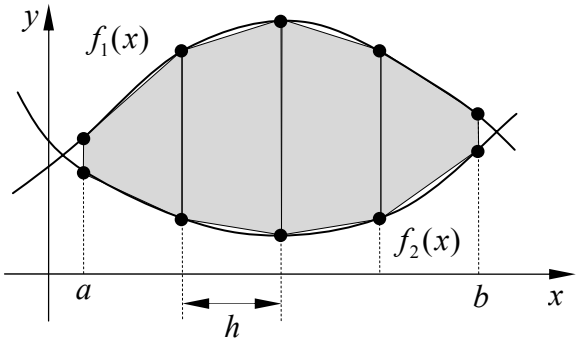
На рисунке высота прямоугольника рассчитывается как разность функций на левой границе интервала (можно использовать и правую границу). На практике обычно вычисляют высоту в *середине интервала* (в точке $x + h/2$), тогда площадь прямоугольника будет более близка к площади полосы исходной фигуры. Заметим, что ширина каждого из прямоугольников равна $h$, поэтому в программе умножение на $h$ можно выполнить в конце цикла:

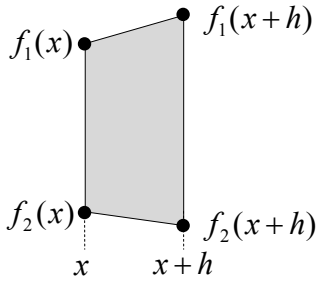
```
S:= 0; x:= a;
нц пока x < b
  S:= S + f1(x+h/2) - f2(x+h/2);
  x:= x + h;
кц
S:= h*S;
```

```
S := 0; x := a;
while x < b do begin
  S := S + f1(x+h/2) - f2(x+h/2);
  x := x + h;
end;
S := h*S;
```

```
вывод 'Площадь ', S;
```
```
writeln('Площадь ', S:8:3);
```

Вместо прямоугольника для замены можно использовать еще одну фигуру, для которой легко считается площадь – *трапецию*:




Площадь трапеции равна произведению полусуммы ее оснований на высоту, то есть для элементарной трапеции с левой границей в точке $x$ получаем

$$\frac{h}{2}\cdot\left[f_1(x)-f_2(x)+f_1(x+h)-f_2(x+h)\right].$$

Простейшая программа выглядит так:

```
S:= 0; x:= a;
нц пока x < b
  S:= S + f1(x) - f2(x);
  S:= S + f1(x+h) - f2(x+h);
  x:= x + h;
кц
S:= h*S/2;
вывод 'Площадь ', S;
```

```
S := 0; x := a;
while x < b do begin
  S := S + f1(x) - f2(x)
         + f1(x+h) - f2(x+h);
  x := x + h;
end;
S := h*S/2;
writeln('Площадь ', S:8:3);
```

Можно заметить, что правое основание одной трапеции совпадает с левым основание следующей, поэтому можно немного изменить программу, чтобы на каждом шаге цикла вычислялась разность функций только в одной точке. Сделайте это самостоятельно.

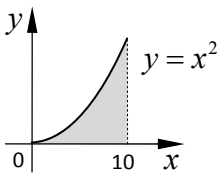
## ? *Контрольные вопросы*

1. Что такое дискретизация?
2. Что дает дискретизация в рассмотренных задачах?
3. Что такое шаг дискретизации? Как должна быть связана его величина с длиной отрезка $[a,b]$?
4. Как зависит точность и время вычислений от выбора шага дискретизации?
5. Почему методы, основанные на дискретизации, всегда дают приближенный результат?
6. Подумайте, можно ли выполнять дискретизацию с переменным шагом. Ответ обоснуйте.

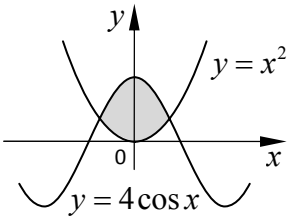
## *Задачи*

1. Измените программу для вычисления длины кривой так, чтобы на каждом шаге цикла вычислять только одно значение функции.
2. Найдите длину параболы $y=x^2$ на интервале $x\in[0,10]$. (Ответ: 101,05)
3. Для примера, разобранного в предыдущем пункте, вычислите длину траектории движения шарика для углов вылета 35,5° и 65,8°. Сравните полученные результаты. Постройте эти траектории с помощью табличного процессора. (Ответ: 10,6 м; 14,9 м)
4. Решите предыдущую задачу при разных значениях шага. Какой шаг вы рекомендуете выбрать для этого случая? Почему?

5. В чем заключается дискретизация при вычислении площади?
6. *Измените программу для вычисления площади методом трапеций так, чтобы повторно не вычислять одни и те же величины.
7. Найдите площадь фигуры, ограниченной параболой $y = x^2$ и осью OX, на интервале $x \in [0,10]$. (Ответ: 333,33)




8. *Найдите площадь фигуры, ограниченной графиками функций $y = x^2$ и $y = 4\cos x$.
(Ответ: 6,304)




9. *Найдите площадь фигуры, ограниченной эллипсом
$$\frac{x^2}{4} + \frac{y^2}{9} = 1.$$
(Ответ: $6\pi$)
10. *Найдите с помощью приближенных методов площадь круга радиуса $R$ = 2. Используя это значение, из формулы $S = \pi \cdot R^2$ определите приближенно число π.

# 9.4. Оптимизация

## 9.4.1. Что такое оптимизация?

**Оптимизация** – это поиск наилучшего (*оптимального*) решения задачи в заданных условиях.

С точки зрения математики, цель оптимизации – выбрать неизвестную величину $x$ (или несколько неизвестных величин – массив) наилучшим образом.

Чтобы задача оптимизации была корректной, нужно определить *целевую функцию* $f(x)$, которая позволяет сравнивать два решения. Оптимальным называется такое решение, при котором целевая функция достигает максимума (если это «доходы», «прибыль») или минимума («расходы», потери»):

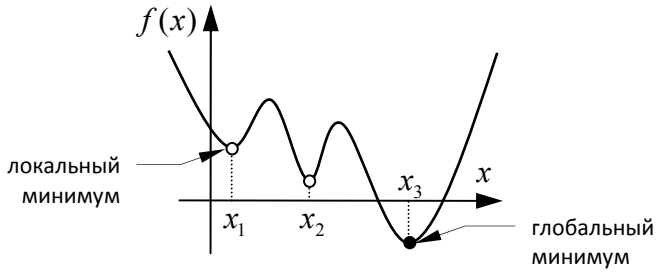
$$f(x) \to \max \text{ или } f(x) \to \min.$$

Чтобы задача оптимизации стала осмысленной, нужно ввести *ограничения*. Например, человек хочет построить загородный дом за минимальную цену (здесь целевая функция – это общая цена строительства, нужно сделать ее минимальной). Очевидно, что лучшее решение – не строить дом вообще, при этом расходы будут равны нулю. Такое «оптимальное» решение получено потому, что мы не задали ограничения (например, нужен двухэтажный дом с гаражом, балконом и камином).

## 9.4.2. Локальные и глобальный минимумы

По традиции в теории оптимизации рассматривают задачу поиска минимума. Если нужно найти максимум, просто меняют знак функции: значение функции $f(x)$ максимально там, где значение функции $-f(x)$ минимально.

В математике различают *локальный* («местный») и *глобальный* («общий») минимум. В точках $x_1$, $x_2$ и $x_3$ функция, график которой показан на рисунке, имеет *локальные минимумы*, этот значит, что слева и справа от этих точек функция возрастает.



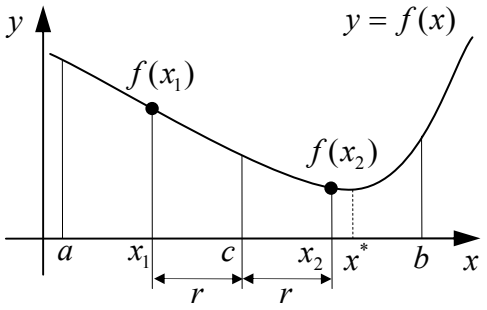

Минимум в точке $x_3$ – *глобальный*, потому что здесь функция имеет наименьшее значение во всей рассматриваемой области.

Очевидно, что нас всегда интересует глобальный минимум. Однако большинство существующих методов оптимизации[5] предназначены именно для поиска *локальных* минимумов вблизи заданной начальной точки (*начального приближения*). Можно представить себе, что график функции – это срез поверхности, на которую устанавливается шарик в некоторой начальной точке; куда этот шарик скатится, такой минимум и будет найден.

Результат оптимизации зависит от выбранного начального приближения.

## 9.4.3. Метод дихотомии

*Дихотомия* (греч. διχοτομία – «деление надвое») – это метод поиска минимума функции, который очень напоминает метод деления отрезка пополам (бисекции). Пусть дана непрерывная функция $f(x)$, имеющая на отрезке $[a,b]$ один минимум в точке $x^*$. Нужно найти это значение $x^*$ с заданной точностью $\varepsilon$.




Как и в методе бисекции, мы последовательно «сжимаем» отрезок, пока его ширина не будет достаточно мала (меньше, чем $2\varepsilon$). На каждом шаге

1) вычисляется середина интервала $c = \dfrac{a+b}{2}$;
2) вычисляются координаты двух точек, симметричных относительно середины: $x_1 = c - r$ и $x_2 = c + r$, где $0 < r < (b-a)/2$ – некоторое число;

[5] Для некоторых типов функций существуют методы глобальной оптимизации, но они, как правило, сложны и выходят за рамки школьного курса.

3) сравниваются значения функции в этих точках: если $f(x_1) > f(x_2)$ минимум функции находится на отрезке $[x_1, b]$, поэтому отрезок $[a, x_1]$ можно отбросить – переместить левую границу в точку $x_1$; если $f(x_1) < f(x_2)$ правая граница интервала перемещается в точку $x_2$.

Остается один вопрос: как выбирать $r$ на каждом шаге? Проще всего вычислять его по формуле $r = k \cdot (b - a)$, где $k$ – постоянный коэффициент ($0 < k < 0{,}5$). Тогда ширина интервала на следующем шаге будет равна

$$\frac{b-a}{2} + k \cdot (b-a) = (0{,}5 + k) \cdot (b-a),$$

то есть составит $0{,}5 + k$ от первоначальной. Для ускорения поиска выгодно, чтобы это отношение было как можно меньше, то есть чтобы коэффициент $k$ был возможно ближе к нулю (ноль выбирать нельзя, потому что при этом точки $x_1$ и $x_2$ совпадут и метод не будет работать). Программа может выглядеть примерно так:

```
k:= 0.01;
delta:= 2*eps;
нц пока b - a > delta
  r := k*(b - a);
  x1 := (a + b)/2 - r;
  x2 := (a + b)/2 + r;
  если f(x1) > f(x2) то
        a := x1
  иначе b := x2;
  все
кц
вывод 'Решение ', (a+b)/2;
```

```
k := 0.001;
delta := 2*eps;
while b - a > delta do begin
  r := k*(b - a);
  x1 := (a + b)/2 - r;
  x2 := (a + b)/2 + r;
  if f(x1) > f(x2) then
       a := x1
  else b := x2;
end;
writeln('Решение: ', (a+b)/2:10:3 );
```

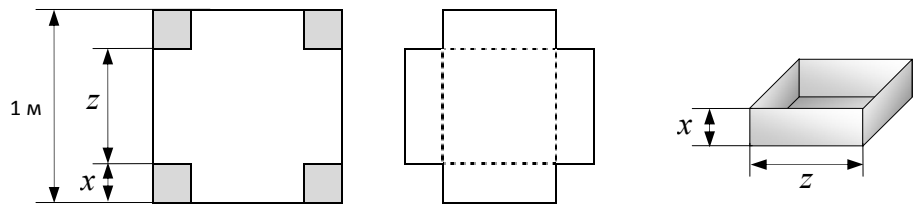
В качестве упражнения можно исследовать работу этой программы при разных значениях $k$, подсчитав для каждого варианта количество шагов цикла, которое потребовалось для получения решения с заданной точностью.

Существует вариант метода дихотомии, при котором на каждом шаге цикла нужно вычислять только одно значение функции (второе «переходит» с предыдущего шага). В этом случае нужно выбирать коэффициент $k$ равный отношению «золотого сечения»:

$$k = \frac{1+\sqrt{5}}{2} \approx 0{,}618...$$

## 9.4.4. Пример: оптимальная раскройка листа

Рассмотрим пример практической задачи оптимизации. В углах квадратного листа железа, сторона которого равна 1 м, вырезают четыре квадрата со стороной $x$. Затем складывают получившуюся развертку (по штриховым линиям на рисунке), сваривают швы и таким образом получается бак:




Требуется выбрать размер выреза $x$ так, чтобы получился бака наибольшего объема.

Для того, чтобы грамотно поставить задачу оптимизации, нужно

1) определить *целевую функцию*: в данном случае выразить объем бака через неизвестную величину $x$;
2) задать ограничения на возможные значения $x$.

Легко видеть, что основание получившегося бака – это квадрат со стороной $z$, а его высота равна $x$. Однако, величина $z$ зависит от $x$ и равна $z = 1 - 2x$, поэтому объем бака вычисляется по формуле $V = x(1-2x)^2$, это и есть целевая функция, для которой нужно найти максимум.

Понятно, что $x$ не может быть меньше нуля. С другой стороны, $x$ не может быть больше, чем половина стороны исходного листа (0,5 м), поэтому ограничения запишутся в виде двойного неравенства: $0 \le x \le 0{,}5$. Заметим, что при $x = 0$ и $x = 0{,}5$ объем бака равен нулю (в первом случае равна нулю высота, во втором – площадь основания).

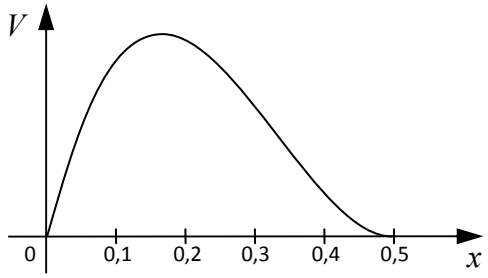
Таким образом, нужно искать минимум целевой функции $f(x) = x(1-2x)^2$ на интервале $[0;0{,}5]$. Для этого можно использовать метод дихотомии (сделайте это самостоятельно). Не забудьте, что приведенный выше вариант программы рассчитан на поиск минимума, а в этой задаче нужно найти максимум.

(Ответ: $1/6$)

## 9.4.5. Использование табличных процессоров

В стандартной поставке *OpenOffice Calc* модуль оптимизации работает только для линейных функций. Для того, чтобы решить рассмотренную выше задачу с баком, нужно установить расширение *Solver for Nionlinear Programming* (*extensions.services.openoffice.org/project/NLPSolver*) и в параметрах модуля оптимизации выбрать один из методов нелинейной оптимизации. В табличном процессоре *Excel* оптимизация выполняется с помощью стандартной надстройки «Поиск решения».

Сначала построим график функции, как мы делали это при решении уравнения.



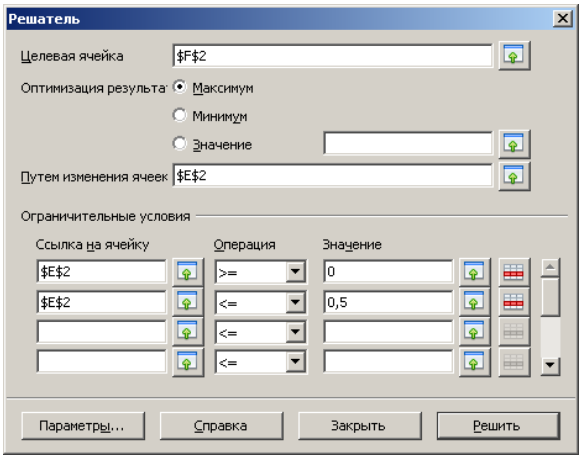

По этому графику видно, что функция имеет единственный максимум в районе точки $x_0 = 0{,}2$. Это значение можно выбрать в качестве начального приближения для поиска.

Выделим в таблице две ячейки (например, E2 и F2), в первую запишем начальное приближение (0,2), а во вторую – формулу для вычисления объема для этого значения $x$:

| | E | F |
|---|---|---|
| 1 | x | Объем |
| 2 | 0,200 | 0,072 |

Задача оптимизации формулируется в виде «найти максимум (или минимум) целевой функции в ячейке …, изменяя значения ячеек … при ограничениях …». В нашей задаче целевая ячейка – F2 (нужно найти ее минимум), изменяемая ячейка – E2.

Выбираем в верхнем меню пункт «Сервис – Поиск решения», в появившемся окне вводим адреса целевой и изменяемой ячеек (для этого можно установить курсор в нужное поле ввода и щелкнуть по ячейке) и ограничения:




После щелчка по кнопке «Решить» в ячейке E2 будет записано оптимальное значение $x$, а в целевой ячейке – соответствующее (максимальное) значение объема.

Надстройка «Поиск решения» позволяет

- находить максимум или минимум целевой функции;
- решать уравнения, задавая желаемое значение целевой функции;
- использовать несколько изменяемых ячеек и диапазонов, например запись

  **A2:A6;B15**

  в списке изменяемых ячеек означает «изменять все ячейки диапазона **A2:A6** и ячейку **B15**»;
- использовать ограничения типа «меньше или равно», «больше или равно», «равно», «целое» и «двоичное» (только 0 или 1).

Кнопка «Параметры» позволяет опытным пользователям менять настройки алгоритма оптимизации.

## *Контрольные вопросы*

1. Что такое оптимизация?
2. Что такое целевая функция?
3. Какое решение называется оптимальным?
4. Почему выражение «самый оптимальный» безграмотно?
5. Что можно сказать о рекламной фразе «Этот крем обеспечивает оптимальный цвет лица»?
6. Зачем нужны ограничения в задаче оптимизации?
7. В чем разница между понятиями «локальный минимум» и «глобальный минимум»?
8. Что такое начальное приближение?
9. Почему результат решения задачи оптимизации чаще всего зависит от выбора начального приближения?
10. Объясните принцип работы метода дихотомии.
11. Обязательно ли при использовании метода дихотомии брать пробные точки симметрично относительно середины отрезка? Ответ обоснуйте.
12. Когда метод дихотомии не будет работать (может выдать неверный ответ)?
13. Подумайте, можно ли задачу решения уравнения сформулировать как задачу оптимизации.

## *Задачи*

1. Примените метод дихотомии для решения задачи оптимальной раскройки, которая разобрана в тексте. Решите задачу, используя разные значения коэффициента $k$, и постройте график зависимости количества шагов цикла от $k$.

2. *Напишите программу, которая реализует метод «золотого сечения» (на каждом шаге вычисляется только одно значение функции).
3. *Банка имеет форму цилиндра[6], полная площадь ее поверхности (боковая поверхность и два круга-основания) равна 100 см$^2$. Определите радиус и высоту банки, которая при этих условиях имеет максимальный объем. (Ответ: R ≈ 2,3 см; H ≈ 4,6 см)
4. *Банка имеет форму цилиндра, ее объем равен 500 см$^3$. Определите радиус и высоту банки, которая при этих условиях имеет минимальную площадь полной поверхности. (Ответ: R ≈ 4,3 см; H ≈ 8,6 см)
5. Фирма «Рога и копыта» хочет провести рекламную кампанию в газетах. Данные о цене рекламного объявления и тиражах газет внесены в таблицу:

| **Газета** | **Тираж** | **Цена 1 объявл.** | **Объявлений** | **Расходы** | **Охват** |
|---|---|---|---|---|---|
| *Ведомости* | 10000 | 1000р. | 1 | 1000 р. | 10000 |
| *Туризм* | 3500 | 570р. | 2 | 1140 р. | 7000 |
| *Спорт* | 6000 | 700р. | 3 | 2100 р. | 18000 |
| *Правда* | 20000 | 1250р. | 4 | 5000 р. | 80000 |
| **Всего** | | | | 9240 р. | 115000 |

   В каждую газету нужно дать не менее одного и не более 6 объявлений. С помощью надстройки «Поиск решения» табличного процессора определите, сколько объявлений нужно дать в каждую газету, чтобы обеспечить общий охват не менее 200000 человек и при этом израсходовать как можно меньше денег. (Ответ: 6, 1, 3, 6)
6. В условиях предыдущей задачи определите, сколько объявлений нужно дать в каждую газету, чтобы обеспечить наибольший общий охват и при этом израсходовать не более 15 000 рублей. (Ответ: 6, 1, 1, 6)

# 9.5. Статистические расчеты

**Статистика** – это наука, которая изучает методы обработки и анализа больших массивов данных.

Табличные процессоры стали незаменимым инструментом для решения этих задач. И *Excel*, и *OpenOffice Calc* содержат несколько десятков встроенных статистических функций.

## 9.5.1. Свойства ряда данных

Простейшая задача статистики – изучить свойства одного *ряда данных*. Ряд данных $X$ длиной $n$ – это множество значений $x_1, x_2, \ldots x_n$. Для ряда можно определить количество элементов, их сумму, среднее значение, минимальное и максимальное значения и т.д. Далее мы приводим как английские названия функций (для *OpenOffice Calc*), так и русские (для русской версии *Excel*).

Пусть ряд данных записан в ячейках A1:A20. Его сумма, среднее, минимальное и максимальное значения могут быть определены с помощью следующих формул:

| | | |
|---|---|---|
| сумма: | `=SUM(A1:A20)` | `=СУММ(A1:A20)` |
| среднее: | `=AVERAGE(A1:A20)` | `=СРЗНАЧ(A1:A20)` |
| минимальное: | `=MIN(A1:A20)` | `=МИН(A1:A20)` |
| максимальное: | `=MAX(A1:A20)` | `=МАКС(A1:A20)` |

Функции **COUNT** (**СЧЕТ**) и **AVERAGE** (**СРЗНАЧ**) учитывают только числа в указанном диапазоне (без учета пустых и текстовых ячеек).

[6] Задачи 16 и 17 предложены В.Я. Лаздиным.

С помощью функции **COUNTIF** (**СЧЕТЕСЛИ**) можно подсчитать число элементов ряда, удовлетворяющих некоторому условию. Например, если в диапазоне A1:A20 записаны школьные отметки, формула

```
=COUNTIF(A1:A20;"=5")
```

вычисляет число пятерок, а формула

```
=COUNTIF(A1:A20;">3")
```

общее число четверок и пятерок (всех ячеек, значения которых больше 3).

Более сложная характеристика ряда – *среднеквадратическое отклонение* или *стандартное отклонение* $\sigma_x$, с помощью которого оценивается «разброс» значений $x_1, x_2, \ldots x_n$ относительно среднего значения ряда $\overline{x}$ :

$$\sigma_x = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \; .$$

Среднеквадратическое отклонение – это неотрицательная величина (подумайте, почему). Чем больше $\sigma_x$, тем больше разброс значений относительно среднего. Для вычисления стандартного отклонения в табличных процессорах используется функция **STDEVP** (**СТАНДОТКЛОНП**):

```
=STDEVP(A1:A20)          =СТАНДОТКЛОНП(A1:A20)
```

## 9.5.2. Условные вычисления

В языке программирования важнейшую роль играют условные операторы (ветвления), позволяющие выбирать один из двух (или нескольких) вариантов обработки данных. В табличных процессорах тоже возможны *условные вычисления*, при которых в ячейку заносится то или иное значение в зависимости от выполнения какого-то условия.

Предположим, что в книжном интернет-магазине «Бука» доставка покупок бесплатна для тех, кто сделал заказ на сумму более 500 рублей, а для остальных доставка стоит 20% от суммы.

| | A | B | C |
|---|---|---|---|
| 1 | **Заказ** | **Сумма** | **Доставка** |
| 2 | 1234 | 256 руб. | 51 руб. |
| 3 | 1345 | 128 руб. | 26 руб. |
| 4 | 1456 | 1 024 руб. | 0 |
| 5 | 1565 | 512 руб. | 0 |
| 6 | 1576 | 345 руб. | 69 руб. |

Таким образом, есть два варианта вычисления стоимости доставки, поэтому в формулах столбца C нужно использовать ветвление. Например, алгоритм вычисления значения ячейки C2 может выглядеть так: «если B2>500, записать в ячейку 0, иначе записать значение B2*0,2». В программе на Паскале мы бы записали

```
if B2 > 500 then
     C2 := 0
else C2 := B2*0.2;
```

В табличных процессорах для условных вычислений используют функцию **IF** (**ЕСЛИ**):

```
=IF(B2>500;0;B2*0,2)          =ЕСЛИ(B2>500;0;B2*0,2)
```

У этой функции три аргумента, разделенные точками с запятой:

1) условие (**B2>500**);
2) значение ячейки в том случае, когда условие истинно (**0**);
3) значение ячейки в том случае, когда условие ложно (**B2*0,2**).

Первый аргумент может быть сложным условием, которое строится с помощью функций NOT (НЕ, отрицание), AND (И, логическое умножение) и OR (ИЛИ, логическое сложение). Напри-

мер, если бесплатная доставка распространяется только на заказы, у которых номер меньше 1500 и сумма больше 500 рублей, в ячейку C2 нужно записать такую формулу:

```
=IF(AND(A2<1500;B2>500);0;B2*0,2)
```

Здесь использовано сложное условие **AND(A2<1500;B2>500)**, которое истинно только при одновременном выполнении обоих простых: **A2<1500** и **B2>500**.

Второй и третий аргументы функции **IF** могут содержать вложенные вызовы этой функции. Пусть, например, для заказов стоимостью более 200 рублей (но не больше 500) стоимость доставки составляет 10% от суммы, то есть

```
if B2 > 500 then
  C2 := 0
else
  if B2 > 200 then
       C2 := B2*0.1
  else C2 := B2*0.2;
```

В табличном процессоре этот алгоритм запишется в виде

```
=IF(B6>500;0;IF(B6>200;B6*0,1;B6*0,2))
```

## 9.5.3. Связь двух рядов данных

Одна из задач обработки данных – установить взаимосвязь между величинами, процессами, явлениями. Пусть существуют два ряда данных одинаковой длины

$$x_1, x_2, \ldots x_n \text{ и } y_1, y_2, \ldots y_n .$$

Например, первый ряд – это температура воздуха за $n$ последних дней, а второй ряд – значения атмосферного давления в те же дни. Требуется определить, есть ли связь между этими рядами и оценить, насколько она сильная.

Для решения этой задачи чаще всего используется *коэффициент корреляции* (англ. *correlation* – взаимоотношение, связь):

$$\rho_{xy} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sigma_x \cdot \sigma_y}.$$

Здесь $\bar{x}$ и $\bar{y}$ – средние значения рядов, а $\sigma_x$ и $\sigma_y$ – их среднеквадратические отклонения.

Величина $\rho_{xy}$ – это безразмерный коэффициент[7], причем можно показать, что всегда $-1 \le \rho_{xy} \le 1$. Если $\rho_{xy} > 0$, то увеличение значения $x$ (в среднем) приводит к увеличению $y$; если же $\rho_{xy} < 0$, при увеличении $x$ значение $y$ чаще всего уменьшается.

Чем больше модуль $\rho_{xy}$, тем сильнее связь между двумя величинами. При $\rho_{xy} = -1$ или $\rho_{xy} = 1$ они строго связаны линейной зависимостью $y = kx + b$, где $k$ и $b$ – некоторые числа. В случае $\rho_{xy} = -1$ эта зависимость убывающая ($k < 0$), а при $\rho_{xy} = 1$ – возрастающая ($k > 0$).

Считается, что между $x$ и $y$ есть сильная связь, если $|\rho_{xy}| > 0{,}5$. При меньших значениях $\rho_{xy}$ делать какие-то далеко идущие выводы не следует (связь слабая или не обнаружена).

Для вычисления коэффициента корреляции в табличных процессорах используется функция **CORREL** (**КОРРЕЛ**):

```
=CORREL(A1:A20;B1:B20)          =КОРРЕЛ(A1:A20;B1:B20)
```

[7] Попробуйте доказать это самостоятельно.

Обратите внимание, что у этой функции два аргумента (два ряда данных одинаковой длины), адреса двух диапазонов отделяются точкой с запятой.

Нужно учитывать, что коэффициент корреляции лучше всего обнаруживает линейную зависимость. Если связь есть, но она далека от линейной, коэффициент корреляции может быть невысок. В таких случаях для установления связи нужно использовать более сложные методы, которые мы здесь рассматривать не будем.

## ? *Контрольные вопросы*

1. Что изучает статистика? Как вы думаете, в чем ее задача?
2. Как влияют пустые ячейки на результат работы функций **COUNT** и **AVERAGE**?
3. Чем отличаются функции **COUNT** и **COUNTIF**?
4. Что показывает среднеквадратическое отклонение?
5. Что показывает коэффициент корреляции?
6. Для двух рядов коэффициент корреляции равен (– 0,5). Что можно сказать о возможной связи этих рядов между собой?
7. Для двух рядов коэффициент корреляции равен 0,1. Что можно сказать о возможной связи этих рядов между собой?

## *Задачи*

1. В электронной таблице значение формулы **=SUM(B1:B2)** равно 5. Чему равно значение ячейки B3, если значение формулы **=AVERAGE(B1:B3)** равно 3? (Ответ: 4)
2. В электронной таблице значение формулы **=SUM(C3:E3)** равно 12. Чему равно значение формулы **=AVERAGE(C3:F3)**, если значение ячейки F3 равно 4? (Ответ: 4)
3. В электронной таблице значение формулы **=SUM(A2:D2)** равно 12. Чему равно значение формулы **=AVERAGE(A2:E2)**, если значение ячейки E2 равно 13? (Ответ: 5)
4. В электронной таблице значение формулы **=AVERAGE(A6:C6)** равно (**–2**). Чему равно значение формулы **=SUM(A6:D6)**, если значение ячейки D6 равно 6? (Ответ: 0)
5. В электронной таблице значение формулы **=AVERAGE(B5:E5)** равно 10. Чему равно значение формулы **=SUM(B5:F5)**, если значение ячейки F5 равно 1? (Ответ: 41)
6. В электронной таблице значение формулы **=AVERAGE(A1:C1)** равно 6. Чему равно значение ячейки D1, если значение формулы **=SUM(A1:D1)** равно 7? (Ответ: -11)
7. В электронной таблице значение формулы **=AVERAGE(A3:D4)** равно 6. Чему равно значение формулы **=AVERAGE(A3:C4)**, если значение формулы **=SUM(D3:D4)** равно 6? (Ответ: 7)
8. В электронной таблице значение формулы **=AVERAGE(C2:D5)** равно 4. Чему равно значение формулы **=SUM(C5:D5)**, если значение формулы **=AVERAGE(C2:D4)** равно 6? (Ответ: -4)
9. *Как изменится значение ячейки C3, если после ввода формул переместить содержимое ячейки B2 в B3?
   (Ответ: уменьшится на 1)

| | A | B | C |
|---|---|---|---|
| 1 | 1 | 2 | |
| 2 | 2 | 6 | =COUNT(A1:B2) |
| 3 | | | =AVERAGE(A1:C2) |

10. Доставка товара в фирме «Рога и копыта» стоит 200 рублей, если в доме есть лифт. Если лифта нет, подъем на каждый этаж стоит 200 рублей:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Заказ | Этаж | Лифт | Доставка |
| 2 | 12 | 5 | нет | 1000 |
| 3 | 34 | 2 | да | 200 |
| 4 | 56 | 8 | да | 200 |

Какую формулу нужно записать в ячейку D2?

(Ответ: **=IF(C2="да";200;B2*200)**)

11. При приеме на работу претенденты проходят два тура собеседования. В каждом туре выставляется отметка от 0 до 100 баллов. На работу принимаются те, кто в каждом туре набрал не менее 80 баллов.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Фамилия** | **1 тур** | **2 тур** | **Принят** |
| 2 | Иванов | 80 | 80 | да |
| 3 | Петров | 90 | 70 | нет |
| 4 | Сидоров | 85 | 90 | да |

Какую формулу нужно записать в ячейку D2?

(Ответ: **=IF(AND(B2>=80;C2>=80);"да";"нет")**)

12. Решите предыдущую задачу при условии, что на работу принимаются все, кто набрал 90 баллов хотя бы в одном туре собеседования.

(Ответ: **=IF(OR(B2>=90;C2>=90);"да";"нет")**)

13. Сниженный тариф на телефонные разговор – 2 рубля за минуту разговора – действует в рабочие дни после 20 часов и в выходные дни. Обычный тариф – 5 рублей за минуту. Дни в таблице нумеруются с 1 до 7 (1 – понедельник).

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Код** | **Время** | **День недели** | **Тариф** |
| 2 | 12 | 21:12:20 | 2 | 2 руб. |
| 3 | 34 | 17:23:50 | 1 | 5 руб. |
| 4 | 56 | 10:21:42 | 7 | 2 руб. |

Какую формулу нужно записать в ячейку D2? Момент времени 20:00:00 в формуле нужно записывать как **TIME(20;0;0)**.

(Ответ: **=IF(OR(B2>TIME(20;0;0);C2>5);2;5)**)

14. При покупке на сумму более 1000 рублей в магазине «Рога и копыта» владельцам дисконтных карт предоставляется скидка 5%.

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Код** | **Цена** | **Дисконтная карта** | **Со скидкой** |
| 2 | 12 | 1 200 руб. | нет | 1 200 руб. |
| 3 | 34 | 1 450 руб. | да | 1 378 руб. |
| 4 | 56 | 750 руб. | да | 750 руб. |

Какую формулу нужно записать в ячейку D2?

(Ответ: **=IF(AND(B2>1000;C2="да");B2*0,95;B2)**)

15. Во время уценки цена всех товаров, которых хранятся на складе более 6 месяцев, снижается на 20% (если цена товара больше 1000 рублей) или на 10% (если цена меньше 1000 рублей).

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Код** | **Цена** | **Хранится (месяцев)** | **Новая цена** |
| 2 | 12 | 1 200 руб. | 3 | 1 200 руб. |
| 3 | 34 | 1 450 руб. | 7 | 1 160 руб. |
| 4 | 56 | 750 руб. | 12 | 675 руб. |

Какую формулу нужно записать в ячейку D2?

(Ответ: **=IF(C4>6;IF(B4>1000;B4*0,8;B4*0,9);B4)**)

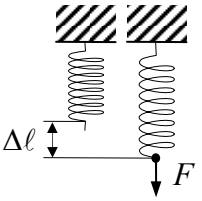
# 9.6. Обработка результатов эксперимента

## 9.6.1. Зачем это нужно?

Многие практические задачи связаны с проведением эксперимента, в результате которого исследователь с помощью измерительных приборов получает массивы данных. Затем эти данные необходимо обработать, для того чтобы выявить закономерности, подтвердить или опровергнуть выводы теории и т.п.

Например, с помощью динамометра и линейки можно экспериментально определить жесткость пружины. Для этого используется закон Гука, связывающий приложенную силу $F$, жесткость пружины $k$ и ее удлинение $\Delta\ell$ линейной зависимостью:

$$F = k \cdot \Delta\ell .$$




Величина $k$ зависит от материала пружины и ее размеров. Для определения жесткости $k$ на нижний конец пружины подвешивают груз известной массы $m$ (так что сила $F = mg$ тоже известна) и измеряют удлинение пружины $\Delta\ell$. Тогда $k = F / \Delta\ell$.

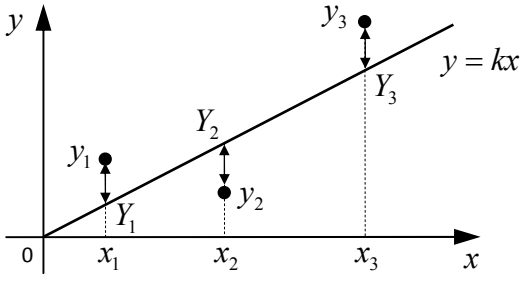
Обычно такой эксперимент проводится несколько раз, в результате получается серия значений $F_i$ (для $i = 1,2,...n$) и соответствующих им удлинений $\Delta\ell_i$. Для каждой пары рассчитанная жесткость $k_i = F_i / \Delta\ell_i$ будет, скорее всего, различной. Конечно, можно принять за $k$ среднее значение по всем полученным измерениям. Однако такой подход не очень хорошо обоснован с научной точки зрения. Поэтому были разработаны другие методы, один из которых рассматривается далее.

## 9.6.2. Метод наименьших квадратов

Предположим, что есть два ряда данных одинаковой длины: $x_1, x_2, ... x_n$ и $y_1, y_2, ... y_n$. Предполагается, что они связаны линейной зависимостью $y = k \cdot x$, где $k$ – неизвестный коэффициент. Требуется найти *оптимальное* значение $k$, которое лучше всего соответствует исходным данным.

Поскольку речь идет о задаче оптимизации, нужно определить функцию, которая позволяет оценить, насколько хорошо выбранная зависимость соответствует исходным данным. Предположим, что выбран некоторый коэффициент $k$, так что для каждого $x_i$ можно найти соответствующее ему значение функции $Y_i = k \cdot x_i$.

В идеале график функции должен проходить через все точки $(x_1, y_1)$, $(x_2, y_2)$, … $(x_n, y_n)$, то есть при всех $i$ должно выполняться условие $Y_i = y_i$. Однако, на практике этого, скорее всего, не будет.



Отклонение полученной линии от исходных данных определяется разностями $Y_i - y_i$: чем они меньше (по модулю), тем лучше соответствие. Поэтому можно сложить квадраты этих величин и выбрать $k$ так, чтобы эта сумма

$$E = \sum_{i=1}^{n} (Y_i - y_i)^2 = (Y_1 - y_1)^2 + (Y_2 - y_2)^2 + ... + (Y_n - y_n)^2$$

была минимальной. Такой подход называется *методом наименьших квадратов*.

Как найти коэффициент $k$ наилучшим образом, то есть так, чтобы сумма квадратов $E$ была минимальной? Для этого заменим $Y_i$ на $k \cdot x_i$ и раскроем скобки в формуле для вычисления $E$:

$$(Y_i - y_i)^2 = (k \cdot x_i - y_i)^2 = k^2 x_i^2 - 2k x_i y_i + y_i^2 .$$

Группируя слагаемые, содержащие $k^2$ и $k$, получаем

$$E = Ak^2 - Bk + C ,$$

где $A = \sum_{i=1}^{n} x_i^2$, $B = 2\sum_{i=1}^{n} x_i y_i$ и $C = \sum_{i=1}^{n} y_i^2$. График зависимости $E$ от $k$ – это парабола, причем ее ветви направлены вверх, потому что $A > 0$ (это сумма квадратов). Вершина параболы (и минимум функции!) находится в точке $k = \frac{B}{2A}$, это и будет оптимальное решение.

Таким образом, алгоритм для определения оптимального значения $k$ приобретает вид:

1) вычислить коэффициенты параболы $A = \sum_{i=1}^{n} x_i^2$ и $B = 2\sum_{i=1}^{n} x_i y_i$;

2) вычислить $k = \frac{B}{2A}$.

Решение получилось простым только потому, что мы выбрали очень простую функцию, линейную с нулевым свободным членом. В более сложных случаях строгое решение задачи оптимизации требует знания высшей математики.

Если исходные данные записаны в массивы **x[1..N]** и **y[1..N]**, программа на Паскале для рассмотренного случая выглядит так:

```
A:= 0; B:= 0;
нц для i от 1 до N
  A := A + x[i]*x[i];
  B := B + x[i]*y[i];
кц
k:= B / A;
```

```
A:= 0; B:= 0;
for i:=1 to N do begin
  A:= A + x[i]*x[i];
  B:= B + x[i]*y[i];
end;
k:= B / A;
```

Чтобы избавиться от лишних операций, умножение на 2 при вычислении B и деление на 2 при вычислении $k$ не выполняется (они взаимно уничтожаются).

Для решения задачи методом наименьших квадратов можно использовать табличные процессоры с модулем поиска решения. Пусть в результате измерений получены точки (1; 1,1), (2; 1,8) и (3; 3,5). Занесем эти координаты в столбцы A и B, в ячейку B1 запишем начальное приближение для $k$, а в столбец C – значения функции $y = k \cdot x$ для значений $x$ из столбца A:

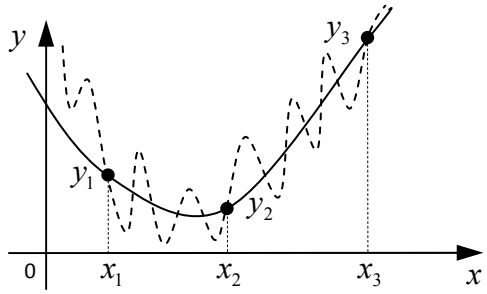
| | A | B | C |
|---|---|---|---|
| 1 | k | 1,000 | |
| 2 | E | =SUMXMY2(B5:B7;C5:C7) | |
| 3 | | | |
| 4 | x | y | Y |
| 5 | 1 | 1,1 | =$B$1*A5 |
| 6 | 2 | 1,8 | =$B$1*A6 |
| 7 | 3 | 3,5 | =$B$1*A7 |

Величина $E$ – это сумма квадратов разностей двух рядов, которая вычисляется с помощью функции **SUMXMY2** (**СУММКВРАЗН**). У этой функции 2 аргумента – ряд $y$ (измеренные значения в столбце B) и ряд $Y$ (вычисленные значения функции в столбце C). Задача оптимизации – найти минимальное значение $E$ (в ячейке B2), изменяя значение $k$ в ячейке B1 – решается с помощью надстройки «Поиск решения».

## 9.6.3. Восстановление зависимостей

Пусть заданы пары значений $x$ и $y$, и предполагается, что они связаны некоторой зависимостью $y = f(x)$, которую нужно найти. Такая задача называется задачей *восстановления зависимости*.

Если вид функции не задан, эта задача *некорректна*, потому что через заданные точки можно провести сколько угодно различных линий (графиков функций), и невозможно сказать, какая из них лучше подходит:

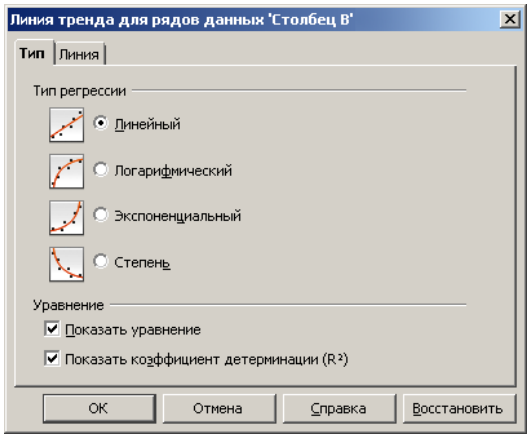



Поэтому для того, чтобы сделать задачу осмысленной нужно заранее задать *вид функции*, так что останется только найти ее неизвестные коэффициенты.

Откуда взять вид функции? В некоторых случаях он известен из физических законов, описывающих явление (так было в примере с исследованием закона Гука). Иногда вид зависимости можно определить по внешнему виду расположения точек. Также можно попробовать функции разного типа и выбрать лучший вариант. Часто используют следующие типы функций:

- *линейную* $y = a \cdot x + b$;
- *логарифмическую* $y = a \cdot \ln x + b$;
- *показательную* (экспоненциальную) $y = a \cdot b^x$;
- *степенную* $y = a \cdot x^b$.

Задача сводится к тому, чтобы выбрать коэффициенты $a$ и $b$ наилучшим образом. Для ее решения «вручную» нужно применять методы вычислительной математики, выходящие за рамки школьного курса. Однако в современных табличных процессорах есть встроенные возможности для решения задачи восстановления зависимостей. Полученные графики оптимальных функций называются *линиями тренда* (англ. *trend* – основное направление развития).




Сначала нужно ввести исходные данные (координаты точек) в таблицу и построить по ним диаграмму типа «Диаграмма XY» (в *Excel* – «Точечная»). Лучше оставить на диаграмме только точки, не соединяя их линией.

Для того, чтобы построить линию тренда, надо щелкнуть правой кнопкой мыши на одной из точек и выбрать пункт «Вставить линию тренда» из

контекстного меню. В появившемся окне можно выбрать вид зависимости. Если установить флажок «Показать уравнение», уравнение линии тренда будет показано на диаграмме. Флажок «Показать коэффициент детерминации ($R^2$)» позволяет увидеть, насколько точно полученная линия соответствует исходным данным. Коэффициент $R^2$ вычисляется по формуле:

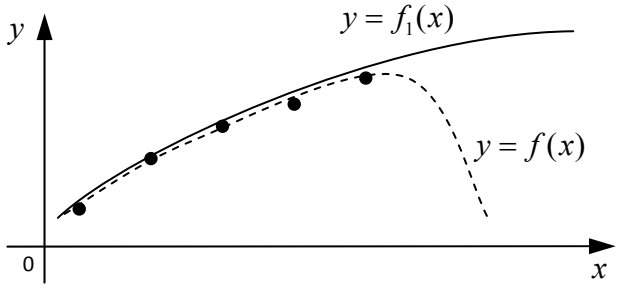
$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - Y_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2},$$

где через $\bar{y}$ обозначено среднее значение ряда $y$. Этот коэффициент всегда не больше 1 (подумайте, почему), чем он больше, тем лучше соответствие. В лучшем случае $R^2 = 1$, при этом $y_i = Y_i$ для всех $i$, то есть все значения функции совпадают с заданными.

Из приведенной формулы видно, что $R^2$ имеет наибольшее значение, когда сумма квадратов отклонений $E = \sum_{i=1}^{n}(y_i - Y_i)^2$ минимальна. Это значит, что задача поиска максимума $R^2$ решается методом наименьших квадратов. Если нужно найти неизвестные коэффициенты функции, которая не входит в стандартный набор (например, $y = a \cdot \sin bx + c$) можно применить метод наименьших квадратов с помощью надстройки «Поиск решения».

## 9.6.4. Прогнозирование

Во многих задачах (например, в экономике) в результате обработки данных нужно сделать *прогноз* на будущее. Если найдена зависимость $y = f(x)$, эта задача решается просто – нужно найти значения функции для тех значений $x$, которые нас интересуют. Однако может получиться так, что функция, которая очень хорошо соответствует имеющимся данным (см. функцию $y = f(x)$ на рисунке), оказывается непригодна для прогноза:




В таких случаях для решения задачи прогнозирования нужно выбирать другую функцию, которая дает меньшее значение $R^2$, но лучше показывает закономерность изменения величины (например, возрастающий характер функции).

### ? *Контрольные вопросы*

1. Объясните, почему экспериментальные исследования требуют специальных методов обработки данных.
2. Объясните суть метода наименьших квадратов. Почему можно считать такой подход решением задачи оптимизации?

### *Задачи*

1. Результаты серии измерений, сделанных для определения жесткости пружины на основе закона Гука, записаны в таблицу:

| $F$, Н | 1 | 3 | 6 | 10 |
|---|---|---|---|---|
| $\Delta\ell$, м | 0,028 | 0,070 | 0,170 | 0,260 |

Используя метод наименьших квадратов, определите жесткость пружины. Решите задачу разными способами (с помощью собственной программы и табличного процессора), сравните результаты.
(Ответ: 37,76 Н/м)

2. *Используя те же данные, что и в предыдущей задаче, найдите жесткость пружины другим способом. Введем величину, обратную жесткости: $\eta = 1/k$, тогда $\Delta\ell = \eta \cdot F$. Сначала, применив метод наименьших квадратов, найдите $\eta$, а затем рассчитайте $k = 1/\eta$. Сравните результат с тем, что получилось в предыдущем задании. Объясните расхождение.
3. Почему задача восстановления зависимости некорректна, если не задан вид функции?
4. Доходы начинающей фирмы (в тысячах рублей) за первые 5 лет работы приведены в таблице:

| **Год** | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **Доход** | 93 | 187 | 270 | 321 | 350 |

С помощью табличного процессора найдите зависимость дохода от года работы (выберите лучший из стандартных вариантов, с наибольшим значением $R^2$). С помощью этой зависимости сделайте прогноз развития фирмы на 2 года вперед.
(Ответ: $y = 164{,}31 \cdot \ln x + 86{,}87$; прогноз: 381, 407)

5. *При изучении волн измерили отклонение уровня поверхности воды $y$ от «нулевого» уровня в одной точке в разные моменты времени $t$:

| $t$ | 0 | 0,2 | 0,4 | 0,6 | 0,8 | 1 |
|---|---|---|---|---|---|---|
| $y$ | 1 | 2 | 1,2 | -0,6 | -2 | -1,5 |

Предполагается, что волна описывается формулой $y = a \cdot \sin(b \cdot t + c)$, где $a$, $b$ и $c$ – некоторые числа. Определите неизвестные коэффициенты методом наименьших квадратов с помощью табличного процессора. В качестве начального приближения можно выбрать $a \approx 1$; $b \approx 4$; $c \approx 1$.
(Ответ: $a \approx 2{,}02$; $b \approx 4{,}92$; $c \approx 0{,}52$)

# Глава 10. Информационная безопасность

## 10.1. Основные понятия

Зависимость современных организаций от компьютерных технологий стала настолько сильной, что вывод из строя компьютерной сети или программного обеспечения может остановить работу предприятия. Чтобы этого не произошло, нужно соблюдать правила *информационной безопасности*.

**Информационная безопасность** — это защищенность информации от любых действий, в результате которых информация может быть искажена или утеряна, а владельцам или пользователям информации нанесен *недопустимый* ущерб.

Прежде всего, в защите нуждается государственная и военная тайна, коммерческая тайна, юридическая тайна, врачебная тайна. Необходимо защищать личную информацию: паспортные данные, данные о банковских счетах, логины и пароли на сайтах, а также любую информация, которую можно использовать для шантажа, вымогательства и т.п.

Конечно, невозможно защититься от любых потерь, поэтому задача состоит в том, чтобы исключить именно *недопустимый* ущерб. С точки зрения экономики, средства защиты не должны стоить больше, чем возможные потери.

*Защита информации* – это меры, направленные на то, чтобы не потерять информацию, не допустить ее искажения, а также не допустить, чтобы к ней получили доступ люди, не имеющие на это права. В результате нужно обеспечить

- *доступность* информации – возможность получения информации за приемлемое время;
- *целостность* (отсутствие искажений) информации;
- *конфиденциальность* информации (недоступность для посторонних).

*Доступность* информации нарушается, например, когда оборудование выходит из строя, или веб-сайт не отвечает на запросы пользователей в результате массовой атаки вредоносных программ через Интернет.

Нарушения *целостности* информации – это кража или искажение информации, например, подделка сообщений электронной почты и других цифровых документов.

*Конфиденциальность* нарушается, когда информация становится известной тем людям, которые не должны о ней знать (происходит перехват секретной информации).

В компьютерных сетях защищенность информации снижается в сравнении с отдельным компьютером, потому что

- в сети работает много пользователей, их состав меняется;
- есть возможность незаконного подключения к сети;
- существуют уязвимости в сетевом программном обеспечении;
- возможны атаки взломщиков и вредоносных программ через сеть.

В России вопросы, связанные с защитой информации, регулирует закон «Об информации, информационных технологиях и о защите информации».

**Технические средства** защиты информации – это замки, решетки на окнах, системы сигнализации и видеонаблюдения, другие устройства, которые блокируют возможные каналы утечки информации или позволяют их обнаружить.

**Программные средства** обеспечивают доступ к данным по паролю, шифрование информации, удаление временных файлов, защиту от вредоносных программ и др.

**Организационные средства** включают

- распределение помещений и прокладку линий связи таким образом, чтобы злоумышленнику было сложно до них добраться;
- политику безопасности организации.

Сервера, как правило, находятся в отдельном (охраняемом) помещении и доступны только администраторам сети. Важная информация должна периодически копироваться на резервные носители (диски или магнитную ленту), чтобы сохранить ее в случае сбоев. Обычные сотрудники (не администраторы)

- имеют право доступа только к тем данным, которые им нужны для работы;
- не имеют права устанавливать программное обеспечение;
- раз в месяц должны менять пароли.

Самое слабое звено любой системы защиты – это человек. Некоторые пользователи могут записывать пароли на видном месте (чтобы не забыть) и передавать их другим, при этом возможность незаконного доступа к информации значительно возрастает. Поэтому очень важно обучить пользователей основам информационной безопасности.

Большинство утечек информации связано с «инсайдерами» (англ. *inside* – внутри) – недобросовестными сотрудниками, работающими в фирме. Известны случаи утечки закрытой информации не через ответственных сотрудников, а через секретарей, уборщиц и другого вспомогательного персонала. Поэтому ни один человек не должен иметь возможности причинить непоправимый вред (в одиночку уничтожить, украсть или изменить данные, вывести из строя оборудование).

### ? *Контрольные вопросы*

1. Что такое информационная безопасность?
2. Что входит в понятие «защита информации»?
3. На какие группы делятся средства защиты информации?
4. Какие меры безопасности обычно применяются в организациях?
5. Почему при объединении компьютеров в сеть безопасность снижается?
6. Кто такие «инсайдеры»?

# 10.2.Вредоносные программы

## 10.2.1. Что такое компьютерный вирус?

**Компьютерный вирус** – это программа, способная создавать свои копии (необязательно совпадающие с оригиналом) и внедрять их в файлы и системные области компьютера. При этом копии могут распространяться дальше.

Как следует из этого определения, основная черта компьютерного вируса – это способность распространяться при запуске.

Вирус – это один из типов вредоносных программ. Однако очень часто вирусами называют любые вредоносные программы (англ. *malware*).

**Вредоносные программы** — это программы, предназначенные для незаконного доступа к информации, для скрытого использования компьютера или для нарушения работы компьютера и компьютерных сетей.

Зачем пишут такие программы? Во-первых, с их помощью можно получить управление компьютером пользователя и использовать его в своих целях. Например, через зараженный компьютер злоумышленник может взламывать сайты и переводить на свой счет незаконно полученные

деньги. Некоторые программы блокируют компьютер и для продолжения работы требуют отправить платное SMS-сообщение.

Зараженные компьютеры, подключенные к сети Интернет, могут объединяться в сеть специального типа – *ботнет* (от англ. *robot* – робот и *network* – сеть). Такая сеть часто состоит из сотен тысяч компьютеров, обладающих в сумме огромной вычислительной мощностью. По команде «хозяина» ботнет может организовать атаку на какой-то сайт. В результате огромного количества запросов сервер не справляется с нагрузкой, сайт становится недоступен, и бизнесмены несут большие денежные потери. Такая атака называется *DoS*-атакой[1] (англ. *DoS* = *Denial of Service*, отказ в обслуживании). Кроме того, ботнеты могут использоваться для подбора паролей, рассылки *спама* (рекламных электронных сообщений) и другой незаконной деятельности.

Во-вторых, некоторые вредоносные программы предназначены для шпионажа – передачи по Интернету секретной информации с вашего компьютера: паролей доступа к сайтам, почтовым ящикам, учетным записям в социальных сетях, банковским счетам и электронным платежным системам. В результате таких краж пользователи теряют не только данные, но и деньги.

В-третьих, иногда вирусы пишутся ради самоутверждения программистами, которые по каким-то причинам не смогли применить свои знания для создания полезного ПО. Такие программы нарушают нормальную работу компьютера: время от времени перезагружают его, вызывают сбои в работе операционной системы и прикладных программ, уничтожают данные.

Наконец, существуют вирусы, написанные ради шутки. Они не портят данные, но приводят к появлению звуковых или зрительных эффектов (проигрывание мелодии; искажение изображения на экране; кнопки, убегающие от курсора и т.п.).

Создание и распространение компьютерных вирусов и вредоносных программ – это уголовные преступление, которое предусматривает (в особо тяжких случаях) наказание до 7 лет лишения свободы (Уголовный кодекс РФ, статья 273).

**Признаки заражения** вирусом:

- замедление работы компьютера;
- уменьшение объема свободной оперативной памяти;
- зависание, перезагрузка или блокировка компьютера;
- ошибки при работе ОС или прикладных программ;
- изменение длины файлов, появление новых файлов (в том числе «скрытых»);
- рассылка сообщений по электронной почте без ведома автора.

Для того, чтобы вирус смог выполнить какие-то действия, он должен оказаться в памяти в виде программного кода и получить управление компьютером.

Поэтому вирусы **заражают** не любые данные, а только программный код, который может выполняться. Например:

- исполняемые программы (с расширениями **`.exe`**, **`.com`**);
- загрузочные сектора дисков;
- пакетные командные файлы (**`.bat`**);
- драйверы устройств;
- библиотеки динамической загрузки (**`.dll`**), функции из которых вызываются из прикладных программ;

[1] Здесь речь идет, строго говоря, о распределенной *DoS*-атаке (англ. *DDoS* = *Distributed DoS*), которая проводится сразу со многих компьютеров.

- документы, которые могут содержать *макросы* – небольшие программы, выполняющиеся при нажатии на клавиши или выборе пункта меню; например, макросы нередко используются в документах пакета *Microsoft Office*;
- веб-страницы (в них можно внедрить программу-*скрипт*, которая выполнится при просмотре страницы на компьютере пользователя).

В отличие от кода программ, файлы с данными (например, тексты, рисунки, звуковые и видеофайлы) только обрабатываются, но не выполняются, поэтому заложенный в них код никогда не должен получить управление компьютером. Однако из-за ошибок в программном обеспечении может случиться так, что специально подобранные некорректные данные вызовут сбой программы обработки и выполнение вредоносного кода[2]. Таким образом, существует некоторый шанс, что вредоносная программа, «зашитая» в рисунок или видеофайл, все-таки запустится.

Сейчас существуют два основных источника заражения вредоносными программами – флэш-диски и компьютерные сети. Компьютер может быть **заражен** при:

- запуске зараженного файла;
- загрузке с зараженного CD(DVD)-диска или флэш-диска;
- автозапуске зараженного CD(DVD)-диска или флэш-диска (вирус автоматически запускается из файла **`autorun.inf`** в корневом каталоге диска);
- открытии зараженного документа с макросами;
- открытии сообщения электронной почты с вирусом или запуске зараженной программы, полученной в приложении к сообщению;
- открытии веб-страницы с вирусом;
- установке активного содержимого для просмотра веб-страницы.

Кроме того, есть вирусы-черви, которые распространяются по компьютерным сетям без участия человека. Они могут заразить компьютер даже тогда, когда пользователь не сделал никаких ошибочных действий.

## 10.2.2. Типы вредоносных программ

К вредоносным программам относятся компьютерные вирусы, черви, троянские программы и др. По «среде обитания» обычно выделяют следующие типы вирусов:

- **файловые** – внедряются в исполняемые файлы, системные библиотеки и т.п.;
- **загрузочные** – внедряются в загрузочный сектор диска или в главную загрузочную запись винчестера (англ. *MBR = Master Boot Record*); опасны тем, что загружаются в память раньше, чем ОС и антивирусные программы;
- **макровирусы** – поражают документы, в которых могут быть макросы;
- **скриптовые вирусы** – внедряются в командные файлы или в веб-страницы (записывая в них код на языке *VBScript* или *JavaScript*);
- **сетевые вирусы** – распространяются по компьютерным сетям.

Некоторые вирусы при создании новой копии немного меняют свой код, для того чтобы их было труднее обнаружить. Такие вирусы называют «полиморфными» (от греч. *πολυ* — много, *μορφη* — форма, внешний вид).

**Червь** — это вредоносная программа, которая распространяется по компьютерным сетям. Наиболее опасны *сетевые черви*, которые используют «дыры» (ошибки в защите, уязвимости) операционных систем и распространяются очень быстро без участия человека. Червь посылает по

[2] В 2002 г. был обнаружен вирус, которые внедрялся в рисунки формата JPEG. Однако он получал управление только из-за ошибки в системной библиотеке *Windows*, которая была быстро исправлена.

сети специальный пакет данных (*эксплойт*, от англ. *exploit* – эксплуатировать), который позволяет выполнить код на удаленном компьютере и внедриться в систему.

Как правило, вскоре после обнаружения уязвимости выпускается обновление программного обеспечения («заплатка», «патч»); если его установить, то червь становится неопасен. К сожалению, системные администраторы не всегда вовремя устанавливают обновления. Это приводит к эпидемиям сетевых червей, которые по статистике вызывают наибольшее число заражений[3]. Зараженные компьютеры используются для рассылки *спама* или массовых *DoS*-атак на сайты в Интернете.

*Почтовые черви* распространяются как приложения к сообщениям электронной почты. Они представляют собой программы, которые при запуске заражают компьютер и рассылают свои копии по всем адресам из адресной книги пользователя. Из-за этой опасности многие почтовые серверы (например, *mail.google.com*) не разрешают пересылку исполняемых файлов.

Чтобы заставить пользователя запустить червя, применяются методы *социальной инженерии*: текст сообщения составляется так, чтобы заинтересовать человека и спровоцировать его на запуск программы, приложенной к письму. В некоторых случаях программа-вирус упакована в архив и защищена паролем, но находится немало людей, которые распаковывают его (пароль указывается в письме) и запускают программу. Часто в почтовых сообщениях содержится только ссылка на сайт, содержащий вирус.

Иногда файл, пришедший как приложение к письму, имеет двойное расширение, например,

«**СуперКартинка.jpg .exe**»

В самом деле, это программа (расширение имени файла **.exe**), но пользователь может увидеть только первые две части имени и попытаться открыть такой «рисунок».

Существуют черви, которые могут распространяться через файлообменные сети, чаты и системы мгновенных сообщений (например, ICQ), но они мало распространены.

Еще одна группа вредоносных программ – **троянские программы** или «троянцы» (трояны). «Троянский конь» — это огромный деревянный конь, которого древние греки подарили жителям Трои во время Троянской войны. Внутри него спрятались воины, которые ночью выбрались, перебили охрану и открыли ворота города. Троянские программы проникают на компьютер под видом «полезных» программ, например, кодеков для просмотра видео или экранных заставок (которые включаются, если некоторое время не работать на компьютере). В отличие от вирусов и червей, они не могут распространяться самостоятельно и часто «путешествуют» вместе с червями. Среди «троянцев» встречаются

- *клавиатурные шпионы* – передают «хозяину» все данные, вводимые с клавиатуры (в том числе коды доступа к банковским счетам и т.п.);
- *похитители паролей* – передают пароли, запомненные, например, в браузерах;
- *утилиты удаленного управления* – позволяют злоумышленнику управлять компьютером через Интернет (например, загружать и запускать любые файлы);
- *логические бомбы* – при определенных условиях (дата, время, команда по сети) уничтожают информацию на дисках.

Большинство существующих вирусов написано для ОС *Windows*, которая установлена более чем на 90% персональных компьютеров.

Известны также вирусы для *Mac OS* X и *Linux*, но не каждому удается их запустить. Дело в том, что обычный пользователь (не администратор) в этих операционных системах не имеет права на изменение системных файлов, поэтому *Mac OS* X и *Linux* считают защищенными от вирусов.

[3] http://www.securelist.com/ru/analysis

Кроме того, вирусы часто полагаются на то, что системные функции размещаются в памяти по определенным адресам. При сборке ядра *Linux* из исходных кодов эти адреса могут меняться, поэтому вирус, работающий на одном дистрибутиве, может не работать на других.

### *Контрольные вопросы*

1. Что такое компьютерный вирус? Чем он отличается от других программ?
2. Что такое вредоносные программы? Какие вредоносные программы вы знаете?
3. Перечислите признаки заражения компьютера вирусом.
4. Какие вредные действия могут совершать вредоносные программы?
5. Какие объекты могут быть заражены вирусами?
6. Какие объекты не заражаются вирусами?
7. При каких действиях пользователя возможно заражение вирусом?
8. Является ли создание и распространение вирусов уголовным преступлением?
9. Какие типы вирусов вы знаете?
10. Что означает сокращение «MBR»?
11. Чем опасны загрузочные вирусы?
12. Что такое макровирусы? Какие файлы они поражают?
13. Что могут заражать скриптовые вирусы?
14. Что такое полиморфные вирусы? Почему их сложно обнаруживать?
15. Что такое сетевой червь?
16. Что такое эксплойт?
17. Почему необходимо сразу устанавливать обновления для операционных систем?
18. С какими целями могут быть использованы компьютеры, зараженные сетевым червем?
19. Почему многие почтовые сервера запрещают пересылку исполняемых файлов?
20. Что такое социальная инженерия? Как она используется авторами вирусов?
21. Что такое троянские программы? Какие типы троянских программ вы знаете?
22. Какие операционные системы лучше защищены от вирусов? Почему?

# 10.3.Защита от вредоносных программ

## 10.3.1. Антивирусные программы

**Антивирус** – это программа, предназначенная для борьбы с вредоносными программами.

Антивирусы выполняют три основные задачи:

1) не допустить заражения компьютера вирусом;
2) обнаружить присутствие вируса в системе;
3) удалить вирус без ущерба для остальных данных.

Код большинства вирусов содержит характерные цепочки байт – *сигнатуры* (от лат. *signare* – «подписать»). Если в файле обнаруживается сигнатура какого-то вируса, можно предположить, что файл заражен. Такой подход используется всеми антивирусными программами. Сигнатуры известных вирусов хранятся в базе данных антивируса, которую нужно регулярно обновлять через Интернет.

Современные антивирусы – это программные комплексы, состоящие из нескольких программ. Чаще всего они включают антивирус-сканер (иногда его называют антивирус-доктор) и антивирус-монитор.

Для того, чтобы **антивирус-сканер** начал работу, пользователь должен его запустить и указать, какие файлы и папки нужно проверить. Это «защита по требованию». Сканеры используют два основных метода поиска вирусов:

- *поиск в файлах сигнатур вирусов*, которые есть в базе данных; после обнаружения файл с вирусом можно вылечить, а если это не получилось – удалить;
- *эвристический анализ* (греч. *ευρηκα* – «нашёл!»), при котором программа ищет в файле код, похожий на вирус.

Эвристический анализ часто позволяет обнаруживать полиморфные вирусы (изменяющие код с каждым новым заражением), но не гарантирует это. Кроме того, случаются ложные срабатывания, когда «чистый» файл попадает под подозрение.

Главный недостаток сканеров состоит в том, что они не могут предотвратить заражение компьютера, потому что начинают работать только при ручном запуске.

**Антивирусы-мониторы** – это программы постоянной защиты, они находятся в памяти в активном состоянии. Их основная задача – не допустить заражения компьютера и получения зараженных файлов извне. Для этого мониторы

- проверяют «на лету» все файлы, которые копируются, перемещаются или открываются в различных прикладных программах;
- проверяют используемые дискеты и флэш-диски;
- перехватывают действия, характерные для вирусов (форматирование диска, замена и изменение системных файлов) и блокируют их;
- проверяют весь поток данных, поступающий из Интернета (сообщения электронной почты, веб-страницы, сообщения ICQ).

Мониторы ведут непрерывное наблюдение, блокируют вирус в момент заражения. Иногда они могут перехватить и неизвестный вирус (сигнатуры которого нет в базе), обнаружив его подозрительные действия.

Главный недостаток антивирусов-мониторов – значительное замедление работы системы, особенно на маломощных компьютерах. Кроме того, мониторы фактически встраиваются в операционную систему, поэтому ошибки разработчиков антивируса могут привести к печальным последствиям (вплоть до вывода ОС из строя). Бывает и так, что при запущенном мониторе некоторые программы работают неправильно или вообще не работают. Тем не менее, не рекомендуется отключать монитор, особенно если вы работаете в Интернете или переносите файлы с помощью флэш-дисков.

Кроме вредоносных программ, современные антивирусы частично защищают компьютер от

- *фишинга* – выманивания паролей для доступа на сайты Интернета с помощью специально сделанных веб-страниц, которые внешне выглядят так же, как «официальные» сайты;
- *рекламных баннеров и всплывающих окон* на веб-страницах;
- *спама* – рассылки нежелательных рекламных сообщений по электронной почте.

Большинство антивирусных программ – условно-бесплатные (*shareware*), пробные версии с ограниченным сроком действия можно свободно загрузить из Интернета. Наиболее известны антивирусы *AVP* (*www.kaspersky.ru*), *DrWeb* (*www.drweb.com*), *Nod32* (*www.eset.com*), *McAfee* (*home.mcafee.com*).

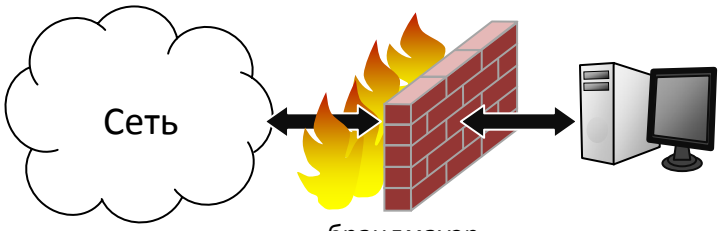
На многих сайтах (*www.kaspersky.ru*, *www.freedrweb.com*) доступны для скачивания лечащие программы-сканеры, которые бесплатны для использования на домашних компьютерах. В отличие от полных версий, в них нет антивируса-монитора, и базы сигнатур не обновляются.

Существуют бесплатные антивирусы, например, *Microsoft Security Essentials* (*www.microsoft.com*), *Avast Home* (*www.avast.com*), *Antivir Personal* (*free-av.com*), *AVG Free* (*free.grisoft.com*). Антивирус *ClamAV* (*www.clamav.net*) – бесплатный и поставляется с исходным кодом.

На сайтах некоторых компаний можно найти *онлайновые антивирусы* (например, *http://www.kaspersky.ru/virusscanner*). Они устанавливают на компьютер специальный сканирующий модуль и проверяют файлы и оперативную память. Как правило, онлайновые антивирусы могут обнаружить вирусы, но не удаляют их, предлагая приобрести коммерческую версию.

## 10.3.2. Брандмауэры

Для защиты отдельных компьютеров и сетей от атак из Интернета (в том числе и вирусных) используются **брандмауэры** (нем. *Brandmauer* – стена между зданиями для защиты от распространения огня). Их также называют «сетевые экраны» или «фаейрволы» (от англ. *firewall*). Брандмауэры запрещают передачу данных по каналам связи, которые часто используют вирусы и программы для взлома сетей.



брандмауэр

На рисунке показана защита одного компьютера с помощью брандмауэра, точно так же защищаются от угроз из Интернета локальные сети.

Брандмауэр входит в состав современных версий ОС *Windows*, в ядро *Linux* также включен встроенный брандмауэр *Netfilter*. Иногда устанавливают дополнительные брандмауэры, например, *Agnitum Outpost* (*www.agnitum.com*), *Kerio Winroute Firewall* (*kerio.ru*) или бесплатную программу *Comodo Personal Firewall* (*www.personalfirewall.comodo.com*).

## 10.3.3. Меры безопасности

Главные вред, который могут нанести вредоносные программы – это потеря данных или паролей доступа к закрытой информации.

Чтобы уменьшить возможный ущерб, рекомендуется регулярно делать резервные копии важных данных на CD(DVD)-дисках или флэш-дисках.

Если вы работаете в сети, желательно включать антивирус-монитор и брандмауэр. Монитор также сразу сообщит об опасности, если вставленный флэш-диск содержит вирус. Все новые файлы (особенно программы!) нужно проверять с помощью антивируса-сканера.

Не рекомендуется открывать подозрительные сообщения электронной почты, полученные с неизвестных адресов, особенно файлы-приложения (помните про методы социальной инженерии – заинтересовать жертву и заставить запустить программу). Опасно также переходить по ссылкам в тексте писем, с большой вероятностью они ведут на сайты, зараженные вирусами.

Если компьютер заражен, нужно отключить его от сети и запустить антивирус-сканер. Очень часто это позволяет удалить вирус, если его сигнатура есть в базе данных. Если антивирус не был установлен раньше, можно попробовать установить его на зараженный компьютер, но это не всегда приводит к успеху (вирус может блокировать установку антивируса).

Если антивирус-сканер не обнаруживает вирус или не может его удалить, можно попытаться (желательно с другого компьютера) найти в Интернете бесплатную утилиту для лечения с новыми базами сигнатур. Например, утилита *CureIt* (*www.freedrweb.com*) не требует установки и может быть запущена с флэш-диска. Даже если удалить вирус не удалось, скорее всего, он будет обнаружен, и программа покажет его название. Следующий шаг – искать в Интернете утилиту для удаления именно этого вируса (например, ряд утилит можно найти на сайте *support.kaspersky.ru*).

В особо тяжелых случаях для уничтожения вирусов приходится полностью форматировать жесткий диск компьютера, при этом все данные теряются.

### ? *Контрольные вопросы*

1. Что такое антивирус? Какие задачи он решает?
2. Что такое сигнатура?
3. Почему нужно регулярно обновлять базы сигнатур антивирусов?
4. Чем отличается антивирус-сканер от антивируса-монитора?
5. Что значит «защита по требованию»?
6. Что такое эвристический анализ? В чем его достоинства и недостатки?
7. Что делает антивирус-монитор? Каковы его недостатки?
8. Что такое фишинг?
9. Что такое спам?
10. Какие ограничения есть у пробных версий коммерческих антивирусов?
11. Что такое онлайновый антивирус?
12. Что такое брандмауэр? Зачем он нужен?
13. В чем заключается основной вред, наносимый вирусами? Как можно уменьшить возможные потери?
14. Как можно улучшить безопасность компьютера при работе в сети Интернет?
15. Какие меры безопасности необходимы при работе с электронной почтой?
16. Какие действия можно предпринять, если компьютер заражен вирусом?

## 10.4. Что такое шифрование?

Один из методов защиты информации от неправомерного доступа – это *шифрование*, то есть кодирование специального вида.

**Шифрование** – это преобразование (кодирование) открытой информации в зашифрованную, недоступную для понимания посторонних.

Шифрование применяется, в первую очередь, для передачи секретной информации по незащищённым каналам связи. Шифровать можно любую информацию – тексты, рисунки, звук, базы данных и т.д.

Человечество применяет шифрование с того момента, как появилась секретная информация, которую нужно было скрыть от врагов. Первый известное науке шифрованное сообщение – египетский текст, в котором вместо принятых тогда иероглифов были использованы другие знаки.

Методы шифрования и расшифровывания сообщения изучает наука *криптология*, история которой насчитывает около четырех тысяч лет. Она состоит из двух ветвей: *криптографии* и *криптоанализа*.

**Криптография** – это наука о способах шифрования информации.
**Криптоанализ** – это наука о методах и способах вскрытия шифров.

Обычно предполагается, что сам алгоритм шифрования известен всем, но неизвестен его *ключ*, без которого сообщение невозможно расшифровать. В этом заключается отличие шифрования от простого кодирования, при котором для восстановления сообщения достаточно знать только алгоритм кодирования.

**Ключ** – это параметр алгоритма шифрования (шифра), позволяющий выбрать одно конкретное преобразование из всех вариантов, предусмотренных алгоритмом. Знание ключа позволяет свободно зашифровывать и расшифровывать сообщения.

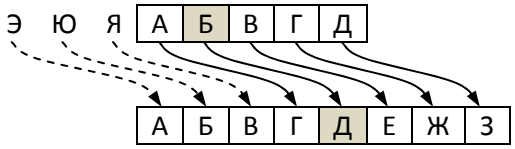
Все шифры (системы шифрования) делятся на две группы – *симметричные* и *несимметричные* (с открытым ключом).

**Симметричный шифр** означает, что и для шифрования, и для расшифровывания сообщений используется один и тот же ключ. В системах **с открытым ключом** используются два ключа – открытый и закрытый, которые связаны друг с другом с помощью некоторых математических зависимостей. Информация шифруется с помощью открытого ключа, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

**Криптостойкость шифра** – это устойчивость шифра к расшифровке без знания ключа.

Стойким считается алгоритм, который для успешного раскрытия требует от противника недостижимых вычислительных ресурсов, недостижимого объёма перехваченных сообщений, или такого времени, что по его истечению защищённая информация будет уже не актуальна.

Шифр Цезаря[4] – один из самых известных и самых древних шифров. В этом шифре каждая буква заменяется на другую, расположенную в алфавите на заданное число позиций $k$ вправо от нее. Алфавит замыкается в кольцо, так что последние символы заменяются на первые. Вот пример шифра Цезаря (со сдвигом 3):




Знаменитая фраза «ПРИШЕЛ УВИДЕЛ ПОБЕДИЛ» при использовании шифра Цезаря со сдвигом 3 будет закодирована так:

ТУЛЫИО ЦЕЛЗИО ТСДИЗЛО

Если первая буква алфавита имеет код 0, вторая – код 1 и т.д., алгоритм шифрования может быть выражен формулой

$$y = (x + k) \mod n,$$

где $x$ – код исходного символа, $k$ – величина сдвига, $y$ – код символа-замены, $n$ – количество символов в алфавите, а запись $(x + k) \mod n$ обозначает остаток от деления $x + k$ на $n$. Операция взятия остатка от деления необходима для того, чтобы «замкнуть» алфавит в кольцо. Например, при использовании русского алфавита (32 буквы[5]) для буквы «Я» (код 31) получаем код заменяющего символа $(31 + 3) \mod 32 = 2$, это буква «В».

Ключом для шифра Цезаря служит сдвиг $k$, если его знать, то сообщение легко расшифровать. Для этого используется формула

$$x = (y - k + n) \mod n.$$

Шифр Цезаря относится к шифрам *простой подстановки*, так как каждый символ исходного сообщения заменяется на другой символ из того же алфавита. Такие шифры легко раскрываются с

[4] Назван в честь римского императора Гая Юлия Цезаря, использовавшего его для секретной переписки.

[5] Будем считать, что буквы Е и Ё совпадают.

помощью частотного анализа, потому что в каждом языке частоты встречаемости букв примерно постоянны для любого достаточно большого текста.

Значительно сложнее сломать *шифр Виженера*[6], который стал естественным развитием шифра Цезаря. Для использования шифра Виженера используется ключевое слово, которое задает переменную величину сдвига. Например, пусть ключевое слово – «ЗАБЕГ». По таблице определяем коды букв:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|---|
| А | Б | В | Г | Д | Е | Ж | З | И | К | … |

Получаем: «З» – 7, «А» – 0, «Б» – 1, «Е» – 5, «Г» – 3. Это значит, что для кодирования первой буквы используется сдвиг 7, для кодирования второй – 0 (символ не меняется) и т.д. Для пятой буквы используется сдвиг 3, а для шестой – снова 7 (начали «проходить» кодовое слова с начала). Фраза «ПРИШЕЛ УВИДЕЛ ПОБЕДИЛ» при использовании шифра Виженера с ключом «ЗАБЕГ» будет закодирована в виде «ЦРЙЭИТ ФЗЛЛЕМ ТХБЖЙЛТ».

Шифр Виженера обладает значительно более высокой криптостойкостью, чем шифр Цезаря. Это значит, что его труднее раскрыть – подобрать нужное ключевое слово. Теоретически, если длина ключа равна длине сообщения, и каждый ключ используется только один раз, шифр Виженера взломать невозможно.

### ? *Контрольные вопросы*

1. Чем отличаются понятия «шифрование» и «кодирование»?
2. Что такое ключ?
3. Как называется наука, изучающая методы шифрования?
4. Что такое симметричный шифр? Какая проблема возникает при использовании симметричного шифра, если участники переписки находятся в разных странах?
5. Что такое несимметричные шифры? На чем основана их надежность?
6. Что такое криптостойкость алгоритма? Какой алгоритм считается криптостойким?

### *Задачи*

1. Зашифруйте с помощью шифра Цезаря со сдвигом 6 высказывание «ЛЮДИ ОХОТНО ВЕРЯТ ТОМУ, ЧЕМУ ЖЕЛАЮТ ВЕРИТЬ».
   (Ответ: «СДКО ФЫФШУФ ИЛЦЕШ ШФТЩ, ЭЛТЩ МЛСЖДШ ИЛЦОШВ»).
2. Напишите программу, которая выполняет шифрование строки с помощью шифра Цезаря.
3. *Попытайтесь расшифровать сообщение, закодированное шифром Цезаря с неизвестным сдвигом: «ХШЖНПУТ ФКХКОЙКТ». Для этого можно написать программу. (Ответ: «РУБИКОН ПЕРЕЙДЕН», сдвиг 5)
4. Используя шифр Виженера с ключом «ЛЕНА», зашифруйте сообщение «НЕЛЬЗЯ ОБИЖАТЬ ГОСТЯ».
   (Ответ: «ШКШЬТД ОМНУАЭБ ГЩЦЯЯ»)
5. *Измените программу для шифрования Цезаря так, чтобы учитывать букву Ё.
6. *Измените программу для шифрования Цезаря так, чтобы учитывать и букву Ё, и пробел.

## 10.5. Хэширование и пароли

В современных информационных системах часто используется вход по паролю. Если при этом где-то хранить пароли всех пользователей, система становится очень ненадежной, потому что «утечка» паролей позволит сразу получить доступ к данным. С другой стороны, кажется, что пароли обязательно где-то нужно хранить, иначе пользователи не смогут войти в систему. Однако,

[6] Назван по имени Блеза Виженера, швейцарского дипломата XVI века.

это не совсем так. Можно хранить не пароли, а некоторые числа, полученные в результате обработки паролей. Простейший вариант – сумма кодов символов, входящих в пароль. Для пароля «A123» такая сумма равна

215 = 65 (код «A») + 49 (код «1») + 50 (код «2») + 51 (код «3»).

Фактически мы определили функцию $H(M)$, которая сообщение $M$ *любой* длины превращает в короткий код $m$ *заданной* длины. Такая функция называются *хэш-функцией* (от англ. *hash* – «мешанина», «крошить»), а само полученное число – *хэш-кодом*, *хэш-суммой* или просто *хэшем* исходной строки. Важно, что зная хэш-код, невозможно восстановить исходный пароль! В этом смысле хэширование – это *необратимое* шифрование.

Итак, вместо пароля «A123» мы храним число 215. Когда пользователь вводит пароль, мы считаем сумму кодов символов этого пароля и разрешаем вход в систему только тогда, когда она равна 215. И вот здесь возникает проблема: существует очень много паролей, для которых наша хэш-функция дает значение 215, например, «B023». Такая ситуация – совпадение хэш-кодов различных исходных строк – называется *коллизией* (англ. *collision* – «столкновение»). Коллизии будут всегда – ведь мы «сжимаем» длинную цепочку байт до числа. Казалось бы, ничего хорошего не получилось: если взломщик узнает хэш-код, то, зная алгоритм его получения, он сможет легко подобрать пароль с таким же хэшем и получить доступ к данным. Однако, это произошло потому, что мы выбрали плохую хэш-функцию.

Математики разработали надежные (но очень сложные) хэш-функции, обладающие особыми свойствами:

1) хэш-код очень сильно меняется при малейшем изменении исходных данных;
2) при известном хэш-коде $m$ невозможно за приемлемое время найти сообщение $M$ с таким хэш-кодом ( $H(M) = m$ );
3) при известном сообщении $M$ невозможно за приемлемое время найти сообщение $M_1$ с таким же хэш-кодом ( $H(M) = H(M_1)$ ).

Здесь выражение «невозможно за приемлемое время» (или «вычислительно невозможно») означает, что эта задача решается только перебором вариантов (других алгоритмов не существует), а количество вариантов настолько велико, что на решение уйдут сотни и тысячи лет. Поэтому даже если взломщик получил хэш-код пароля, он не сможет за приемлемое время получить сам пароль (или пароль, дающий такой же хэш-код).

Чем длиннее пароль, тем больше количество вариантов. Кроме длины, для надежности пароля важен используемый набор символов. Например, очень легко подбираются пароли, состоящие только из цифр. Если же пароль состоит из 10 символов и содержит латинские буквы (заглавные и строчные) и цифры, перебор вариантов (англ. *brute force* – метод «грубой силы») со скоростью 10 млн. паролей в секунду займет более 2000 лет.

Надежные пароли должны состоять не менее чем из 7-8 символов; пароли, состоящие из 15 символов и более взломать методом «грубой силы» практически невозможно. Нельзя использовать пароли типа «12345», «qwerty», свой день рождения, номер телефона. Плохо, если пароль представляет собой известное слово, для этих случаев взломщики используют подбор по словарю. Сложнее всего подобрать пароль, который представляет собой случайный набор заглавных и строчных букв, цифр и других знаков[7].

Сегодня для хэширования в большинстве случаев применяют алгоритмы MD5, SHA1 и российский алгоритм, изложенный в ГОСТ Р 34.11 94 (он считается одним из самых надежных). В криптографии хэш-коды чаще всего имеют длину 128, 160 и 256 бит.

[7] Однако такой пароль сложно запомнить.

Хэширование используется также для проверки правильности передачи данных. Различные контрольные суммы, используемые для проверки правильности передачи данных, – это не что иное, как хэш-коды.

### ? *Контрольные вопросы*

1. Что такое хэширование? хэш-функция? хэш-код?
2. Какую хэш-функцию вы используете, когда начинаете искать слово в словаре?
3. Что такое коллизии? Почему их должно быть как можно меньше?
4. Какие требования предъявляются к хэш-функциям, которые используются при хранении паролей?
5. Что значит «вычислительно невозможно»?
6. Взломщик узнал хэш-код пароля администратора сервера. Сможет ли он получить доступ к секретным данным на сервере?
7. Какие свойства пароля влияют на его надежность?
8. Как выбрать надежный пароль?
9. Какие алгоритмы хэширования сейчас чаще всего применяются?
10. *Предложите какой-нибудь свой метод хэширования. Подумайте, как часто при его использовании могут происходить коллизии.

## 10.6. Современные алгоритмы шифрования

Государственным стандартом шифрования в России является алгоритм, зарегистрированный как ГОСТ 28147-89. Он является *блочным* шифром, то есть шифрует не отдельные символы, а 64-битные блоки. В алгоритме предусмотрено 32 цикла преобразования данных с 256-битным ключом, за счет этого он очень надёжен (обладает высокой криптостойкостью). На современных компьютерах раскрытие этого шифра путем перебора ключей («методом грубой силы») займет не менее сотен лет, что делает такую атаку бессмысленной. В США используется аналогичный блочный шифр AES.

В Интернете популярен алгоритм RSA, названный так по начальным буквам фамилий его авторов – Р. Райвеста (R. Rivest), А. Шамира (A. Shamir) и Л. Адлемана (L. Adleman). Это алгоритм с *открытым* ключом, стойкость которого основанный на использовании свойств простых чисел. Для его взлома нужно разложить очень большое число на простые сомножители. Эту задачу сейчас умеют решать только перебором вариантов. Поскольку количество вариантов огромно, для раскрытия шифра требуется много лет работы современных компьютеров.

Для применения алгоритм RSA требуется построить открытый и секретный ключи следующим образом.

1. Выбрать два больших простых числа, $p$ и $q$.
2. Найти их произведение $n = p \cdot q$ и значение $\varphi = (p-1)\cdot(q-1)$.
3. Выбрать число $e$ ($1 < e < \varphi$), которое не имеет общих делителей с $\varphi$.
4. Найти число $d$, которое удовлетворяет условию $d \cdot e = k\varphi + 1$ для некоторого целого $k$.
5. Пара значений $(e,n)$ – это открытый ключ RSA (его можно свободно публиковать), а пара $(d,n)$ – это секретный ключ.

Передаваемое сообщение нужно сначала представить в виде последовательности чисел в интервале от 0 до $n-1$. Для шифрования используют формулу

$$y = x^e \bmod n,$$

где $x$ – число исходного сообщения, $(e,n)$ – открытый ключ, $y$ – число закодированного сообщения, а запись $x^e \bmod n$ обозначает остаток от деления $x^e$ на $n$. Расшифровка сообщения выполняется по формуле

$$x = y^d \bmod n.$$

Это значит, что зашифровать сообщение может каждый (открытый ключ общеизвестен), а прочитать его – только тот, кто знает секретный показатель степени $d$.

Для лучшего понимания мы покажем работу алгоритма RSA на простом примере. Возьмем $p = 3$ и $q = 7$, тогда находим $n = p \cdot q = 21$ и $\varphi = (p-1)\cdot(q-1) = 12$. Выберем $e = 5$, тогда равенство $d \cdot e = k\varphi + 1$ выполняется, например, при $d = 17$ (и $k = 7$). Таким образом, мы получили открытый ключ (5,21) и секретный ключ (17,21).

Зашифруем сообщение «123» с помощью открытого ключа (5,21). Получаем

$$1 \Rightarrow 1^5 \bmod 21 = 1, \quad 2 \Rightarrow 2^5 \bmod 21 = 11, \quad 3 \Rightarrow 3^5 \bmod 21 = 12,$$

то есть зашифрованное сообщение состоит из чисел 1, 11 и 12. Зная секретный ключ (17,21), можно его расшифровать:

$$1 \Rightarrow 1^{17} \bmod 21 = 1, \quad 11 \Rightarrow 11^{17} \bmod 21 = 2, \quad 12 \Rightarrow 12^{17} \bmod 21 = 3.$$

Мы получили исходное сообщение.

Конечно, вы заметили, что при шифровании и расшифровке приходится вычислять остаток от деления очень больших чисел (например, $12^{17}$) на $n$. Оказывается, само число $12^{17}$ в этом случае находить не нужно. Достаточно записать в обычную целочисленную переменную, например, $x$, единицу, а потом 17 раз выполнить преобразование $x = 12 \cdot x \bmod 21$. После этого в переменной x будет значение $12^{17} \bmod 21 = 3$. Попробуйте доказать правильность этого алгоритма.

Для того, чтобы расшифровать сообщение, нужно знать секретный показатель степени $d$. А для этого, в свою очередь, нужно найти сомножители $p$ и $q$, такие что $n = p \cdot q$. Если $n$ велико, это очень сложная задача, ее решение перебором вариантов на современном компьютере займет сотни лет. В 2009 году группа ученых из разных стран в результате многомесячных расчетов на сотнях компьютеров смогла расшифровать сообщение, зашифрованное алгоритмом RSA с 768-битным ключом. Поэтому сейчас надежными считаются ключи с длиной 1024 бита и более. Если будет построен работающий квантовый компьютер, взлом алгоритма RSA будет возможен за очень небольшое время.

При использовании симметричных шифров всегда возникает проблема: как передать ключ, если канал связи ненадежный? Ведь получив ключ, противник сможет расшифровать все дальнейшие сообщения. Для алгоритма RSA этой проблемы нет, сторонам достаточно обменяться открытыми ключами, которые можно показывать всем желающим.

У алгоритма RSA есть еще одно достоинство: его можно использовать для цифровой подписи сообщений. Она служит для доказательства авторства документов, защиты сообщений от подделки и умышленных изменений.

**Цифровая подпись** – это набор символов, который получен в результате шифрования сообщения с помощью личного секретного кода отправителя.

Отправитель может передать вместе с исходным сообщением такое же сообщение, зашифрованное с помощью своего секретного ключа (это и есть цифровая подпись). Получатель расшифровывает цифровую подпись с помощью открытого ключа. Если она совпала с незашифрованным сообщением, можно быть уверенным, что его отправил тот человек, который знает секретный код. Если сообщение было изменено при передаче, оно не совпадет с расшифрованной цифровой подписью. Так как сообщение может быть очень длинным, для сокращения объема передаваемых данных чаще всего шифруется не всё сообщение, а только его хэш-код.

Во многих современных программах есть возможность шифровать данные с паролем. Например, офисные пакеты *OpenOffice.org* и *Microsoft Office* позволяют шифровать все создаваемые документы (для их просмотра и/или изменения нужно ввести пароль). При создании архива (на-

пример, в архиваторах 7ZIP, *WinRAR*, *WinZip*) также можно установить пароль, без которого извлечь файлы невозможно.

В простейших задачах для шифрования файлов можно использовать бесплатную программу *Шифровальщик* (*www.familytree.ru/ru/cipher.htm*), версии которой существуют для *Linux* и *Windows.* Программы *TrueCrypt* (*www.truecrypt.org*), *BestCrypt* (*www.jetico.com*) и *FreeOTFE* (*freeotfe.org*) создают логические диски-контейнеры, информация на которых шифруется. Свободно распространяемая программа *DiskCryptor* (*diskcryptor.net*) позволяет шифровать разделы жестких дисков и даже создавать шифрованные флэш-диски и CD/DVD диски.

Программа *GnuPG* (*gnupg.org*) также относится к свободному программному обеспечению. В ней поддерживаются симметричные и несимметричные шифры, а также различные алгоритмы электронной цифровой подписи.

### *Контрольные вопросы*

1. Какой алгоритм шифрования принят в России в качестве государственного стандарта?
2. Что такое блочный алгоритм шифрования?
3. К какому типу относится алгоритм RSA? На чем основана его криптостойкость?
4. Что такое цифровая подпись?
5. Как можно использовать алгоритм RSA для цифровой подписи?

### *Задачи*

1. *Напишите программу, которая строит открытый и секретный ключи RSA для небольших множителей $p$ и $q$.
2. *Напишите программу, которая шифрует и расшифровывает сообщения с помощью алгоритма при небольших значениях открытого и секретного ключей.

## 10.7. Стеганография

При передаче сообщений можно не только применять шифрование, но и скрывать сам факт передачи сообщения.

**Стеганография** – это наука о скрытой передаче информации путем скрытия самого факта передачи информации.

Древнегреческий историк Геродот описывал, например, такой метод: на бритую голову раба записывалось сообщение, а когда его волосы отрастали, он отправлялся к получателю, который брил его голову и читал сообщение.

Классический метод стеганографии – *симпатические* (невидимые) *чернила*, которые проявляются только при определенных условиях (нагрев, освещение, химический проявитель). Например, текст, написанный молоком, можно прочитать при нагреве.

Сейчас стеганография занимается скрытием информации в текстовых, графических, звуковых и видеофайлах с помощью программного «внедрения» в них нужных сообщений.

Простейшие способ – заменять младшие биты файла, в котором закодировано изображение. Причем это нужно сделать так, чтобы разница между исходным и полученным рисунками была неощутима для человека. Например, если в черно-белом рисунке (256 оттенков серого), яркость каждого пикселя кодируется 8 битами. Если поменять 1-2 младших бита этого кода, «встроив» туда текстовое сообщение, фотография, в которой нет четких границ, почти не изменится. При замене 1 бита каждый байт исходного текстового сообщения хранится в младших битах кодов 8 пикселей. Например, пусть первые 8 пикселей рисунка имеют такие коды:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 10101101 | 10010100 | 00101010 | 01010010 | 10101010 | 10101010 | 10101011 | 10101111 |

Чтобы закодировать в них код буквы «И» ($11001000_2$), нужно изменить младшие биты кодов:

| 10101101 | 10010101 | 00101010 | 01010010 | 10101011 | 10101010 | 10101010 | 10101110 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

Получателю нужно взять эти младшие биты и «собрать» их вместе в один байт.

Для звуков используются другие методы стеганографии, основанные на добавлении в запись коротких условных сигналов, которые обозначают 1 и 0 и не воспринимаются человеком на слух. Возможна также замена одного фрагмента звука на другой.

Для подтверждения авторства и охраны авторских прав на изображения, видео и звуковые файлы применяют *цифровые водяные знаки* – внедренную в файл информацию об авторе. Они получили свое название от старых водяных знаков на деньгах и документах. Для того чтобы установить авторство фотографии, достаточно расшифровать скрытую информацию, записанную с помощью водяного знака.


Иногда цифровые водяные знаки делают видимыми (текст или логотип компании на фотографии или на каждом кадре видеофильма). На многих сайтах, занимающихся продажей цифровых фотографий, видимые водяные знаки размещены на фотографиях, предназначенных для предварительного просмотра.

### ? *Контрольные вопросы*

1. Что такое стеганография?
2. Какие методы стеганографии существовали до изобретения компьютеров?
3. Как можно добавить текст в закодированное изображение?
4. На чем основаны методы стеганографии для звуковых и видеоданных?
5. Что такое цифровые водяные знаки? Зачем они используются?

# 10.8. Безопасность в Интернете

## 10.8.1. Угрозы безопасности

Если компьютер подключен к Интернету, появляются дополнительные угрозы безопасности. Атаку через сеть могут проводить злоумышленники и *боты* (программы-роботы), находящиеся в других городах и странах. Можно выделить три основные цели злоумышленников:

- *использование вашего компьютера* для взлома других компьютеров, атак на сайты, рассылки спама, подбора паролей и т.п.;
- *кража секретной информации* – данных о банковских картах, имен и паролей для входа на почтовые сервера, в социальные сети, платежные системы;
- *мошенничество* – хищение чужого имущества путем обмана.

Первые две угрозы связаны, главным образом, с вредоносными программами: вирусами, червями и «троянцами», которые позволяют злоумышленнику управлять компьютером через сеть и получать с него данные.

Мошенничество процветает потому, что многие пользователи Интернета очень доверчивы и неосторожны. Классический пример мошенничества – так называемые «нигерийские письма», приходящие по электронной почте. Пользователя от имени какого-то бывшего высокопоставленного лица просят принять участие в переводе крупных денежных сумм за границу, обещая выплачивать большие проценты. Если получатель соглашается, мошенники постепенно выманивают у него деньги.

*Фишинг* (англ. *phishing*, искажение слова *fishing* – рыбная ловля) – это выманивание паролей. Для этого чаще всего используются сообщения электронной почты, рассылаемые якобы от имени администраторов банков, платежных систем, почтовых служб, социальных сетей. В сообщении говорится, что ваш счёт (или учетная запись) заблокирован, и дается ссылка на сайт, который внешне выглядит как настоящий, но расположен по другому адресу (это можно проверить в адресной строке браузера). Неосторожный пользователь вводит своё кодовое имя и пароль, с помощью которых мошенник получает доступ к данным или банковскому счету.

Антивирусы и последние версии браузеров содержат специальные модули для обнаружения подозрительных сайтов («*антифишинг*») и предупреждают о заходе на такой сайт. Кроме того, нужно помнить, что администраторы сервисов никогда не просят пользователя сообщить свой пароль по электронной почте.

Мошенничество может быть связано и с вредоносными программами. В 2010 году несколько миллионов компьютеров в России было заражено троянской программой *Winlock*, которая блокировала компьютер и требовала отправить платное SMS-сообщение для снятия блокировки.

## 10.8.2. Правила личной безопасности

Вредоносные программы, распространяющиеся через Интернет, представляю серьезную угрозу безопасности данных. Нужно помнить, что многих проблем можно избежать, если работать в Интернете только из-под ограниченной учетной записи (без прав администратора). Кроме того, желательно своевременно обновлять программное обеспечение; особенно важно устанавливать «заплатки», связанные с безопасностью.

Чтобы ваши пароли не украли, лучше не запоминать их в браузере (иногда они хранятся в открытом виде и могут быть украдены троянской программой). Заходя под своим именем в закрытую зону сайта с другого компьютера, нужно отмечать флажок «*Чужой компьютер*», иначе следующий человек, открывший эту страницу, сможет получить доступ к вашим данным.

На многих сайтах предусмотрена возможность восстановления пароля по секретному вопросу. Этот вопрос нужно выбирать так, чтобы никто другой не знал ответа на него и, самое важное, не мог его выведать. Например, ответы на вопросы «Как звали Вашу первую собаку?», «Какое Ваше любимое блюдо?» и т.п. часто можно найти на персональных страничках авторов в социальных сетях (в заметках, подписях к фотографиям и т.п.). Если мама автора имеет свою страничку, на ней, скорее всего можно найти ее девичью фамилию, поэтому вопрос «Какова девичья фамилия вашей матери?» тоже лучше не использовать.

Нужно понимать, что размещая какую-то информацию в Интернете, вы делаете ее доступной для широкого круга лиц, включая работодателей, милицию, официальные органы и даже преступников. Возможны ситуации, когда эта информация (личные данные, фотографии, высказывания на форумах и в блогах) может быть использована против вас, даже если она находится в закрытом разделе сайта.

Для передачи информации, которую необходимо сохранить в тайне, лучше применять шифрование (например, упаковать данные в архив с паролем).

Наибольший уровень безопасности обеспечивается при денежных расчетах через Интернет: вместо протокола *HTTP* используют защищенный протокол *HTTPS* (англ. *Hypertext Transfer Protocol Secure* – безопасный *HTTP*), который предусматривает шифрование данных (например, с помощью алгоритма *RSA*). Поэтому нужно проверять, чтобы адрес на странице ввода пароля в таких системах начинался с «**`https://`**», а не с «**`http://`**».

Современные молодые люди часто общаются в чатах, форумах и т.п., в том числе с теми, кого они не знают лично. Продолжение такого *виртуального* (компьютерного, электронного) зна-

комства в реальной жизни весьма опасно, потому что нередко участники чатов и форумов представляются не теми, кем они являются на самом деле.

## ? *Контрольные вопросы*

1. Какие угрозы безопасности существуют при подключении к Интернету?
2. Какие схемы интернет-мошенничества вам известны?
3. Какие меры безопасности нужно соблюдать при работе в Интернете?
4. Как обеспечивается безопасность обмена данными при денежных расчетах в Интернете?