К. Ю. Поляков, Е. А. Еремин

ИНФОРМАТИКА

10 класс

(базовый и углублённый уровни)

(в 2 частях)

Учебник

Часть 1

Рекомендовано
к использованию при реализации имеющих государственную
аккредитацию образовательных программ начального общего,
основного общего, среднего общего образования

Москва
БИНОМ. Лаборатория знаний
2019

УДК 004.9
ББК 32.97
П54

Поляков К. Ю.

П54 Информатика (базовый и углублённый уровни) (в 2 частях). 10 класс. Ч. 1 : учебник / К. Ю. Поляков, Е. А. Еремин. — М. : БИНОМ. Лаборатория знаний, 2019. — 352 с. : ил.

ISBN 978-5-9963-4588-5 (Ч. 1)
ISBN 978-5-9963-4590-8

Учебник предназначен для изучения информатики на базовом и углублённом уровнях в 10 классах общеобразовательных организаций. Содержание учебника опирается на изученный в 7–9 классах курс информатики для основной школы.

Рассматриваются теоретические основы информатики, аппаратное и программное обеспечение компьютера, компьютерные сети, алгоритмизация и программирование, информационная безопасность.

Учебник входит в учебно-методический комплект, включающий также учебник для 11 класса, методическое пособие и задачник.

Соответствует федеральному государственному образовательному стандарту среднего общего образования и примерной основной образовательной программе среднего общего образования.

УДК 004.9
ББК 32.97

ISBN 978-5-9963-4588-5 (Ч. 1)
ISBN 978-5-9963-4590-8

От авторов

Вы держите в руках учебник информатики для 10 класса. Его можно использовать для изучения этого предмета как на базовом, так и на углублённом уровне. Базовый уровень — это только «верхний пласт» — тот необходимый минимум, который должен знать каждый современный человек. Но даже если вы изучаете информатику на базовом уровне, никто не запрещает вам заглянуть в другие разделы — вдруг там окажется что-то интересное.

Параграфы, которые изучаются только на углублённом уровне, отмечены значком ⊕.

Если посмотреть на оглавление, может показаться, что многое из представленного в учебнике материала вам уже знакомо. Действительно, в 7–9 классах вы изучали понятия «информация» и «информационный процесс», кодирование данных, программное обеспечение и основы программирования. Теперь вам предстоит выйти на следующий уровень владения материалом, когда человек может не только воспроизводить полученные знания, но и применять их на практике. Цель этого учебника — дать такие знания, которые позволят вам грамотно решать задачи, не рассмотренные в самом учебнике.

Общество, в котором мы живём, часто называют информационным. Прежде всего потому, что человеку приходится перерабатывать значительно бóльшие объёмы информации, чем двадцать или пятьдесят лет назад. Чтобы не утонуть в этом потоке, важно уметь структурировать информацию, правильно представлять данные в соответствии с поставленной задачей, использовать возможности современной техники и программного обеспечения.

Главная цель изучения информатики в школе — научиться грамотно применять информационные технологии для решения жизненных задач. Для этого требуется алгоритмическое мышление, которое сейчас становится необходимым навыком для успешной профессиональной работы практически в любой отрасли.

В основной школе вы, вероятно, изучали школьный алгоритмический язык или язык Паскаль. В старшей школе авторы предлагают вам познакомиться с новым языком, который называется Python. Вы увидите, что он во многом напоминает уже известные вам языки программирования, но в то же время предоставляет широкие возможности для создания современных программ. Немаловажно, что Python — это язык «широкого профиля», — он активно используется во всём мире и как учебный язык, и как язык для научных вычислений, и как язык профессионального программирования для решения разнообразных задач, включая программирование веб-сайтов.

Мы старались сделать так, чтобы содержание учебника как можно меньше зависело от программного обеспечения, установленного на ваших компьютерах. Весь курс можно успешно изучать, используя только свободное программное обеспечение (СПО) — операционную систему *Linux*, офисный пакет *OpenOffice* или *LibreOffice*, свободно распространяемые среды программирования.

Практические работы к главам и другие учебные материалы выложены на сайте поддержки учебника:

kpolyakov.spb.ru/school/osnbook.htm

Специальные навигационные значки на полях выделяют важные элементы. Значком отмечен материал, который должен быть известен вам из основной школы (он изложен очень кратко — только для того, чтобы вы вспомнили основные результаты). Значок означает важное определение или утверждение. Значок говорит о том, что при выполнении задания нужно использовать дополнительные источники, например сеть Интернет. Проектные и исследовательские работы отмечены значком . Значок служит для выделения дополнительного задания или разъяснения. Значок обозначает работу в парах или в группе. Значок выделяет межпредметные связи. Задания повышенной сложности отмечены «звёздочкой» (*).

После параграфов приведены возможные темы проектов — исследовательских работ разного уровня сложности. Мы надеемся, что каждый из вас сможет выбрать тему по душе, получить удовольствие от творческого выполнения этой работы, а потом рассказать о её результатах в классе или на школьной конференции, опубликовать отчёт в Интернете. Большинство проектов можно делать в команде из 2–3 человек.

В заключение нам хочется поблагодарить наших коллег, которые взяли на себя труд прочитать предварительные версии отдельных глав учебника и высказать множество полезных замечаний, позволивших сделать учебник более точным, ясным и понятным:

- *А. П. Шестакова*, кандидата педагогических наук, заведующего кафедрой информатики и вычислительной техники Пермского государственного педагогического университета;
- *М. А. Ройтберга*, доктора физико-математических наук, заведующего лабораторией прикладной математики Института математических проблем биологии РАН, г. Пущино;
- *С. С. Михалковича*, кандидата физико-математических наук, доцента кафедры алгебры и дискретной математики, г. Ростов-на-Дону;
- *В. М. Гуровица*, учителя информатики школы № 2007, г. Москва;
- *Т. Ф. Хирьянова*, преподавателя кафедры информатики МФТИ, г. Москва;
- *О. А. Тузову*, учителя информатики школы № 550, г. Санкт-Петербург;
- *А. В. Паньгина*, инженера Центра информационных технологий, г. Сосновый Бор;
- *Н. Е. Леко*, учителя информатики МОУ СОШ № 9, г. Тихвин;
- *С. В. Гриневича*, учителя информатики МАОУ СОШ № 146, г. Пермь;
- *Н. Б. Линева*, старшего преподавателя факультета ВМиК МГУ;
- *В. Н. Разумова*, учителя информатики МОУ «Большеелховская средняя общеобразовательная школа», с. Большая Елховка, Республика Мордовия;
- *Ю. М. Розенфарба*, учителя информатики МОУ Межозёрная СОШ, Челябинская область;
- *К. А. Малеванова*, руководителя направления телематики ООО «ОБИТ».

Техника безопасности

Как вы знаете, для работы компьютеров необходим **электрический ток**. Для стационарных компьютеров используют электрическое питание с напряжением 220 вольт, это напряжение может быть опасно для жизни. Поэтому нужно строго выполнять правила электробезопасности:

- электрическая сеть должна включать надёжную систему заземления;
- провода должны быть размещены в специальных коробах, чтобы максимально снизить вероятность их повреждения;
- нужно вынимать кабели из розеток за вилку, нельзя дёргать их за сам провод.

Для того чтобы сбои электропитания не привели к выходу компьютера из строя или потере данных, желательно использовать источники бесперебойного питания (ИБП). В аварийной ситуации ИБП позволяет переключить компьютер на питание от своих аккумуляторов и затем нормально выключить его.

Длительная работа за компьютером может нанести вред вашему здоровью. Вы смотрите на экран монитора, и хрусталики ваших глаз «настраиваются» на одно достаточно короткое расстояние. Кроме того, глаз воспринимает не привычный отражённый свет от предметов, а прямой мерцающий свет из монитора. Всё это приводит к **утомлению глаз** и может вызвать ухудшение зрения.

Если вы неправильно сидите за компьютером, например сильно сгибаетесь вперёд, увеличивается **нагрузка на позвоночник**, в первую очередь — на шею.

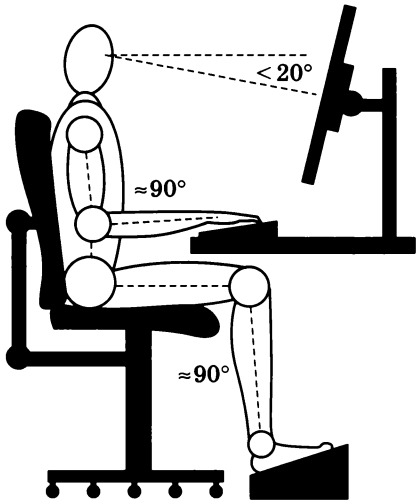
Из-за неправильного положения рук при длительной работе с клавиатурой и мышью может появиться **синдром запястья** — сдавливание нерва, которое приводит к онемению и болям в кистях рук.

До начала работы

- Входите в кабинет без верхней одежды, в сменной обуви.
- Не включайте электрическое питание в кабинете.
- Не включайте и не выключайте компьютеры без разрешения учителя.
- Не бегайте по кабинету, не делайте резких движений.

Как правильно сидеть за компьютером?

Существует специальная наука — *эргономика*, которая изучает взаимодействие человека с предметами, окружающими его на рабочем месте. Её задача — разработать правила, которые обеспечивают безопасный труд и минимальную нагрузку на организм человека.



- Для работы с компьютером нужно использовать специальную мебель, которая обеспечивает правильную посадку и позволяет удобно расположить периферийные устройства (клавиатуру, монитор, принтер и др.).
- Нужно сидеть ровно, не наклоняясь вперёд или назад и не сутулясь, чтобы не усиливать нагрузку на позвоночник.
- Нельзя работать, развалившись в кресле, такая поза быстро вызывает утомление.
- Нельзя скрещивать ноги, класть ногу на ногу.
- Взгляд должен быть направлен перпендикулярно экрану монитора, экран должен быть на расстоянии, равном длине вытянутой руки (не менее 50 см).
- Ноги ставьте на пол или специальную подставку для ног так, чтобы угол сгиба коленного сустава был около 90 градусов.
- Предплечья должны находиться на той же высоте, что и клавиатура, угол сгиба локтевого сустава должен быть около 90 градусов.
- Не выгибайте кисти в стороны, старайтесь, чтобы линия кисти была продолжением предплечья.

В последние годы разработаны специальные *эргономичные клавиатуры*. В них основной блок клавиш разбит на две части, так что пользователю приходится расставлять локти, и это снижает нагрузку на мышцы и суставы.

Во время работы

- Не трогайте провода и разъёмы соединительных кабелей компьютера.
- Не прикасайтесь к экрану монитора и задним сторонам всех устройств.
- Нельзя работать на компьютере с открытым системным блоком.
- Работайте на клавиатуре чистыми и сухими руками.
- Не кладите на аппаратуру посторонние предметы.
- Запрещается работать вдвоём за одним компьютером.
- Не работайте за компьютером при плохом самочувствии.
- Ученикам 10–11 классов можно непрерывно работать за компьютером не более 35 минут в течение урока.
- Каждые 5 минут старайтесь отводить взгляд от экрана и смотреть на предметы, находящиеся вдали (например, в окно).
- При работе за компьютером нельзя принимать пищу.
- Рабочее место должно быть хорошо освещено, источник естественного или искусственного освещения не должен создавать бликов на экране монитора.
- Необходимо проводить регулярное проветривание и влажную уборку помещения.
- Нельзя кататься на креслах с колесиками, так можно получить травму.
- Не пытайтесь самостоятельно устранять неисправности компьютера, немедленно сообщайте о них учителю.

- При появлении посторонних звуков, запаха гари, дыма и т. п. нужно сразу прекратить работу и сообщить об этом учителю.
- Общее время использования интерактивной доски на уроках не должно превышать 30 минут, а время непосредственной работы учащихся с интерактивной доской — 10 минут.

Ресурсосбережение

Все современные компьютеры имеют специальные энергосберегающие режимы работы (например, «ждущий» и «спящий» режимы), в которых они потребляют значительно меньше электроэнергии, чем обычно. Например, в спящем режиме настольных компьютеров (он также называется «гибернацией») отключается экран монитора, останавливаются дисководы жёстких дисков и отключается питание материнской платы. Содержимое оперативной памяти сохраняется в специальном файле на диске и восстанавливается из этого файла при включении компьютера.

Если пользователь некоторое время не работает с компьютером (не использует мышь, клавиатуру, сенсорный экран и т. п.), энергосберегающий режим включается автоматически. «Интервал неактивности», после которого это происходит, определяет сам пользователь, изменяя настройки компьютера.

Ресурсосбережение особенно важно для мобильных устройств (ноутбуков, планшетных компьютеров и смартфонов), которые питаются от аккумуляторов. Чтобы продлить время их работы без подзарядки, рекомендуется отключать ненужные в данный момент возможности, например поиск сетей Bluetooth и Wi-Fi, использование GPS-навигации, мобильный Интернет.

Один из главных потребителей энергии — дисплей, поэтому для него лучше не устанавливать максимальную яркость. В спящем режиме дисплей мобильного устройства отключается.

Многие смартфоны показывают, какие программы расходуют больше всего электроэнергии, эти программы тоже можно отключать, если они не нужны.

Проекты

В старшей школе, как и в основной, часть вашей работы на уроках информатики будет связана с выполнением проектных работ. Проект — это ваша самостоятельная работа, посвящённая тем вопросам, которые вам интересны. Примерные темы проектов даны в учебнике после каждого параграфа. Кроме того, вы, конечно, можете выбирать и свои собственные темы для проектов.

Проект можно выполнять одному или в группе, если работа очень объёмная и одному человеку сложно с ней справиться. Так, например, пишут программы для компьютеров. Современные программы могут содержать сотни тысяч и даже миллионы строк кода, поэтому их создают целые команды профессиональных программистов, каждый из которых выполняет свою часть работы.

Проект — это работа, которая выполняется в течение ограниченного времени, поэтому вам нужно определить, когда вы планируете завершить проект, и постараться уложиться в этот срок (например, к окончанию четверти или учебного года).

У проекта должен быть результат. Это может быть какой-то предмет, например модель самолёта или корабля. Результатом может стать проведённое мероприятие — концерт, спектакль, выставка рисунков, обустройство своего двора или подъезда. Проекты на уроках информатики обычно завершаются разработкой информационных продуктов — программ, докладов, презентаций, отчётов о проведённых исследованиях, видеофильмов и др.

Работа над проектом состоит из нескольких этапов.

Постановка задачи. На этом этапе нужно определить тему проекта и понять, в чём вы хотите разобраться, что изучить, каких результатов планируете достичь, какой продукт получится после завершения работы.

Выбор методов исследования. При выполнении проекта вы можете применять теоретические методы (к ним относятся формализация, анализ и синтез, индукция и дедукция и др.), экспериментальные (лабораторный опыт, моделирование, вычислительный эксперимент) или эмпирические (наблюдение, интервью, анкетирование, опрос, фотографирование, измерение). В отчёте нужно будет объяснить, почему именно выбранные вами методы лучше всего подойдут для достижения цели.

Подготовка исходных данных. Любые исследования основаны на каких-то исходных данных. Это могут быть числовые данные из литературы или Интернета; книги и статьи по теме проекта, рисунки, видеофильмы и др.

Перед тем как начать исследование, нужно постараться проверить исходные данные — выполнить их *верификацию* (от лат. *verus* — истинный). Вы знаете, что, например, в Интернете очень много неверной и ложной информации. Лучше всего брать данные из книг и журналов, потому что эти материалы всегда проходят рецензирование и ошибок там значительно меньше.

Проведение исследования. Это основной этап проекта, во время которого вы должны получить результаты, позволяющие ответить на интересующие вас вопросы.

Анализ результатов. После завершения исследования нужно провести анализ результатов и сделать выводы. Прежде всего, необходимо проверить достоверность полученных результатов. Если, например, ваши данные говорят о нарушении фундаментальных законов физики (например, законов сохранения массы и энергии), то это повод серьёзно задуматься о возможной ошибке.

Подумайте, удалось ли получить те результаты, которые были целью проекта. Учтите, что отрицательный результат — это тоже результат, например, может получиться и так, что поставленная вами цель недостижима (пока).

Подготовка отчёта. Для того чтобы с результатами ваших исследований смогли познакомиться другие люди, их нужно оформить в виде отчёта. Это может быть устный или письменный доклад, презентация, видеоролик, веб-сайт или даже научная статья. Ваш отчёт можно разместить в Интернете на школьном сайте или даже на своём собственном.

Очень полезно проводить публичную защиту проекта — это позволит вам обсудить результаты работы со своими товарищами и учителями, получить отзывы, выявить сильные и слабые места проекта, наметить направления для будущих исследований. Вы научитесь выступать перед аудиторией и отвечать на вопросы слушателей, а эти качества очень понадобятся вам в будущем.

Проекты — это ваши первые шаги в исследовательской работе. Возможно, такая работа вам понравится, и вы решите связать свою жизнь с наукой, которая постепенно открывает для человечества тайны окружающего нас мира.

Глава 1
ИНФОРМАЦИЯ И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

§ 1
Информатика и информация

Ключевые слова:

- информатика
- информация
- знания
- данные

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 возникла необходимость применить научный подход к задачам хранения и поиска накопленной информации. В это время зародилось новое научное направление, в котором изучалась *документальная* информация, т. е. информация 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, позволив автоматизировать рутинную работу. Считается, что 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).

---

1) Впервые этот термин использовал немецкий учёный К. Штейнбух в 1957 году (в немецком языке — *Informatik*). В 1962 году Ф. Дрейфус ввёл слово *informatique* во французский язык, затем оно было переведено на английский (англ. *informatics*).

Современная информатика, которая стала самостоятельной наукой в 70-х годах XX века, изучает теорию и практику обработки информации с помощью компьютерных систем. Обычно к информатике относят следующие научные направления:

- **теоретическую информатику** (теорию информации, теорию кодирования, математическую логику, теорию автоматов и др.);
- **вычислительную технику** (устройство компьютеров и компьютерных сетей);
- **алгоритмизацию и программирование** (создание алгоритмов и программ);
- **прикладную информатику** (персональные компьютеры, прикладные программы, информационные системы и т. д.);
- **искусственный интеллект** (распознавание образов, понимание речи, машинный перевод, логические выводы, алгоритмы самообучения).

Раньше эти вопросы частично рассматривались в других науках — математике, лингвистике (науке о языке), электронике и др. В теоретической информатике затрагиваются некоторые вопросы, относящиеся к кибернетике — науке об управлении, и к теории систем. С появлением компьютеров стало ясно, что все эти направления тесно связаны, и постепенно начала формироваться новая область научной деятельности. Информатика — это область науки в процессе становления, и круг её вопросов в будущем может измениться.

В нашем курсе мы также познакомимся с *информационными технологиями*, которые связаны с применением компьютеров во всех областях современной жизни: при оформлении документов; при подготовке книг и журналов к печати; для расчёта зарплаты; для продажи билетов на поезда и самолёты; для автоматизации производства; при проектировании зданий, кораблей, станков и т. д. Во всех этих сферах используется понятие *«информация»*.

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 рассматривают информационные процессы в живой природе. Социологи изучают ценность и полезность информации в человеческом обществе.

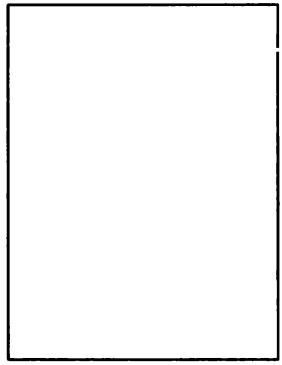
Специалистов по компьютерной технике в первую очередь интересует представление информации в виде знаков.

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

Информация **нематериальна**.

Давайте сравним два изображения одинакового размера (рис. 1.1). На первом из них пусто, а на втором мы видим фотографию. Вряд ли кто-то способен долго разглядывать чистый лист, а на фотографию можно долго смотреть, открывая всё новые и новые детали. Почему так?

Первый рисунок разглядывать неинтересно, там все одинаково — везде белый цвет. На втором рисунке есть *разнообразие*, он неоднороден. Поэтому можно сказать, что он содержит больше информации, чем первый.



1 2

Рис. 1.1

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

Зачем вообще нам нужна информация? Дело в том, что наши знания всегда в чём-то неполны, в них есть *неопределённость*. Например, вы стоите на остановке и не знаете, на каком именно автобусе вам нужно ехать в гости к другу (его адрес известен). Неопределённость мешает вам решить свою задачу. Нужный

номер автобуса можно определить, например, по карте с маршрутами транспорта или найти в Интернете с помощью смартфона. Очевидно, что при этом вы получите новую информацию, которая увеличит знание и уменьшит неопределённость.

При получении новой информации **уменьшается неопределённость знания.**

Многие выдающиеся учёные XX века (Н. Винер, У. Эшби, К. Шеннон, А. Урсул, А. Моль, В. М. Глушков) давали своё определение информации, но ни одно из них не стало общепринятым. Дело в том, что слово «информация» используется в самых различных ситуациях для обозначения того общего, что есть в разговоре людей, обмене письмами, чтении книги, прослушивании музыки, передаче сообщения через компьютерную сеть и т. д. Поэтому дать строгое определение информации не удаётся, 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».

Как получают информацию

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*) — поступает через глаза (по разным оценкам, 80–90% всей получаемой нами информации);
- *звуковая информация* (аудиальная, от англ. *audio*) — поступает через уши;
- *вкусовая информация* — поступает через язык;
- *обонятельная информация* (запахи) — поступает через нос;
- *тактильная информация* — мы её получаем с помощью осязания (кожи), «на ощупь»;
- информация, получаемая от органов человека, например с помощью *«мышечного чувства»* (человеческий мозг получает импульсы от мышц и суставов при перемещении частей тела).

Некоторые животные чувствуют магнитное поле Земли и используют его для выбора направления движения.

Формы представления информации

Информация может быть представлена (зафиксирована, закодирована) в различных *формах*:

- *текстовая информация* — последовательность символов (букв, цифр, знаков); в тексте важен порядок их расположения, например КОТ и ТОК — два разных текста, хотя они состоят из одинаковых символов;
- *числовая информация*;
- *графическая информация* (рисунки, картины, чертежи, карты, схемы, фотографии);
- *звуковая информация* (звучание голоса, мелодии, шум, стук, шорох и т. п.);
- *мультимедийная информация*, которая объединяет несколько форм представления 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.

Информация в природе

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

Ещё бóльшую роль играет информация в **живой природе**. Даже простейшие растения (синие водоросли) и животные (амёбы) 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). Животные используют информацию инстинктивно — для того, чтобы выжить, избежать опасности, продолжить род.

Наследственная информация растений и животных, определяющая строение, внешний вид, предрасположенность к болезням, хранится и передаётся из поколения в поколение с помощью молекул дезоксирибонуклеиновой кислоты (ДНК). Участки ДНК — *гены* — строятся из фрагментов четырёх типов (нуклеотидов), которые можно назвать генетическим алфавитом. Современный уровень развития биологии позволяет с помощью ДНК клонировать организмы, т. е. создавать точные копии из одной клетки-образца.

Человек, информация, знания

Обо всех изменениях в окружающем мире человек узнаёт с помощью своих органов чувств: сигналы от них («первичная» информация) постоянно поступают в мозг. Чтобы понять эти сигналы, т. е. получить информацию, человек использует **знания** — свои представления о природе, обществе, самом себе. Знания позволяют человеку принимать решения, определяют его поведение и отношения с другими людьми.

Можно считать, что знания — это модель мира, которая есть у человека. Получив информацию («поняв» сигналы, поступившие от органов чувств), он корректирует эту модель, дополняет свои знания.

Всегда ли полученная информация увеличивает наши знания? Очевидно, что нет. Например, информация о том, что $2 \cdot 2 = 4$, вряд ли увеличит ваши знания, потому что вы это уже знаете, эта информация для вас 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 всё, что было в полученном сообщении, знания не изменяются.

Вместе с тем сообщение «учёт вибронных взаимодействий континуализирует моделирование диссипативных структур» (или сообщение на неизвестном языке) также не увеличивает знания, потому что эта фраза вам непонятна. Иначе говоря, имеющихся знаний не хватает для того, чтобы воспринять новую информацию.

Эти идеи послужили основой *семантической* (смысловой) теории информации, предложенной в 60-х годах XX века советским математиком Ю. А. Шрейдером. На рисунке 1.2 показано, как зависит количество полученных знаний I от того, какая доля информации θ в сообщении уже известна получателю.

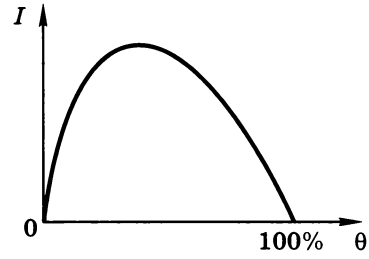


Рис. 1.2

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 методы, например измеряют длину сообщения, которое получено в результате кодирования этой информации.

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 знает, как нужно действовать в той или иной ситуации («я знаю, как играть в футбол»). К процедурным знаниям относятся алгоритмы решения различных задач.

Для того чтобы сохранить знания и передать другим людям, нужно выразить их на каком-то языке (например, рассказать, записать, нарисовать и т. п.). Только после этого их можно хранить, обрабатывать, передавать, причём 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 записанная (зафиксированная) информация.

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, т. е. получить информацию. Обрабатывая и упорядочивая информацию, человек выявляет закономерности — получает **знания**, которые он затем снова использует для извлечения информации из полученных сообщений.

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

Свойства информации

В идеале информация должна быть:

- *объективной* (не зависящей от чьего-либо мнения);
- *понятной* для получателя;
- *полезной* (позволяющей получателю решать свои задачи);
- *достоверной* (полученной из надёжного источника);
- *актуальной* (значимой в данный момент);
- *полной* (достаточной для принятия решения).

Конечно, информация не всегда обладает всеми этими свойствами. Информация в сообщении «В стакане мало молока» необъективна (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 жизненные проблемы. Вместе с тем в экстремальной ситуации, когда человек оказывается один на один с природой, такие знания очень полезны, потому что сильно увеличивают шансы на выживание, т. е. помогают достичь цели.

Слухи, байки, искажённая информация (в том числе дезинформация) — это примеры недостоверной информации. К сожалению, много недостоверной информации размещено в сети Интернет, иногда это делается умышленно (например, в ходе «информационных войн»).

Сообщение «10 лет назад тут был ларёк с мороженым» неактуально, эта информация устарела. Информация в сообщении «Сегодня будет концерт» неполна, потому что не указано время и участники концерта, и из-за этого мы не можем принять решение (идти или не идти?).

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 её достоверность никто не гарантирует.

В будущем ожидается 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» становится актуальным как никогда.

Информация в технике

Практически все современные технические устройства (телевизоры, телефоны, стиральные машины, системы управления самолётами 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 тем самым настроить станок на изготовление другой детали.

Многие опасные, тяжёлые и утомительные работы за человека могут выполнить **роботы**, у которых датчики заменяют органы чувств. Например, человекоподобный робот (*андроид*) Asimo (рис. 1.3, **www.robotonline.net**), разработанный фирмой Honda, умеет распознавать предметы, жесты, звуки, узнавать лица, разговаривать через домофон, передавать [illegible] через Интернет.

Рис. 1.3

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 «электронный нос» и «электронный язык»; они построены на основе химических датчиков.

Сейчас в теоретической информатике считается, что компьютер может хранить и обрабатывать только *данные*, но не информацию. Многие учёные полагают, что машина принципиально не может научиться понимать смысл информации и делать выводы[1]. Эту точку зрения подтверждает фактический провал проекта «компьютеров пятого поколения» (Япония, 1980-е гг.), в ходе которого планировалось создать машины, общающиеся с человеком на естественном языке. Тем не менее учёные 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 удаётся использовать огромные вычислительные мощности компьютеров для того, чтобы найти неизвестные ранее закономерности, которые можно использовать на практике.

Выводы

- Информация нематериальна. Она характеризует разнообразие в окружающем мире.
- Знания — это модель мира, которая есть у человека. При получении новой информации уменьшается неопределённость знания.
- Зафиксированная информация — это данные. Компьютеры работают только с данными.
- Информатика изучает широкий круг вопросов, связанных с автоматической обработкой данных.

Интеллект-карты

Для того чтобы всё, что вы узнали в параграфе, было проще понять (а потом — вспомнить), мы будем рисовать специальные схемы, которые называют **интеллект-картами** или диаграммами связей (по-английски — *mind maps*). В центре размещается главное понятие параграфа. В первом параграфе таких понятий два — «информатика» и «информация», от них отходят стрелки к другим связанным с ними понятиям: информационные технологии, знания, данные. Дальше схема ветвится, как дерево (рис. 1.4).

---

[1] Тем не менее суперкомпьютер Watson фирмы IBM, умеющий отвечать на вопросы, заданные на естественном языке, в 2011 году выиграл у лучших игроков в телевизионной викторине Jeopardy! (аналог телепередачи «Что? Где? Когда?»).

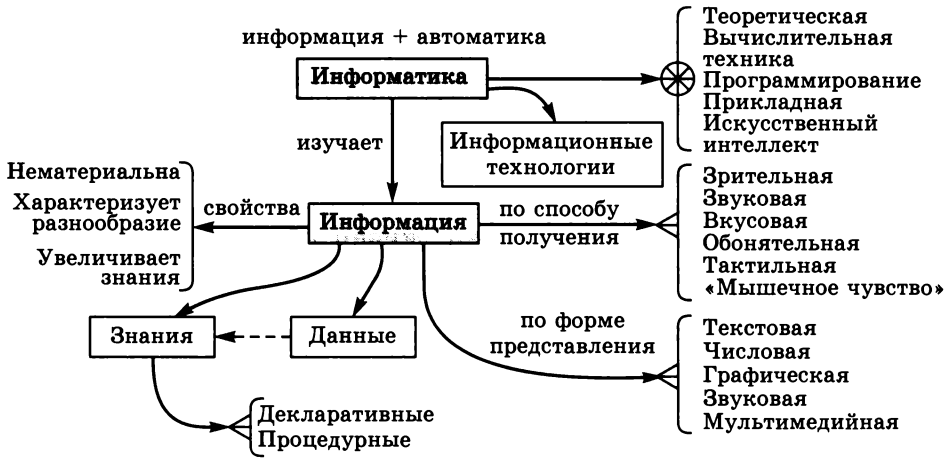


Рис. 1.4

Штриховые линии обозначают дополнительные связи между элементами схемы.

На этой интеллект-карте используются два значка, обозначающие разные типы связей между элементами схемы. Значок $\lessdot$ обозначает **различные виды** чего-либо, например различные формы представления информации. Значок $\circledast$ показывает **части целого**, например области науки, которые входят в информатику. Часто этот тип связи можно описать словом «содержит», например компьютер содержит процессор, память, устройства ввода и вывода.

Вопросы и задания

1. Использовали ли вы для решения ваших задач результаты научного направления «искусственный интеллект»?
2. Как связана неопределённость нашего знания с получением информации?
3. Как связаны информация и сложность объекта?
4. Почему, на ваш взгляд, термин «информация» трудно определить?
5. Согласны ли вы с определением информации, которое дал Н. Винер? Как вы его понимаете?
6. Что такое информация в неживой природе?
7. Как вы думаете, почему клонирование человека запрещено во многих странах, в том числе и в России?
8. Всякая ли информация увеличивает знания? Почему?
9. В чём, на ваш взгляд, разница между понятиями «данные», «информация», «знания»?

10. Почему считают, что компьютер может работать только с данными?
11. Приведите примеры необъективной, непонятной, бесполезной, недостоверной, неактуальной и неполной информации.
12. Может ли информация быть достоверной, но бесполезной? Достоверной, но необъективной? Объективной, но недостоверной? Актуальной, но непонятной?
13. Какие изменения произошли в жизни общества в результате широкого распространения Интернета?
14. Как вы считаете, смогут ли компьютеры научиться понимать смысл данных?

Подготовьте сообщение

а) «Информация в жизни общества»
б) «Интернет и изменение уклада жизни людей»
в) «Информационное общество: плюсы и минусы»
г) «Как оценить смысл информации?»
д) «Интеллект-карты (mind maps)»

Проекты

а) Количество информации в живой и неживой природе
б) ДНК и наследственная информация

§ 2
Что можно делать с информацией?

Ключевые слова:

- материальный носитель
- передача информации
- канал связи
- сигнал
- сообщение
- избыточность
- обработка информации
- кодирование
- поиск информации
- структурирование
- сортировка
- хранение информации

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

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

Изменения, происходящие с информацией (т. е. изменения свойств носителя), называются **информационными процессами**. Все эти процессы можно свести к двум основным:

- **передача информации** (данные передаются с одного носителя на другой);
- **обработка информации** (данные изменяются).

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

Для хранения информации тоже используется какой-то носитель. Однако при этом никаких изменений не происходит, поэтому хранение информации нельзя назвать процессом.

Передача информации

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

Информация проходит от **источника** к **приёмнику** через **канал связи**, в котором она должна быть связана с каким-то **материальным носителем** (рис. 1.5). Например, при разговоре людей носитель информации — это звуковые волны в воздухе. Информацию можно передавать с помощью света, лазерного луча, системы телефонной или почтовой связи. В компьютерных системах данные передаются с помощью электрических импульсов или радиоволн (в беспроводных устройствах).

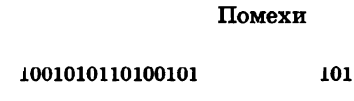


Рис. 1.5

Для передачи информации свойства носителя должны изменяться. Такие изменения называются сигналами.

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

Примеры сигналов — это изменение частоты и громкости звука, вспышки света, изменение напряжения на контактах или силы тока.

Для того чтобы превратить информацию в сигналы (**закодировать**), источник использует кодирующее устройство (блок К на рис. 1.5). Приёмник обнаруживает сигналы с помощью своих органов чувств (или датчиков). Для получения информации приёмник должен «понять» (**декодировать**) сигналы, этим занимается декодирующее устройство (блок Д на рис. 1.5). Например, для того чтобы передавать и принимать информацию с помощью радиоволн, человеку нужны вспомогательные кодирующее и декодирующее устройства: радиопередатчик, преобразующий звук в радиоволны, и радиоприёмник, выполняющий обратное преобразование. Такие устройства расширяют возможности человека — без них он может принимать только те сигналы, которые воспринимаются органами чувств.

С помощью одного сигнала (одного изменения) невозможно передать много информации. Поэтому чаще всего используется не одиночный сигнал, а *последовательность сигналов*, которая называется **сообщением**. Важно понимать, что сообщение — это только «оболочка» для передачи информации, а информация — это *содержание* сообщения. Приёмник должен сам «извлечь» (декодировать) 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 раньше использовался на флоте. Вместе с тем одно и то же сообщение может нести разную информацию для разных приёмников. Так, фраза «В Сантьяго идёт 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, искажённого помехами, можно было восстановить, оно должно быть *избыточным*, т. е. в нём должны быть «лишние» элементы, без которых смысл всё 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% (если удалить 60–80% текста, его смысл всё равно удаётся восстановить).

В курсе информатики мы будем рассматривать передачу информации именно как передачу сообщений между компьютерными системами, отвлекаясь от смысла сообщений.

Обработка информации

Обработка — это изменение информации: её формы или содержания. Среди важнейших видов обработки можно назвать:

- *создание новой информации*, например решение задачи с помощью вычислений или логических рассуждений;
- *кодирование* — запись информации с помощью некоторой системы знаков для передачи и хранения; один из вариантов кодирования — шифрование, цель которого — скрыть смысл (содержание) информации от посторонних;
- *поиск информации*, например в книге, в библиотечном каталоге, на схеме или в Интернете;
- *структурирование* — выделение важных элементов в сообщениях и установление связей между ними;
- *сортировка* — расстановка данных в заданном порядке, например расстановка чисел по возрастанию или убыванию, расстановка слов по алфавиту; задача сортировки — облегчить поиск и анализ информации.

Для обработки информации человек использует в первую очередь свой мозг. *Нейроны* (нервные клетки) 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 неразрешима. Дело, по-видимому, в том, что мозг решает такие задачи не «в лоб», не путём 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 расчётов по известным алгоритмам. Однако, в отличие от человека, компьютеру недоступны фантазия, размышления, творчество.

Хранение информации

Для хранения информации человек прежде всего использует свою память. Наш мозг — это одно из самых совершенных хранилищ информации, во многом превосходящее компьютерные средства.

К сожалению, человек многое забывает. Кроме того, необходимо передавать знания другим людям, в том числе и следующим поколениям. Поэтому в древности люди записывали информацию на камне, папирусе, берёсте, пергаменте, затем — на бумаге. В XX веке появились новые средства хранения информации: перфокарты и перфоленты, магнитные ленты и магнитные диски, оптические диски, флэш-память.

В любом случае информация хранится на каком-то **носителе**, который обладает «памятью», т. е. может находиться в разных состояниях, переходить из одного состояния в другое при каком-то внешнем воздействии, и сохранять своё состояние после прекращения этого воздействия.

При записи информации свойства носителя меняются: на бумагу наносятся текст и рисунки; на магнитных дисках и лентах намагничиваются отдельные участки; на оптических дисках образуются области, по-разному отражающие свет. При хранении эти свойства остаются неизменными, что позволяет потом читать записанную информацию. Отметим, что процессы записи и чтения — это процессы передачи информации.

Выводы

- Основные информационные процессы — это передача и обработка информации.
- Информация всегда связана с каким-то носителем (объектом или средой).
- Канал связи — это среда и технические устройства, с помощью которых передаётся информация.
- Сигнал — это изменение свойств носителя, которое используется для передачи информации.
- Сообщение — это последовательность сигналов.
- Одна и та же информация может быть передана с помощью разных сообщений. Одно и то же сообщение может нести разную информацию для разных приёмников.
- Обработка информации — это изменение её формы или содержания. Важнейшие виды обработки — это создание новой информации, кодирование, поиск информации, структурирование, сортировка.

- Информация хранится на носителе, который обладает «памятью». При хранении информации свойства носителя не изменяются.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы понимаете различие между понятиями «канал связи» и «носитель информации»?
2. Чем отличается получение информации от получения сообщения? Приведите примеры, когда приём сообщения не означает приёма информации.
3. Представьте, что придумали язык, в котором нет избыточности. В чём будет его недостаток?
4. Как вы думаете, какой вариант русского языка обладает наибольшей избыточностью: разговорный, литературный, юридический, язык авиадиспетчеров?
5. В каком из перечисленных выше языков наиболее важна помехоустойчивость? За счёт чего она достигается?
6. При каких видах обработки информации меняется её содержание?
7. При каких видах обработки информации меняется только форма её представления?
8. В каких задачах компьютер не может соревноваться с человеком? В каких ситуациях человек явно уступает компьютеру?
9. Какие средства хранения информации используются в компьютерной технике? Какие из них уже вышли или выходят из употребления? Почему?
10. Предложите критерии оценки памяти. Оцените по этим критериям достоинства и недостатки человеческой памяти по сравнению с компьютерной.

Подготовьте сообщение

а) «Компьютер и человек: кто сильнее?»
б) «Носители информации: вчера, сегодня, завтра»

Проекты

а) Избыточность русского языка
б) Влияние информатики на русский язык

§ 3
Структура информации

Ключевые слова:

- структурирование
- множество
- список
- таблица
- иерархия (дерево)
- граф
- матрица смежности
- весовая матрица

Зачем структурировать информацию?

Давайте сравним четыре сообщения.

Первое:

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

Второе:

Как ехать в Васино из Москвы?

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

Третье:

Таблица 1.1

Этапы поездки

Откуда	Куда	Транспорт
Москва	Ивановск	Самолёт
Ивановск	Ореховск	Электричка
Ореховск	пос. Ольховка	Паром (р. Слоновая)
пос. Ольховка	с. Васино	Попутная машина

Четвёртое (рис. 1.6).

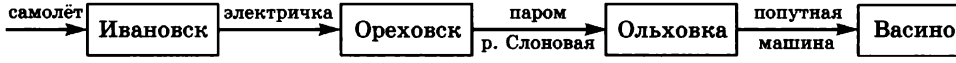


Рис. 1.6

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

Зачем 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 определённую идею. Благодаря особенностям человеческого восприятия, при таком выделении структуры улучшается передача информации от автора к читателю.

При автоматической (компьютерной) обработке правильно выбранная структура данных облегчает доступ к ним, позволяет быстро найти нужные данные.

Средства, облегчающие поиск информации, знакомы вам по работе с книгами. Самый простой (но очень утомительный!) способ найти в книге то, что нужно, — перелистывать страницу за страницей. Однако в большинстве книг есть оглавление (рис. 1.7), которое позволяет сразу найти нужный раздел, и это значительно ускоряет поиск.

В словарях слова всегда расставлены в алфавитном порядке (представьте себе, что было бы, если бы они были расположены произвольно!). Поэтому, открыв словарь в любом месте, мы можем сразу определить, куда дальше листать страницы для поиска нужного слова — вперёд или назад.

В больших книгах используют *индексы* — списки основных терминов с указанием страниц, на которых они встречаются (см. рис. 1.7).

Оглавление:

Словарь:

автомат — *automaton*
автор — *author*
адрес — *address*
алгебра — *algebra*
алгоритм — *algorithm*
архив — *archive*
архитектура — *architecture*
асимметрия — *asymmetry*

Индекс:

Рис. 1.7

Структурирование — это выделение важных элементов в информационных сообщениях и установление связей между ними.

Цели структурирования для человека — облегчение восприятия и поиска информации, выявление закономерностей. При компьютерной обработке структурирование ускоряет поиск нужных данных.

Знакомые структуры данных

Множество — это набор неповторяющихся элементов. Множество можно задать перечислением элементов или логическим выражением, которое истинно для всех элементов множества и ложно для всех элементов, не входящих в множество. Множество может состоять из конечного числа элементов, бесконечного числа элементов или вообще быть пустым. Множества, с которыми работает компьютер, не могут быть бесконечными, потому что его память конечна.

В документах множество часто оформляют в виде маркированного списка, например:

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

В таком списке порядок элементов не важен, от перестановки элементов множество не меняется (рис. 1.8).

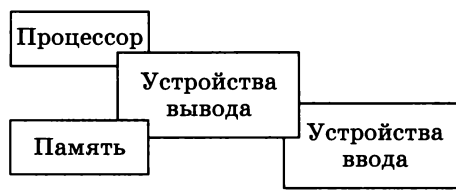


Рис. 1.8

Список — это упорядоченная последовательность элементов. Для каждого элемента в списке, кроме первого, можно назвать предыдущий элемент; для каждого элемента, кроме последнего, — следующий.

В отличие от множества элементы в списке могут повторяться. Список обычно упорядочен (отсортирован) по какому-то правилу, например по алфавиту, по важности, по последовательности действий и т. д. В тексте при оформлении списков часто используют нумерацию, например:

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

Переставить местами элементы такого списка нельзя (это будет уже другой список). Список можно задать перечислением элементов, с первого до последнего:

(надеть носки, надеть ботинки, выйти из дома),

а также представить в виде цепочки связанных элементов (рис. 1.9).

Рис. 1.9

В информатике списки часто используют в качестве моделей составных объектов. Например, предложение можно представить в виде списка слов, абзац — в виде списка предложений, текст — в виде списка абзацев.

Таблицы применяют для хранения информации об объектах, имеющих одинаковый набор свойств. Информация в базах данных представлена в виде таблиц: строка таблицы, содержащая информацию об одном объекте, называется **записью**, а столбец (название и значения свойства) — **полем**.

Таблица 1.1 в начале параграфа — это таблица типа «объект — свойства», где объект — это этап поездки, а свойства — начальный и конечный пункты, а также вид транспорта. Таблица 1.2 относится к другому типу: «объект — объект». Она устанавливает связь между двумя типами объектов (моделью рюкзака и городами).

Таблица 1.2

Продажи рюкзаков (шт.)

Модель	Пенза	Рязань	Абакан
Странник-S	128	115	215
Странник-L	74	226	124
Странник-XL	15	71	155

Таблица результатов кругового футбольного турнира (табл. 1.3) — это тоже таблица типа «объект — объект», но здесь объекты в строках и столбцах одинаковые. В каждой ячейке таблицы записан счёт — результат матча между командами.

Таблица 1.3

Результаты матчей

	Кубань	Рубин	Зенит
Кубань		3:0	1:2
Рубин	0:3		2:0
Зенит	2:1	0:2	

Деревья

Дерево — это структура данных, которая служит для описания **иерархии** — многоуровневой схемы, где одни элементы подчинены другим. Несколько деревьев образуют **лес**. Дерево состоит из **узлов (вершин)** и связей между ними — **дуг**.

Самый первый узел, расположенный на верхнем уровне, — это **корень** дерева. От корня отходят **ветви** дерева. Конечные узлы, из которых не выходит ни одна дуга, называются **листьями**. Все остальные узлы, кроме корня и листьев, — это промежуточные узлы.

Путь — это последовательность узлов, где каждый следующий связан с предыдущим.

Высота дерева — это наибольшая длина пути от корня дерева к листу.

Поддерево — это часть дерева, которая тоже представляет собой дерево.

Деревья используют для описания классификаций и файловой системы, для перебора вариантов.

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.

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.

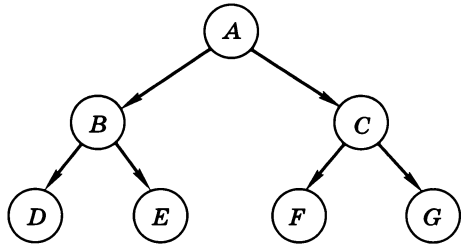


Рис. 1.10

В дереве на рис. 1.10 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.

Алгоритм вычисления арифметического выражения тоже может быть представлен в виде дерева (рис. 1.11).

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

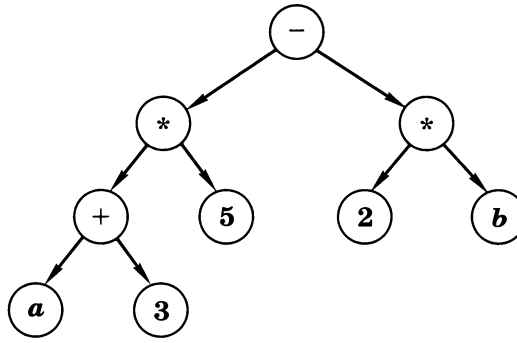


Рис. 1.11

Здесь листья — это числа и переменные, тогда как корень и промежуточные узлы — знаки операций. В таком дереве каждый узел может иметь не более двух сыновей, поэтому оно называется **двоичным** (или **бинарным**). От расположения сыновей зависит результат вычитания и деления, поэтому используют термины «левый сын» и «правый сын», «левое поддерево» и «правое поддерево». Такое дерево называется **упорядоченным**.

Вычисления идут «снизу вверх», от листьев к корню. Показанное дерево можно записать так:

$$(-(*(+(a, 3), 5), *(2, b))).$$

Самое интересное, что скобки здесь необязательны; если их убрать, то выражение всё 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 ещё одно умножение

$$- ((a+3)*5)\ (2*b)$$

и наконец, вычитание:

$$((a+3)*5) - (2*b).$$

Для получения префиксной записи мы обходили все узлы дерева в порядке «корень — левое поддерево — правое поддерево». Действительно, сначала записана метка корня («—»), затем все метки левого поддерева, а затем — все метки правого поддерева. Для поддеревьев используется тот же порядок обхода.

Если же обойти дерево в порядке «левое поддерево — правое поддерево — корень», получается *постфиксная* форма (операция

после данных). Например, рассмотренное выше выражение может быть записано в виде

$$a\ 3 + 5 * 2\ b * -$$

Для вычисления такого выражения скобки также не нужны, и это очень удобно для автоматических расчётов. Когда программа на языке программирования высокого уровня переводится в машинные коды, все выражения записываются в бесскобочной постфиксной форме и именно так и вычисляются. Постфиксная форма для компьютера удобнее, чем префиксная, потому что, когда программа доходит до знака операции, все данные для выполнения этой операции уже готовы.

Графы

Граф — это набор **вершин (узлов)** и связей между ними **(рёбер)**. Данные о структуре графа можно записать в виде **матрицы смежности**: таблицы, в которой единица на пересечении строки *X* и столбца *Y* означает, что между узлами *X* и *Y* есть связь; ноль указывает на то, что связи нет (рис. 1.12).

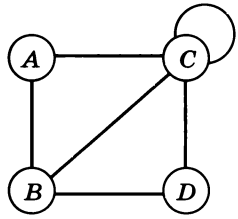


Рис. 1.12

Петля — это ребро, которое начинается и заканчивается в одной и той же вершине.
Степень вершины — это количество рёбер, с которыми связана эта вершина. При этом петля считается дважды (с вершиной связаны оба конца ребра!).

Граф можно описать 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:

$$(A\ (B.\ C),\ B\ (A.\ C.\ D),\ C\ (A,\ B,\ C,\ D),\ D\ (B,\ C))$$

Строго говоря, граф — это математический объект, а не рисунок. Конечно, его можно нарисовать на плоскости, но матрица смежности (и список смежности) не даёт никакой информации о том, как именно располагать узлы друг относительно друга. Для таблицы на рис. 1.12 возможны, например, такие варианты (рис. 1.13).

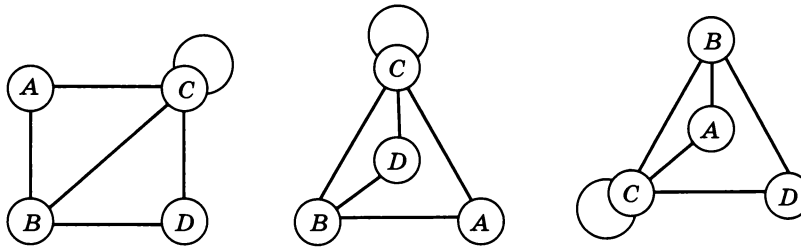


Рис. 1.13

Так как каждое ребро графа имеет два конца, сумма степеней всех вершин графа — всегда чётное число: оно равно удвоенному количеству рёбер. Отсюда следует, что число вершин графа, имеющих нечётную степень, должно быть чётно (иначе сумма степеней вершин окажется нечётной). Эти результаты известны как **лемма о рукопожатиях**. Такое название связано с математической задачей: доказать, что в любой компании число людей, пожавших руку нечётному числу других людей, чётно.

Путь — это последовательность рёбер, по которым можно перейти из одного узла в другой.
Связный граф — это граф, между любыми вершинами которого существует путь.
Цикл — это замкнутый путь в графе. В графе на рис. 1.12 есть циклы *ABCA*, *BCDB*, *ABDCA*.
Дерево — это связный граф, в котором нет циклов.
Взвешенный граф — это граф, с каждым ребром которого связано некоторое число (**вес ребра**). Оно может означать, например, стоимость проезда или длину дороги. Данные о взвешенном графе хранятся в виде **весовой матрицы** — таблицы, в которой на пересечении строки X и столбца Y записывается вес ребра из вершины X в вершину Y, а пустая клетка означает, что ребра между этими вершинами нет (рис. 1.14).

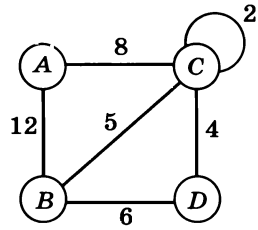


Рис. 1.14

Если в графе немного вершин, весовая матрица позволяет легко определить наилучший маршрут из одного узла в другой простым перебором вариантов, например с помощью дерева.

Рассмотрим граф, заданный весовой матрицей (числа определяют стоимость поездки между соседними пунктами), — рис. 1.15.

Рис. 1.15

Найдём наилучший путь из *A* в *B* — такой, при котором общая стоимость поездки минимальная. Этапы построения дерева перебора для этой задачи показаны на рис. 1.16. Чёрным фоном выделена конечная точка того пути из *A* в *B*, который на данном шаге построения дерева имеет наименьшую стоимость. Числа около рёбер показывают стоимость поездки по этому участку, а индексы у названий вершин обозначают общую стоимость проезда в данную вершину из вершины *A*.

Обратите внимание, что мы не доводили до конца все ветви дерева, например не проверяли различные варианты продолжения пути *ACED*. Дело в том, что к моменту, показанному на рис. 1.16, *в*, мы нашли путь *ACEB* стоимостью 6, поэтому любое продолжение пути *ACED* (который уже имеет стоимость 6!) не сможет улучшить результат.

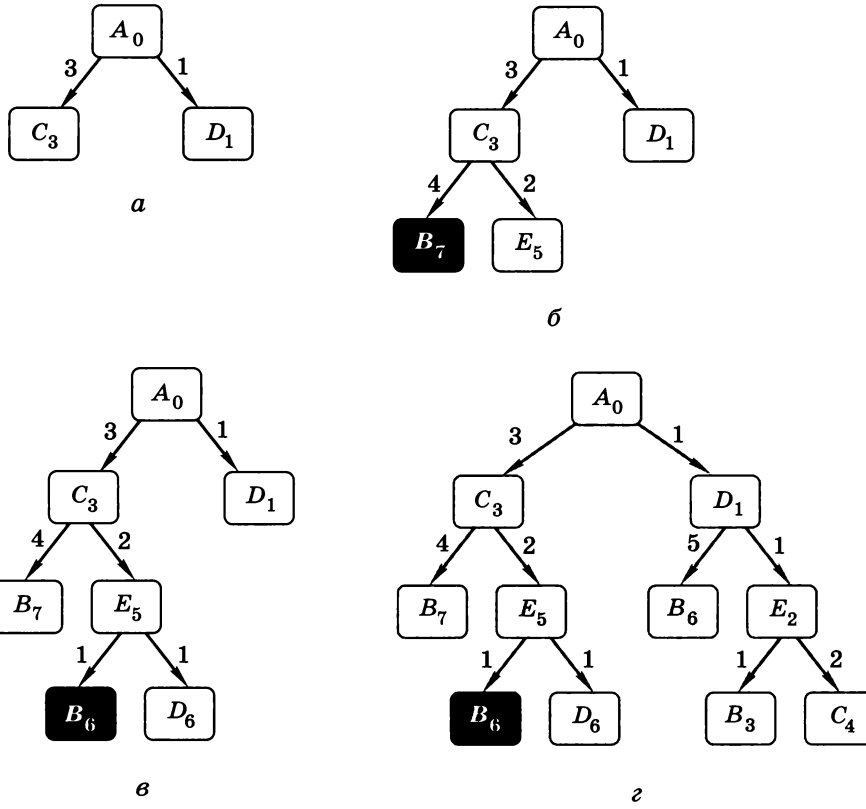
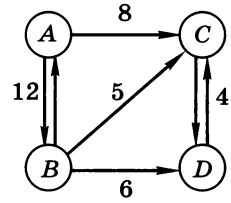


Рис. 1.16

Ориентированный граф (коротко — **орграф**) — это граф, в котором рёбра имеют направления. Рёбра в орграфе называются **дугами**. Орграф может служить моделью сети дорог с односторонним движением, его матрица смежности и весовая матрица могут быть несимметричны относительно главной диагонали, выделенной серым фоном (рис. 1.17).



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

Рис. 1.17

Пример ориентированного графа показан на рис. 1.18.

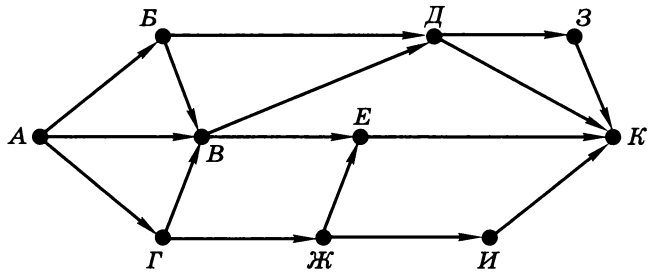


Рис. 1.18

В этом графе можно выделить особые вершины — *А* и *К*. Из вершины *А* дуги только выходят, в неё не входит ни одна дуга. Такая вершина называется **истоком**. Из вершины *К*, наоборот, не выходит ни одна дуга, все дуги в неё входят — это **сток**. Можно представить себе, что какой-то поток «вливается» в граф через исток *А*, проходит через граф — систему труб — и «выливается» через сток *К*.

Граф на рис. 1.18 не содержит циклов, такие графы называются **ациклическими**. В ациклическом орграфе можно найти количество различных путей от истока к стоку.

Будем двигаться по стрелкам от начальной вершины *А*, записывая около каждой вершины количество возможных путей из вершины *А* в эту вершину. На каждом шаге ищем вершины, в которые можно попасть только из уже отмеченных вершин; метка для каждой из них получается суммированием меток всех вершин, откуда можно попасть в данную вершину (рис. 1.19).

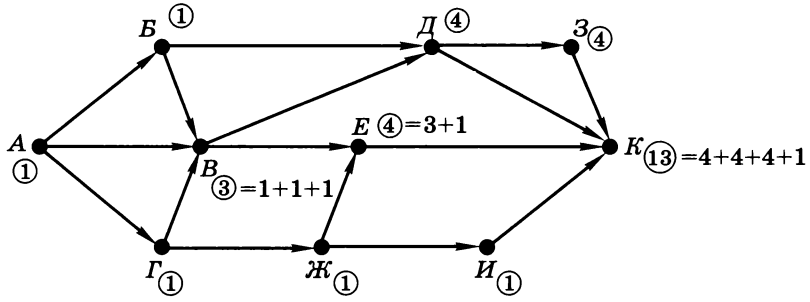


Рис. 1.19

В данном случае существует 13 различных путей из вершины *А* в вершину *К*.

Для каждой вершины графа на рис. 1.19 можно указать те вершины, которые нужно обработать до неё. Например, результат для вершины *Е* (количество путей из *А* в *Е*) можно подсчитать только тогда, когда будут известны результаты для вершин *В* и *Ж*. Говорят, что вершины этого графа связаны *отношением частичного порядка*: вершина *Е* следует за вершинами *В* и *Ж*. Поэтому можно составить список вершин в том порядке, в котором нужно выполнять вычисления, например, так: (*А*, *Б*, *Г*, *В*, *Ж*, *Е*, *Д*, *И*, *З*, *К*).

Такая операция называется *топологической сортировкой*.

Приведённый список — не единственный, например можно переставить вершины *Б* и *Г*, потому что они не связаны отношением частичного порядка: всё равно, какую из них обрабатывать раньше.

Выводы

- Структурирование — это выделение важных элементов в информационных сообщениях и установление связей между ними.
- Множество — это набор неповторяющихся элементов. Множества, с которыми работает компьютер, не могут быть бесконечными, потому что его память конечна.
- Список — это упорядоченная последовательность элементов.
- Дерево — это структура данных, которая служит для описания иерархии — многоуровневой схемы, где одни элементы «подчинены» другим. Несколько деревьев образуют лес.
- Граф — это набор вершин (узлов) и связей между ними — рёбер.
- Матрица смежности — это таблица, в которой единица на пересечении строки и столбца обозначает ребро между соответствующими вершинами, а ноль — отсутствие связи.
- Взвешенный граф — это граф, с каждым ребром которого связано некоторое число — вес ребра. Взвешенный граф описывается весовой матрицей.
- Ориентированный граф (орграф) — это граф, в котором каждое ребро имеет направление. Матрица смежности и весовая матрица орграфа могут быть несимметричными.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Зачем структурируют информацию?
2. Как используются оглавление, словарь и индекс для быстрого поиска нужной информации? Чем эти средства отличаются друг от друга?
3. 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-накопителе.
4. Чем отличаются множество и список?
5. Предложите, как можно записать табличные данные в виде списка.
6. У корня дерева четыре потомка, и все они являются листьями. Нарисуйте это дерево. Сколько в нём узлов?
7. В чём разница между понятиями «ребро» и «дуга»?
8. Как по матрице смежности определить, есть ли петли в графе?
9. Как по весовой матрице определить, сколько ребёр содержит неориентированный граф? Ориентированный граф?
10. Как по весовой матрице определить степени всех вершин в неориентированном графе? В ориентированном графе?
11. Как по весовой матрице определить длину заданного пути в графе (например, длину пути *ADEBC* в графе, заданном весовой матрицей на рис. 1.15)?
12. Для графа на рис. 1.19 запишите другие варианты списка вершин, в котором сохраняется частичный порядок. У кого из вас получилось больше вариантов?

Подготовьте сообщение

а) «Задача о Кёнигсбергских мостах»
б) «Лемма о рукопожатиях»

Проекты

а) Решение логических задач с помощью графов
б) Программа для поиска оптимального пути
в) Программа для поиска количества различных путей

ЭОР к главе 1 на сайте ФЦИОР (http://fcior.edu.ru)

- Виды и свойства информации
- Информация и информационные процессы
- Информация, информационные процессы в обществе, природе и технике
- Что изучает «Информатика»
- Классификация информационных процессов
- Единицы измерения информации

Практические работы к главе 1

Работа № 1 «Оформление документа»
Работа № 2 «Таблицы и списки»
Работа № 3 «Деревья»
Работа № 4 «Графы»

Глава 2
КОДИРОВАНИЕ ИНФОРМАЦИИ

§ 4
Дискретное кодирование

Ключевые слова:

- знаковая система
- знак
- символ
- аналоговый сигнал
- цифровой сигнал
- дискретизация

Кодирование — это представление информации в форме, удобной для её хранения, передачи и автоматической обработки.
Код — это правило, по которому сообщение преобразуется в цепочку знаков.
Язык — это система знаков и правил, используемая для записи и передачи информации.
Формальный язык — это язык, в котором однозначно определяется значение каждого слова, а также правила построения предложений и придания им смысла.

Знаковые системы

Для хранения, передачи и обработки информации нужно как-то зафиксировать её, записать с помощью некоторой системы знаков (знаковой системы).

Знак — это заменитель какого-то объекта, который вызывает в сознании человека образ этого объекта. Например, знак напоминает горнолыжника, такой знак называется *пиктограммой*.

Символ — это знак, о значении которого люди договорились, например «§» — это символ параграфа, а ₽ — символ рубля. Если значение символов неизвестно, мы не сможем понять смысл сообщения, записанного с их помощью. Поэтому многие древние тексты до сих пор не расшифрованы.

Знаки бывают зрительные (буквы, цифры, ноты, дорожные знаки), слуховые (звуки устной речи, звуковые сигналы), осяза-

тельные (азбука Брайля для слепых людей), обонятельные (их используют животные).

Знаковая система определяется **алфавитом** (набором используемых знаков) и правилами выполнения операций с этими знаками.

Знакомые всем знаковые системы — это **естественные языки**, с помощью которых люди общаются в быту (русский, английский, китайский и др.). В естественных языках кроме правил есть и исключения. Кроме того, одно и то же слово может иметь различный смысл в зависимости от *контекста*, т. е. отрывка текста, в котором оно употребляется.

В научных публикациях такая ситуация недопустима, потому что смысл текста должен быть понят однозначно. В таких случаях используют **формальные языки**, в которых каждое слово и словосочетание имеют чётко определённое единственное значение, и нет никаких исключений. Примеры формальных языков — математические и химические формулы, нотная запись, языки программирования. Все формальные языки — *искусственные*, они разработаны людьми для обмена информацией в специальных областях знаний.

Как вы знаете из курса основной школы, в компьютерах для кодирования всех видов информации используется специальная знаковая система — **двоичный код**. Давайте разберёмся, почему такое решение оказалось самым лучшим.

Аналоговые и дискретные сигналы

Как вы уже знаете, информация передаётся в закодированном виде с помощью сигналов. Согласно определению, сигнал — это изменение свойств носителя, которое используется для передачи информации. Изменение выбранного свойства (например, силы тока, напряжения, освещённости) во времени можно описать в виде функции. Далее такую функцию тоже будем называть сигналом, как это принято в электронике и вычислительной технике.

В любой компьютерной системе очень важно обеспечить надёжный обмен данными в условиях, когда на сигнал действуют помехи. Поэтому желательно выбрать такой способ кодирования информации, который позволяет лучше всего решить эту задачу.

Элементы электронных устройств, как правило, обмениваются данными с помощью электрических сигналов; для получения информации приёмник должен измерить этот сигнал (чаще всего — напряжение на контактах или силу тока). В таких устройствах,

как радиоприёмник и микрофон, 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 (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 значения. В большинстве случаев эти искажения — случайные ошибки, не поддающиеся учёту. Фактически приёмник получает не исходный сигнал, посланный источником (сплошная линия на рис. 2.1), а искажённый (штриховая линия).

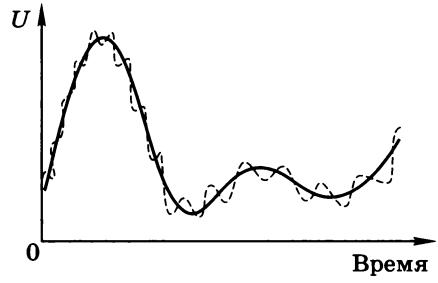


Рис. 2.1

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

Если использовать аналоговые компьютеры, мы будем при каждом расчёте 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 инженеров, разрабатывающих компьютеры, и они нашли интересное решение: если не удаётся точно измерить сигнал, нужно вообще отказаться от его измерения, а просто через некоторый интервал времени T определять, в каком из двух состояний находится сигнал (эти состояния можно обозначить как 1 и 0)[1]. Тогда мы получаем огромное преимущество: при небольших помехах искажение сигнала не влияет на передачу данных: если напряжение выше некоторого порога U_1, то считается, что сигнал равен 1, а все сигналы, меньшие другого порога U_0, считаются равными нулю (рис. 2.2).

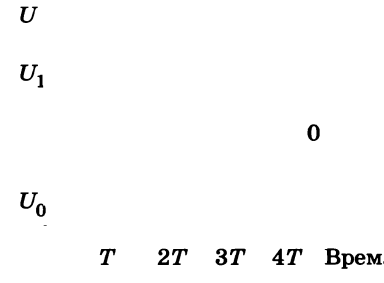


Рис. 2.2

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, принадлежащих некоторому конечному множеству.

Обратите внимание на важный момент — мы естественным образом пришли к необходимости использования дискретных сигналов, когда потребовалось точно и однозначно воспринимать передаваемую информацию с учётом неизбежных помех.

Так как каждому значению дискретного сигнала всегда можно поставить в соответствие определённый знак, такой сигнал можно рассматривать как сообщение, записанное с помощью конечного набора знаков (алфавита).

---

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

Этот принцип применим не только к компьютерам. Переход от наскальных рисунков к алфавитному письму, где каждый знак имеет чётко определённое значение, — это переход от аналоговых сигналов к дискретным, цель которого — максимально исключить неоднозначное понимание смысла. Код Морзе и двоичный код — это тоже дискретные коды.

Дискретизация

Поскольку данные в компьютерах передаются с помощью дискретных сигналов, они могут хранить и обрабатывать только **дискретную информацию**, т. е. такую, которая может быть записана с помощью конечного количества знаков некоторого алфавита. Поэтому для ввода любых данных в компьютер их нужно перевести в дискретный код. Например, линия, нарисованная на бумаге, при сканировании представляется в памяти компьютера в виде отдельных элементов — пикселей. Такая процедура называется дискретизацией.

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

Дискретизация — это представление непрерывного объекта в виде множества отдельных элементов.

Дискретизацию мы используем и в жизни. Например, когда измеряют температуру воздуха, обычно округляют её до целых градусов, хотя температура изменяется непрерывно, а не скачками: она может быть равной и 18,25 °С, и 18,251 °С, и 18,2513 °С и т. д.

Множество вещественных чисел непрерывно (между любыми двумя различными числами есть ещё бесконечно много других), а множество целых чисел — дискретно. Таким образом, при округлении мы выполняем дискретизацию данных.

Все приборы, которые показывают результаты измерений в цифровом виде, выполняют дискретизацию. Например, стрелка в обычном спидометре автомобиля может принимать любое положение, это непрерывный (*аналоговый*) прибор. А цифровой спидометр показывает дискретные данные — скорость с округлением до 1 км/ч.

Всем известное иррациональное число π содержит бесконечное количество знаков в дробной части. Если мы хотим записать, чему равно π, 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 (рис. 2.3).

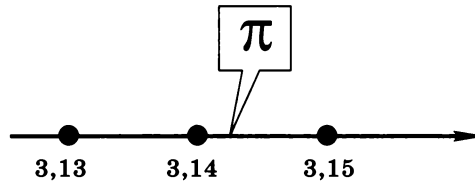
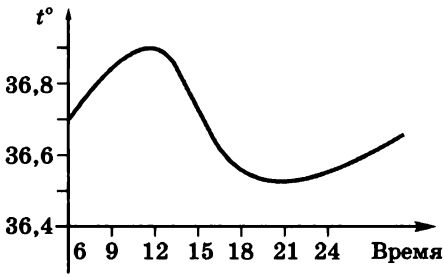
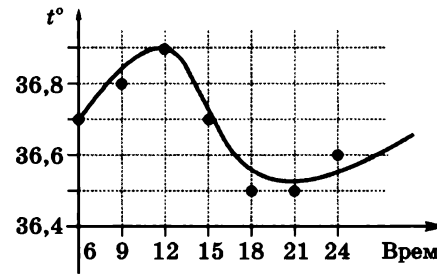


Рис. 2.3

Изменение высоты столбика термометра — это аналоговые данные, а *записанная* температура, округлённая до десятых долей градуса (например, 36,6°), — дискретные. Дискретность состоит в том, что записанные значения температуры изменяются скачкообразно (через 0,1°), — это дискретизация по уровню, или *квантование*. Кроме того, обычно температуру больного измеряют не непрерывно, а несколько раз в день — появляется дискретизация по времени (рис. 2.4).



Аналоговые данные



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°

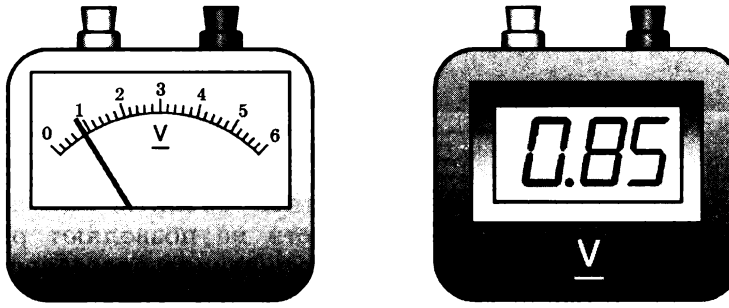
Дискретные данные

Рис. 2.4

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 и т. д. Однако в любой практической задаче есть некоторый предел, после которого увеличение точности уже никак не влияет на конечный результат.

Из приведённого примера понятно, что непрерывность и дискретность — это не свойство самой информации, а свойство её *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.

Ещё один пример — аналоговые («стрелочные») и цифровые вольтметры, которые измеряют одну и ту же величину, но выводят результат измерения в разном виде (рис. 2.5).



Аналоговый прибор

Дискретный прибор

Рис. 2.5

С одной стороны, переход к дискретному представлению информации делает более надёжной передачу данных (если обе стороны одинаково понимают используемые знаки). С другой стороны, при дискретизации часть информации теряется.

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 бесконеч-

ное количество цифр, но в расчётах чаще всего достаточно знать это значение с точностью не более 10 знаков.

Идеальная непрерывность существует только в теории. Мы считаем дерево, пластмассу, металл непрерывными, 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», рис. 2.6). Классическая («плёночная») фотография считается аналоговой, но при увеличении снимка с фотоплёнки нельзя бесконечно получать всё новые и новые детали — предел «уточнения» определяется величиной зерна светочувствительного материала.

Рис. 2.6

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 глаза. 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 специальные приборы (например, бинокли и микроскопы).

Выводы

- Знак — это заменитель какого-то объекта, который вызывает в сознании человека образ этого объекта.
- Знаковая система определяется алфавитом (набором используемых знаков) и правилами выполнения операций с этими знаками.

- Формальный язык — это язык, в котором однозначно определяется значение каждого слова, а также правила построения предложений и придания им смысла.
- Аналоговый сигнал — это сигнал, который в любой момент времени может принимать любые значения в заданном диапазоне.
- Дискретный (цифровой) сигнал — это последовательность значений, каждое из которых принадлежит некоторому конечному множеству.
- Дискретизация — это представление непрерывного объекта в виде множества отдельных элементов.

Интеллект-карта

Рис. 2.7

Вопросы и задания

1. В чём различие между пиктограммой и символом?
2. Приведите примеры знаковых систем, которые не упоминаются в учебнике.
3. Какие бытовые устройства работают с аналоговыми сигналами?
4. Какие системы связи используют аналоговые сигналы, а какие — дискретные?
5. Почему при использовании аналоговой техники передача информации всегда происходит с искажениями?
6. Почему с помощью дискретного сигнала можно передавать информацию практически без искажений? Искажается ли форма такого сигнала под воздействием помех?

7. Сигнал изменяется в моменты времени, кратные 1 секунде, и может принимать одно из 16 возможных значений. Можно ли назвать такой сигнал дискретным?
8. Что такое дискретизация по времени и дискретизация по уровню?
9. Приведите пример дискретизации сигнала: а) только по уровню; б) только по времени. Нарисуйте графики процессов.
10. Почему при дискретизации, как правило, происходит потеря информации? В каких случаях потери информации не будет?
11. Как можно уменьшить потери информации при дискретизации? Почему не стоит стремиться максимально уменьшить эти потери?
12. Приведите примеры, когда одна и та же информация может быть представлена в аналоговой и дискретной формах.

*13. Выясните, какие музыкальные инструменты позволяют извлекать только дискретные звуки (заранее определённые ноты), а какие — звук любой частоты.

14. Объясните фразу «при увеличении точности дискретизации свойства непрерывной и дискретной информации практически совпадают».

Проекты

а) Аналоговые и дискретные музыкальные инструменты
б) Дискретность в полиграфии

§ 5
Равномерное и неравномерное кодирование

Ключевые слова:

- кодирование
- равномерный код
- неравномерный код
- правило умножения
- правило сложения

Алфавит — это набор знаков, который используется в языке.
Мощность алфавита — это количество знаков в алфавите.
Равномерный код — это код, в котором все кодовые слова имеют одинаковую длину.
Неравномерный код — это код, в котором кодовые слова имеют различную длину.
Двоичное кодирование — это кодирование с помощью двух знаков.
1 бит — это одна двоичная цифра (один знак сообщения, записанного в двоичном коде).

Равномерное кодирование

Если алфавит языка состоит из M знаков (имеет мощность M), то количество различных сообщений длиной L знаков вычисляется как

$$N = M^L.$$

Двоичный код — это сообщение, использующее алфавит из двух знаков (0 и 1), поэтому для двоичного кода эта формула запишется в виде

$$N = 2^L.$$

Например, восьмиразрядная ячейка памяти компьютера может хранить одно из $2^8 = 256$ различных значений.

Если заданное количество вариантов не равно степени числа 2, выбирают длину кода с запасом. Например, для кодирования номера спортсмена в интервале от 1 до 200 нужно использовать не меньше, чем 8 двоичных разрядов (бит), поскольку 7 бит позволяют закодировать только 128 различных значений (этого мало!), а 8 бит — уже 256:

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

В середине XX века в СССР была разработана электронно-вычислительная машина «Сетунь», в которой для хранения и обработки данных использовался троичный код с алфавитом {–1, 0, 1}. В таком компьютере восьмиразрядная ячейка памяти могла хранить одно из $3^8 = 6561$ значения.

Правило умножения

При дополнительных ограничениях количество возможных знаков в разных позициях сообщения может различаться, поэтому нужно использовать более общее **правило умножения**:

$$N = M_1 \cdot M_2 \cdot \ldots \cdot M_L.$$

Здесь M_k — это возможное количество вариантов выбора знака в позиции k.

Задача 1. Сколько существует различных сообщений длины 5 в четырёхбуквенном алфавите {A, B, C, X}, если известно, что буква «X» может появляться только на первом или на последнем месте?

Решение. По условию на первом и последнем местах может быть любая из четырёх букв, поэтому $M_1 = M_5 = 4$. В остальных трёх позициях могут быть любые буквы, кроме «Х», поэтому $M_2 = M_3 = M_4 = 3$. Тогда общее количество различных сообщений равно $N = 4 \cdot 3 \cdot 3 \cdot 3 \cdot 4 = 432$.

Ответ: 432.

Задача 2. Сколько существует пятизначных десятичных чисел, в которых каждая цифра встречается только один раз?

Решение. Представим себе, что нам нужно заполнить пять ячеек разными цифрами. У нас есть 10 цифр (от 0 до 9). Чтобы получить пятизначное число, в первой ячейке мы не можем использовать цифру 0, т. е. можем заполнить её девятью способами. Во вторую ячейку мы не можем записывать ту цифру, которую выбрали для первой ячейки, поэтому в ней может находиться одна из 9 оставшихся цифр. При заполнении третьей ячейки нужно учитывать, что для первых двух цифры уже выбраны и их использовать нельзя, остаётся 8 свободных цифр, и т. д. (рис. 2.8).

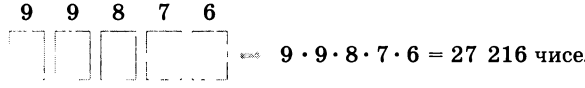


Рис. 2.8

Ответ: 27 216 чисел.

Задача 3. Вася составляет 5-буквенные слова, в которых есть только буквы «К», «Р», «О», «Т», причём буква «О» используется в каждом слове ровно 1 раз. Каждая из других допустимых букв может встречаться в слове любое количество раз или не встречаться совсем. Словом считается любая допустимая последовательность букв, не обязательно осмысленная. Сколько существует таких слов, которые может написать Вася?

Решение. Единственная буква «О» может стоять в любой из 5 позиций. Предположим, что мы определили её место, тогда на каждом из оставшихся 4 мест может оказаться любая из остальных трёх букв («К», «Р» или «Т). Таким образом, при любой заданной позиции буквы «О» Вася может составить $3^4 = 81$ слово, а всего существует $5 \cdot 3^4 = 405$ пятибуквенных слов с единственной буквой «О».

Ответ: 405.

Неравномерное кодирование

При неравномерном кодировании коды различных знаков могут иметь разную длину. Это делается для того, чтобы сократить длину сообщения, используя сведения о частотах встречаемости различных знаков. Знаки, которые встречаются в сообщениях чаще других (например, для текстов на русском языке — это пробел и буквы «О», «Е» и «А»), получают более короткие коды, а редко встречающиеся знаки — более длинные.

Задача 4. По каналу связи передаются сообщения, каждое из которых содержит 16 букв «А», 8 букв «Б», 4 буквы «В» и 4 буквы «Г» (других букв в сообщениях нет). Каждую букву кодируют двоичной последовательностью. Какой код из приведённых ниже следует выбрать, чтобы длина сообщения была как можно меньше?

Код 1:	А — 0,	Б — 10,	В — 110,	Г — 111;
код 2:	А — 00,	Б — 01,	В —10,	Г — 11.

Решение. Считаем общее количество бит в сообщении для кода 1:

$$16 \cdot 1 + 8 \cdot 2 + 4 \cdot 3 + 4 \cdot 3 = 56 \text{ бит}$$

и общее количество бит в сообщении для кода 2:

$$16 \cdot 2 + 8 \cdot 2 + 4 \cdot 2 + 4 \cdot 2 = 64 \text{ бит.}$$

Код 1 даёт наименьшую длину сообщения, поэтому выбираем его.

Ответ: код 1.

Задача 5. По каналу связи передаются сообщения, содержащие только 4 буквы: «А», «И», «С», «Т». В любом сообщении больше всего букв «А», следующая по частоте буква — «С», затем — «И». Буква «Т» встречается реже, чем любая другая. Для передачи сообщений нужно использовать один из следующих неравномерных двоичных кодов:

код 1:	А — 1,	И — 01,	С — 001,	Т — 000;
код 2:	А — 101,	И — 11,	С — 0,	Т — 100

так, чтобы сообщения были как можно короче. Какой код следует выбрать?

Решение. Для того чтобы длина сообщения была наименьшей, должно выполняться правило: «чем чаще встречается буква, тем короче её код». К сожалению, это правило не выполняется для кодов 1 и 2: в коде 1 длина кодового слова для буквы «С» боль-

ше, чем длина кодового слова для буквы «И» (а лучше было бы сделать наоборот!); для кода 2 длина кодового слова для буквы «А» — не самая маленькая из всех.

Предположим, что в сообщении буква «А» встречается α раз, буква «И» — β раз, буква «С» — γ раз и буква «Т» — δ раз, причём по условию задачи $\alpha > \gamma > \beta > \delta$. При кодировании с помощью кода 1 получаем сообщение длиной

$$S_1 = \alpha + 2\beta + 3\gamma + 3\delta,$$

а при кодировании с помощью кода 2 длина сообщения равна

$$S_2 = 3\alpha + 2\beta + \gamma + 3\delta.$$

Находим разность: $S_2 - S_1 = (3\alpha + 2\beta + \gamma + 3\delta) - (\alpha + 2\beta + 3\gamma + 3\delta) = 2\alpha - 2\gamma$. Поскольку $\alpha > \gamma$, получаем: $S_2 - S_1 > 0$, т. е. код 1 более экономичный.

Ответ: код 1.

Задача 6. Сколько символов можно закодировать с помощью кода Морзе, используя кодовые слова различной длины — от 1 до 5 знаков?

Решение. Кодовые слова длиной, скажем, 2 знака можно выбирать независимо от кодовых слов другого размера. Поэтому достаточно найти количество кодовых слов N_L для каждого возможного значения длины кодового слова L и сложить эти числа:

$$N = N_1 + N_2 + N_3 + N_4 + N_5.$$

Это правило называется **правилом сложения**.

Так как в коде Морзе используются всего два знака (точка и тире), из общей формулы получаем: $N_L = 2^L$. Поэтому $N = 2^1 + 2^2 + 2^3 + 2^4 + 2^5 = 62$.

Ответ: 62.

Выводы

- Кодирование — это представление информации в форме, удобной для её хранения, передачи и обработки. Правило такого преобразования называется кодом. Кодом называют также набор знаков закодированного сообщения.
- Если алфавит языка состоит из M символов (имеет мощность M), количество различных сообщений длиной L знаков вычисляется как $N = M^L$. Для двоичного кода эта формула запишется в виде $N = 2^L$.

- Правило умножения: $N = M_1 \cdot M_2 \cdot \ldots \cdot M_L$, где M_k — это возможное количество вариантов выбора знака в позиции k сообщения.
- Правило сложения: $N = N_k + N_{k+1} + \ldots + N_m$, где N_q ($q = k, \ldots, m$) — это количество различных сообщений длиной q.

Интеллект-карта

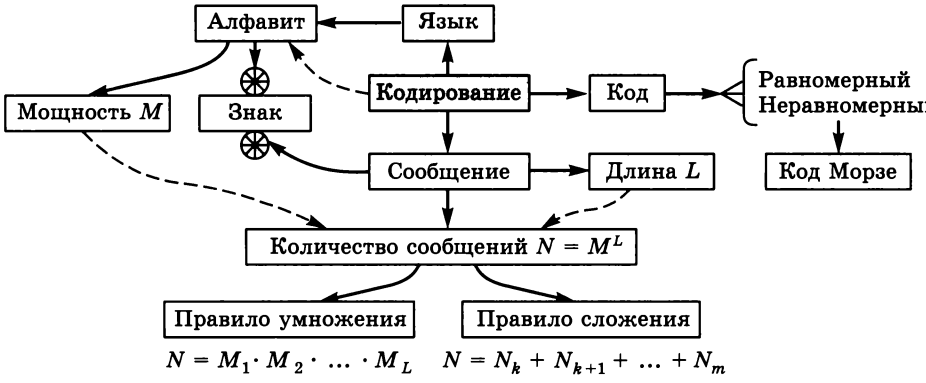


Рис. 2.9

Подготовьте сообщение

«Основные формулы комбинаторики»

Проекты

а) Оптимальный код Морзе для русского языка
б) Программа для частотного анализа текстов

§ 6
Декодирование

Ключевые слова:

- декодирование
- условие Фано
- префиксный код
- постфиксный код

Декодирование — это восстановление информационного сообщения из последовательности кодов.

Сообщения, записанные с помощью равномерного кода, всегда декодируются однозначно. Для этого достаточно разбить сообщение на группы известной длины и декодировать каждую группу по кодовой таблице.

Условие Фано

В некоторых случаях даже при использовании неравномерного кода не требуется вводить символ-разделитель. Для этого достаточно выполнения условия Фано: ни одно кодовое слово не совпадает с началом другого кодового слова. Такой код называют **префиксным**.

Пример 1. Пусть для кодирования первых пяти букв русского алфавита используется кодовая таблица:

А	Б	В	Г	Д
000	10	01	110	001

Это неравномерный код, поскольку в нём есть двух- и трёхзначные кодовые слова. Построим для этой кодовой таблицы дерево, в котором от каждого узла (кроме листьев) отходят два ребра, помеченные цифрами 0 и 1. Чтобы найти код символа, нужно пройти по стрелкам от корня дерева к нужному листу, выписывая метки стрелок, по которым мы переходим (рис. 2.10).

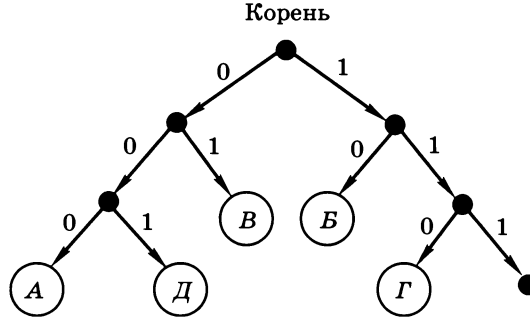


Рис. 2.10

Заметим, что ни одна буква не лежит на пути от корня к другой букве. Это значит, что условие Фано выполняется, и любую правильную кодовую последовательность можно однозначно декодировать с начала. 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
```

Пример 2. Рассмотрим другую кодовую таблицу:

А	Б	В	Г	Д
000	01	10	011	100

Здесь условие Фано не выполняется, поскольку код буквы «Б» (01) совпадает с началом кода буквы «Г» (011), а код буквы «Д» (100) начинается с кода буквы «В» (10). Дерево для этой кодовой таблицы выглядит так (рис. 2.11).

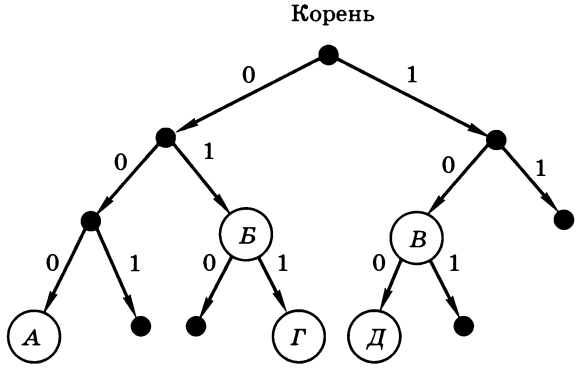


Рис. 2.11

Тем не менее по кодовой таблице можно проверить, что выполнено «обратное» условие Фано: ни одно кодовое слово не совпадает с окончанием другого кодового слова (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 удаётся только перебором вариантов.

Пример 3. 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 сообщение однозначно, возможно, 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 четвёртая 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»:

АБВ 1101,

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 удаётся, потому что в таблице нет кодов 1, 10 и 100. Таким образом, сообщение может быть декодировано двумя способами: АБАГД и АБВГА.

Пример 4. Для кодирования некоторой последовательности, состоящей из букв «А», «Б», «В» и «Г», решили использовать неравномерный двоичный код, удовлетворяющий условию Фано. Для буквы «А» использовали кодовое слово 0, для буквы «Б» — кодовое слово 110. Требуется найти коды для букв «В» и «Г», при которых суммарная длина всех четырёх кодовых слов минимальна.

Построим дерево, которое содержит известные кодовые слова для букв «А» и «Б» (рис. 2.12).

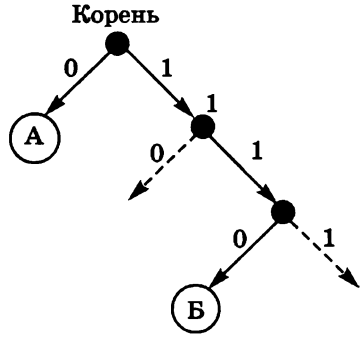


Рис. 2.12

Штриховыми линиями показаны ветви дерева, на которых нужно разместить коды оставшихся букв. Поскольку требуется обеспечить выполнение условия Фано, кодовые слова для них должны соответствовать листьям этого дерева. В данном случае есть две свободных ветви, и нужно выбрать два кодовых слова (для букв «В» и «Г»). Поэтому можно использовать кодовые слова 10 и 111, так что суммарная длина всех четырёх кодовых слов будет равна 1 + 3 + 2 + 3 = 9 (рис. 2.13).

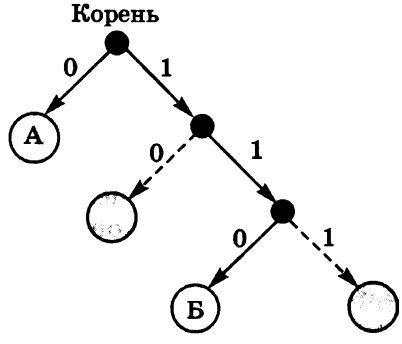


Рис. 2.13

Если бы требовалось выбрать не два, а три кодовых слова (скажем, для букв «В», «Г» и «Д»), на одной из ветвей (самой короткой!) нужно было бы сделать развилку (рис. 2.14).

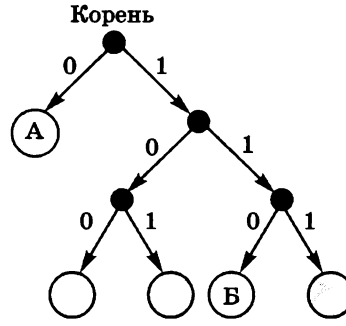


Рис. 2.14

Листья дерева, которые соответствуют оптимальному выбору трёх кодовых слов, выделены фоном на рис. 2.14.

Граф Ал. А. Маркова

Пример 3 показывает, что неоднозначное декодирование возможно тогда, когда начало кода одной буквы совпадает с концом кода другой, поэтому можно переместить границу между кодами букв в сообщении. Например, последовательность 01011 может быть декодирована как АВ (01011) и как БГ (01011). Следовательно, нужно обратить внимание на те цепочки, которые встречаются как в начале, так и в конце кодовых слов.

Покажем, как найти сообщения, которые декодируются неоднозначно. Для таблицы из примера 3 построим граф Ал. А. Маркова следующим образом:

1. Определим все последовательности, которые совпадают с началом какого-то кодового слова и одновременно с концом какого-то кодового слова; в данном случае это три последовательности:
   0 (начало кода буквы «А» и конец кода буквы «Б»);
   1 (начало кода буквы «Г» и конец кода буквы «Д»);
   10 (начало кода буквы «Д» и конец кода буквы «Б»).
   Цепочки 01 и 11 не учитываем, потому что они совпадают с кодами букв «А» и «Г».
2. Добавим к этому множеству {0, 1, 10} пустую строку, которую обычно обозначают буквой Λ (прописная гречес-

кая буква «лямбда»); элементы полученного множества {Λ, 0, 1, 10} становятся вершинами графа (рис. 2.15).

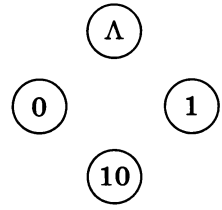


Рис. 2.15

3. Соединим вершины дугами (направленными рёбрами) по такому правилу: две вершины X и Y соединяются дугой, если последовательная запись кода вершины X, кода некоторой буквы (или нескольких букв) и кода вершины Y даёт код ещё одной буквы (рис. 2.16).

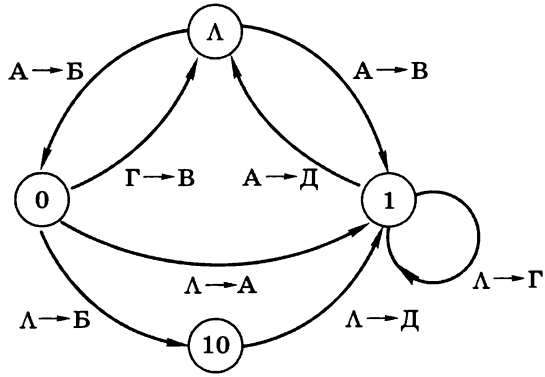


Рис. 2.16

Например, последовательная запись пустой строки (Λ), кода буквы «А» (01) и цепочки 0 даёт цепочку 010, которая совпадает с кодом буквы «Б»; поэтому рисуем дугу из вершины Λ в вершину 0; у этой дуги пишем «А → Б» и т. д. Поскольку код буквы «Г» можно записать как 11 = 1Λ1, у вершины 1 появляется петля «Λ → Г».

Любое сообщение декодируется однозначно тогда и только тогда, когда в полученном таким образом графе нет циклов, включающих вершину Λ.

В нашем графе есть несколько таких циклов, например:

- цикл Λ0Λ, соответствующий сообщению ΛА0ГΛ = 01011; это сообщение может быть расшифровано как АВ и как БГ;
- цикл Λ1Λ, соответствующий сообщению ΛА1АΛ = 01101; это сообщение может быть расшифровано как АД и как ВА;
- цикл Λ01Λ, соответствующий сообщению ΛА01АΛ = 010101; это сообщение может быть расшифровано как ААА и как БД;
- цикл Λ0101Λ, соответствующий сообщению ΛА0101АΛ = = 01010101; это сообщение может быть расшифровано как АБД и как БДА.

Кроме того, из-за петли у вершины 1 неоднозначно декодируется любая последовательность вида 01...101, где многоточие обозначает любое количество единиц. Например, сообщение 0111101 может быть декодировано как АГД или ВГА (см. пример 3).

Пример 5. Существуют коды, для которых условия Фано не выполняются, но все сообщения однозначно декодируются. В кодовой таблице

А	Б	В
0	11	010

код буквы «А» совпадает как с началом, так и с окончанием кода буквы «В», т. е. код не является ни префиксным, ни постфиксным.

Проверим, можно ли однозначно декодировать сообщения, построенные с помощью такого кода. Множество цепочек, которые совпадают с началом и концом кодовых слов, состоит из пустой строки и единицы: {Λ, 1}. Граф, построенный с помощью приведённого выше алгоритма, содержит две вершины и одну петлю (рис. 2.17).

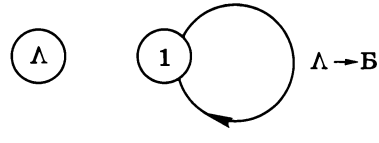


Рис. 2.17

В этом графе нет цикла, содержащего вершину Λ, поэтому любое сообщение, записанное с помощью такого кода, декодируется однозначно. Это можно показать и простыми рассуждениями:

1) все цепочки 11 в сообщении — это коды букв «Б», иначе они не могут образоваться;
2) все цепочки 010 — это коды букв «В»;
3) остальные знаки в сообщении могут быть только нулями — это коды букв «А».

Выводы

- Декодирование — это восстановление информационного сообщения из последовательности кодов.
- Сообщения, записанные с помощью равномерного кода, всегда декодируются однозначно.
- Для того чтобы сообщение, закодированное с помощью неравномерного кода, можно было однозначно декодировать, достаточно выполнения условия Фано: ни одно кодовое слово не является началом другого кодового слова. Такой код называют префиксным.
- Условие Фано выполняется тогда и только тогда, когда в дереве, построенном по кодовой таблице, все вершины-знаки являются листьями.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Перечислите достаточные условия, при которых можно однозначно декодировать сообщение, закодированное с помощью неравномерного кода.
2. Как построить дерево для проверки выполнения обратного условия Фано?
3. В каких случаях для декодирования приходится использовать перебор вариантов?
4. *Работа в парах.* Придумайте слово из 5–6 букв и закодируйте его с помощью неравномерного кода, для которого выполняется условие Фано. Предложите напарнику декодировать сообщение.

Проекты

а) Программа для декодирования сообщений
б) Программа для проверки условия Фано

§ 7
Алфавитный подход к оценке количества информации

Ключевые слова:

- алфавит
- мощность алфавита
- двоичный код
- бит

1 байт = 8 бит = 2^3 бит.
1 Кбайт (килобайт) = 1024 байта = 2^{10} байт = 2^{13} бит.
1 Мбайт (мегабайт) = 1024 Кбайт = 2^{10} Кбайт = 2^{20} байт = $= 2^{23}$ бит.
1 Гбайт (гигабайт) = 1024 Мбайт.
1 Тбайт (терабайт) = 1024 Гбайт.
1 Пбайт (петабайт) = 1024 Тбайт.

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 (знаков): 0 и 1. Как вы знаете, каждая двоичная цифра несёт 1 бит информации, поэтому полная информация в сообщении 0101001101001110 равна 16 бит.

Теперь представим себе, что нужно закодировать программу для Робота, который умеет выполнять команды «вперёд», «назад», «влево» и «вправо». Для этого можно использовать алфавит, состоящий из 4 символов: ↑↓→←. Сколько информации содержится в сообщении ↑←↑↑→↓↓↓↓→←? Каждый полученный символ может быть любым из 4 символов алфавита, а для кодирования одного из 4 вариантов требуется уже 2 бита. Поэтому полное сообщение из 11 символов содержит $11 \cdot 2 = 22$ 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 минимальное количество бит информации i, приходящихся на каждый символ сообщения, так чтобы выполнилось условие $2^i \geq M$:

$N = 2^i$	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 L, это и есть полное количество информации:

$$I = L \cdot i.$$

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 несёт 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 приёмник, 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, её содержание не имеет никакого значения, важен только объём. Почтальону всё 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 объёма текста чаще всего применяют именно алфавитный подход.

Задача 1. Найдите количество информации на 10 страницах текста (на каждой странице 32 строки по 64 символа) при использовании алфавита из 256 символов.

Решение. Сначала определим информационную ёмкость одного символа. Так как $256 = 2^8$, один символ несёт $i = 8$ бит, или 1 байт информации.

Вычислим общее количество символов L. На одной странице содержится

$$32 \cdot 64 = 2^5 \cdot 2^6 = 2^{11} \text{ символов,}$$

тогда во всей книге — $L = 10 \cdot 2^{11}$ символов. Информационный объём текста равен

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

Ответ: 20 Кбайт.

Задача 2. В некоторой стране автомобильный номер длиной 6 символов составляется из прописных 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 целым количеством байт. Сколько байт памяти необходимо для хранения 10 автомобильных номеров?

Решение. Для записи номера используется 12 различных букв и 10 цифр (0..9), так что алфавит состоит из 22 знаков. Для кодирования одного знака требуется не менее 5 бит ($2^4 = 16 < 22 \le 32 = 2^5$), поэтому на весь номер необходимо $6 \cdot 5 = 30$ бит.

По условию на каждый номер выделяется целое число байт, поэтому берём ближайшее значение, которое содержит не менее 30 бит, это 4 байта (32 бита). Для хранения 10 номеров нужно выделить $10 \cdot 4 = 40$ байт.

Ответ: 40 байт.

Задача 3. Для регистрации на сайте некоторой страны пользователю требуется придумать пароль длиной не более 11 символов. В качестве символов используются десятичные цифры и 12 различных букв местного алфавита, причём все буквы используются в двух регистрах: как строчные, так и прописные (регистр буквы имеет значение!). Для хранения каждого такого пароля на компьютере отводится минимально возможное и одинаковое целое количество байт, при этом используется посимвольное кодирование, и все символы кодируются одинаковым и минимально возможным количеством бит. Сколько байт памяти необходимо для хранения 60 паролей?

Решение. По условию в пароле можно использовать 10 цифр (0..9) + 12 прописных букв местного алфавита + 12 строчных букв, всего $10 + 12 + 12 = 34$ символа. Для кодирования одного из 34 символов нужно выделить 6 бит памяти, так как $2^5 = 32 < 34 \le 2^6 = 64$ (5 бит не хватает, а 6 — достаточно).

Для хранения всех 11 символов пароля нужно $11 \cdot 6 = 66$ бит. Поскольку пароль должен занимать целое число байт, берём ближайшее значение, содержащее не меньше, чем 66 бит: это 9 байт (72 бита). На 60 паролей нужен выделить $60 \cdot 9 = 540$ байт.

Ответ: 540 байт.

Выводы

- Алфавитный подход к оценке количества информации состоит в следующем:
  1) определяем мощность алфавита M;
  2) по таблице степеней числа 2 определяем минимальное количество бит информации i, приходящихся на каждый символ сообщения, так чтобы выполнилось условие $2^i \geq M$;
  3) умножаем i на число символов в сообщении L, это и есть полное количество информации: $I = L \cdot i$.
- При использовании алфавитного подхода считается, что каждый символ несёт одинаковое количество информации. Частоты встречаемости символов и сочетаний символов не учитываются.
- Количество информации определяется только длиной сообщения и мощностью алфавита.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Приведите примеры ситуаций, когда смысл информации особого значения не имеет, а важен только её объём.
2. Какие утверждения справедливы 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 приёмника;
   е) разные символы могут нести разное количество информации.
3. Технический документ перевели с одного языка на другой (считаем, что это было сделано максимально близко к тексту). Изменился ли смысл документа? Изменился ли его объём?
4. Как вы думаете, почему компьютеру легко извлечь несколько предложений с конкретных страниц документа, но трудно составить аннотацию к нему?
5. *Работа в парах*. Придумайте задачу на вычисление количества информации и решите её. Затем предложите напарнику сделать то же самое. Сравните ваши результаты.

§ 8
Системы счисления

Ключевые слова:

- система счисления
- позиционная система
- основание
- разряд
- развёрнутая форма записи числа
- схема Горнера

Система счисления — это правила записи чисел с помощью специальных знаков — цифр, а также соответствующие правила выполнения операций с этими числами.

Позиционная система счисления — это такая система, в которой значение цифры (её вес) полностью определяется её местом (позицией) в записи числа.

Алфавит системы счисления — это используемый в ней набор цифр.

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

Целые числа

В быту мы используем краткую форму записи чисел в десятичной системе счисления, например 6375. В такой записи 6 стоит в третьем разряде и обозначает 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

$$6375 = 6 \cdot 10^3 + 3 \cdot 10^2 + 7 \cdot 10^1 + 5 \cdot 10^0.$$

Это **развёрнутая форма** записи числа, т. е. его разложение по степеням числа 10. Здесь 10 — это **основание системы счисления**.

Это же число можно представить в другой форме, которая называется **схемой Горнера**:

$$6375 = ((6 \cdot 10 + 3) \cdot 10 + 7) \cdot 10 + 5.$$

С помощью схемы Горнера можно найти значение числа с помощью наименьшего количества операций умножения и сложения, не используя возведение в степень.

Аналогичные выражения можно получить и в общем виде, для чисел, записанных в любой системе счисления. Пусть основание системы счисления равно $p > 1$. Её алфавит состоит из p

цифр[1] от 0 до $p-1$, т. е. «старшая» (наибольшая) цифра в позиционной системе счисления на единицу меньше, чем основание.

Рассмотрим четырёхзначное 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 развёрнутой форме

разряды → 3 2 1 0

$$a_3a_2a_1a_0 = a_3 \cdot p^3 + a_2 \cdot p^2 + a_1 \cdot p^1 + a_0 \cdot p^0$$

или с помощью схемы Горнера:

$$a_3a_2a_1a_0 = ((a_3p + a_2) \cdot p + a_1) \cdot p + a_0.$$

Из этих формул следует, что a_0 — это остаток от деления исходного числа на основание p. Поэтому если запись числа в системе с основание p закачивается на 0, то это число нацело делится на p, если заканчивается на два нуля ($a_1 = a_0 = 0$) — делится на p^2, и т. д.

Развёрнутую форму записи числа и схему Горнера можно использовать для **перевода числа в десятичную систему**. Например, пусть число 12345 записано в пятеричной системе счисления (системе с основанием 5). Нижний индекс 5 в записи 1234_5 обозначает основание системы счисления (для десятичной системы основание не указывают). Тогда

$$1234_5 = 1 \cdot 5^3 + 2 \cdot 5^2 + 3 \cdot 5^1 + 4 \cdot 5^0 = \\ = 125 + 2 \cdot 25 + 3 \cdot 5 + 4 = 194$$

$$1234_5 = ((1 \cdot 5 + 2) \cdot 5 + 3) \cdot 5 + 4 = \\ = (7 \cdot 5 + 3) \cdot 5 + 4 = 38 \cdot 5 + 4 = 194$$

Развёрнутую запись числа можно использовать для обратного перехода **от десятичной системы к системе с основанием p**. Вспомним, 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 найдём a_2 как остаток от деления на p. Разделив на p ещё раз, получаем последнюю цифру a_3.

---

[1] При $p > 10$ используются также и латинские буквы.

Переведём, например, число 194 в пятеричную систему счисления ($p = 5$). Найдём остаток от деления на 5:

$$194 = 38 \cdot 5 + 4.$$

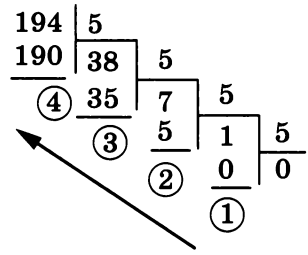
Таким образом, мы нашли последнюю цифру — 4. Частное равно 38, повторяем ту же операцию:

$$38 = 7 \cdot 5 + 3.$$

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

$$7 = 1 \cdot 5 + 2,$$

третья с конца цифра — 2, а четвёртая — 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:



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 остаётся выписать найденные остатки в обратном порядке.

Такой алгоритм фактически использует схему Горнера, «раскручивая» её в обратном порядке. При каждом делении частное и остаток определяются однозначно, поэтому представление числа в любой позиционной системе единственно.

Рассмотренные приёмы позволяют записать любое неотрицательное число в заданной позиционной системе счисления. Признаком отрицательного числа служит знак «—», после которого по тем же правилам записывается модуль числа.

Задача 1. Число 71 в системе счисления с основанием x записывается как 56_x. Найдите x.

Решение. Представим это число в развёрнутой форме:

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

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

Ответ: 13.

Задача 2. Решите уравнение $16_x + 33_x = 52_x$.

Решение. Представим все числа в развёрнутой форме:

$$16_x = x + 6; \quad 33_x = 3 \cdot x + 3; \quad 52_x = 5 \cdot x + 2.$$

Тогда исходное уравнение приводится к виду $4 \cdot x + 9 = 5 \cdot x + 2$, его решение: $x = 7$.

Ответ: 7.

Задача 3. Число 71 в системе счисления с основанием x записывается как 155_x. Найдите x.

Решение. Представим это число в развёрнутой форме:

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

Квадратное уравнение $71 = x^2 + 5x + 5$ имеет два решения: $x_1 = -11$ и $x_2 = 6$. Искомое основание системы счисления положительно, поэтому правильный ответ — 6.

Ответ: 6.

Задача 4. Найдите все основания систем счисления, в которых запись числа 24 оканчивается на 3.

Решение. Здесь удобно использовать схему Горнера, из которой следует

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

где x — неизвестное основание системы счисления, а k — некоторое натуральное число или 0. Отсюда сразу получаем $21 = k \cdot x$, т. е. все интересующие нас основания являются делителями числа 21. Это могут быть основания 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.

Ответ: 7 и 21.

Задача 5. Найдите все десятичные числа, не превосходящие 40, запись которых в системе счисления с основанием 4 оканчивается на 11.

Решение. 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 $k = 0, 1, 2, 3, \ldots$, находим соответствующие числа $N = 5, 21, 37, 53, \ldots$. Из них только 5, 21 и 37 удовлетворяют условию (не больше 40).

Ответ: 5, 21 и 37.

Задача 6. Все 5-буквенные слова, составленные из букв «А», «О», «У», записаны в алфавитном порядке. Вот начало списка:

1. ААААА
2. ААААО
3. ААААУ
4. АААОА

...

Найдите слово, которое стоит на 140-м месте от начала списка.

Решение. Как ни странно, эта задача прямо связана с позиционными системами счисления. В словах используется набор из трёх разных символов, для которых задан порядок (алфавитный). Заменив буквы «А», «О» и «У» соответственно на цифры 0, 1 и 2 выпишем начало списка:

1. 00000
2. 00001
3. 00002
4. 00010

Это числа, записанные в троичной системе счисления в порядке возрастания. Тогда легко понять, что на 140-м месте от начала списка стоит число 139, записанное в троичной системе счисления:

$$139 = 12011_3.$$

Заменив обратно цифры на буквы, получаем ответ: 12011 → ОУАОО.

Ответ: ОУАОО.

Задача 7. Значение арифметического выражения $9^{2017} + 3^{2015} - 9$ записали в системе счисления с основанием 3. Определите, сколько цифр 0, 1 и 2 содержится в этой записи.

Решение. Сначала вспомним, что в десятичной системе число 10^N записывается как единица и N нулей: $\underbrace{100\ldots0}_{N}$, число $10^N - 1$

записывается как N девяток: $\underbrace{99\ldots9}_{N}$, а число $10^N - 10^M$ — как $N - M$ девяток, за которыми стоят M нулей:

$$10^N - 10^M = \underbrace{99\ldots9}_{N-M}\underbrace{00\ldots0}_{M}.$$

В другой позиционной системе получим подобные результаты, только вместо девяток нужно использовать старшую цифру системы счисления (в троичной системе — цифру 2):

$$3^N = \underbrace{10\ldots0_3}_{N}, \quad 3^N - 1 = \underbrace{2\ldots2_3}_{N}, \quad 3^N - 3^M = \underbrace{2\ldots2}_{N-M}\underbrace{0\ldots0_3}_{M}.$$

Теперь вернёмся к нашей задаче. Запишем выражение через степени числа 3:

$$9^{2017} + 3^{2015} - 9 = (3^2)^{2017} + 3^{2015} - 3^2 = 3^{4034} + 3^{2015} - 3^2.$$

Слагаемое 3^{4034} даст в троичной записи одну единицу, а разность $3^{2015} - 3^2$ представляет собой $2015 - 2 = 2013$ цифр 2.

Общая длина троичной записи числа $9^{2017} + 3^{2015} - 9$ определяется старшим слагаемым 3^{4034}, она равна 4035 (число 3^{4034} записывается в троичной системе как единица, за которой следуют 4034 нуля). Поэтому число нулей в этой записи равно $4035 - 1 - 2013 = 2021$.

Ответ: 2021 ноль, одна единица и 2013 двоек.

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 развёрнутую 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, a_4 — это отдельные

цифры, стоящие соответственно в разрядах −1, −2, −3 и −4. Это число может быть записано в развёрнутой форме

$$\begin{matrix}\text{разряды} \rightarrow & -1 & -2 & -3 & -4 \\ 0, & a_1 & a_2 & a_3 & a_4\end{matrix} = a_1 \cdot p^{-1} + a_2 \cdot p^{-2} + a_3 \cdot p^{-3} + a_4 \cdot p^{-4}$$

или с помощью схемы Горнера:

$$0,a_1a_2a_3a_4 = p^{-1} \cdot (a_1 + p^{-1} \cdot (a_2 + p^{-1} \cdot (a_3 + p^{-1} \cdot a_4))).$$

Умножив это число на p, получаем $a_1,a_2a_3a_4$. Если взять целую часть результата, мы получим цифру a_1. Таким же способом можно найти оставшиеся цифры дробной части: на каждом шаге берём дробную часть, умножаем её на p и запоминаем *целую часть* результата — это и будет очередная цифра записи числа в системе с основанием p. Например, переведём число 0,9376 в пятеричную систему:

Вычисления	Целая часть	Дробная часть
$0,9376 \cdot 5 = 4,688$	4	0,688
$0,688 \cdot 5 = 3,44$	3	0,44
$0,44 \cdot 5 = 2,2$	2	0,2
$0,2 \cdot 5 = 1$	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 становится равной нулю. Это означает, что все остальные цифры дробной части — нули. Всегда ли это произойдёт? 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 $^1/_3$ 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, переведём число 25,375 в шестеричную систему:

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

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

Выводы

- Для перевода числа из системы счисления с основанием p в десятичную систему можно использовать запись числа в развёрнутой форме или схему Горнера.
- Для перевода целого 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, и затем выписать найденные остатки в обратном порядке.
- Для перевода дробной части числа из десятичной системы в систему счисления с основанием p нужно умножать её на p, отбрасывая целую часть на каждом шаге, пока не получится 0 или не будет достигнута требуемая точность. Затем нужно выписать отброшенные целые части в том порядке, в котором они были получены.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как связан в позиционной системе вес цифры и разряд, в котором она стоит?
2. Чем хороша схема Горнера с точки зрения вычислений?
3. Как перевести число из любой позиционной системы в десятичную?
4. Какие цифры входят в алфавит девятеричной системы?
5. Как вы думаете, можно ли использовать систему счисления с основанием 1000000? В чём могут быть проблемы?
6. Сформулируйте алгоритм перевода числа из семеричной системы в десятичную.
7. Сформулируйте алгоритм перевода числа из десятичной системы в семеричную.
8. Как по записи числа в пятеричной системе сразу увидеть, делится ли оно на 5? На 25? На 125?
9. Для каждого из чисел запишите следующее число (большее данного на единицу) в той же системе счисления: 10111_2, 222_3, 323_4, 1234_5, 35555_6, 456_7, 77_8.
10. Сформулируйте алгоритм перевода дробной части десятичного числа в шестеричную систему счисления.
11. Как вы думаете, почему не все конечные десятичные дроби можно представить в виде конечных дробей в других системах счисления?
12. Какие дробные десятичные числа можно записать в виде конечной дроби в шестеричной системе счисления? Ответ обоснуйте.

Проекты

а) Схема Горнера в вычислениях
б) Программа для перевода смешанных чисел в другую систему счисления

§ 9
Двоичная система счисления

Ключевые слова:

- метод подбора
- кодирование дробных чисел

Алфавит **двоичной системы счисления** состоит из двух цифр: 0 и 1.
Все данные в компьютерных устройствах хранятся и обрабатываются как числа, представленные в двоичной системе счисления.
Для построения двоичной записи числа можно использовать общий способ (деление на 2 и выписывание остатков от деления в обратном порядке).

Перевод чисел в двоичную систему

Для перевода небольших чисел в двоичную систему счисления удобно использовать **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 — развёрнутая запись числа в двоичной системе счисления, поэтому краткая запись состоит из цифр, обведённых кружками. Единицы стоят в шестом, третьем, втором и нулевом разрядах:

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

Обратите внимание, что число, равное 2^N, в двоичной системе записывается как единица, за которой следуют N нулей.

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

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

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

Цифра числа	Вычисления	Результат
1	1	
0	$1 \cdot 2 + 0$	2
0	$2 \cdot 2 + 0$	4
1	$4 \cdot 2 + 1$	9
1	$9 \cdot 2 + 1$	19
0	$19 \cdot 2 + 0$	38
1	$38 \cdot 2 + 1$	77

Арифметические операции

Двоичные числа, как и десятичные, можно складывать в столбик, начиная с младшего разряда. При этом используют следующие правила (таблицу сложения):

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

В двух последних случаях, когда сумма $2 = 10_2$ или $3 = 11_2$ не может быть записана с помощью одного разряда, происходит перенос в следующий разряд.

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

$$\begin{array}{r} 11111 \\ 10110_2 \\ +\ 111011_2 \\ \hline 1010001_2 \end{array}$$

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

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

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

Чтобы понять принцип, временно вернёмся 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 заём из предыдущего разряда, в котором стоит 2. В результате к младшему разряду добавляется 10 (основание системы счисления), а в предыдущем цифра 2 уменьшается до 1. Теперь можно выполнить вычитание: $1 + 10 - 9 = 2$. В старшем разряде вычитаем из оставшейся единицы ноль:

$$\begin{array}{r} \dot{2}1 \\ 110 \\ -\ 9 \\ \hline 12 \end{array}$$

Здесь точкой сверху обозначен разряд, из которого берётся заём.

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

$$\begin{array}{r} \dot{2}001 \\ 1991\,10 \\ -\ 9 \\ \hline 1992 \end{array}$$

Что изменится в двоичной системе? Когда берётся заём, в «рабочий» разряд добавляется уже не 10, а $10_2 = 2$ (основание

системы счисления), а все «промежуточные» разряды (между «рабочим» и тем, откуда берется заём) заполняются не девятками, а единицами (старшей цифрой системы счисления). Например:

$$\begin{array}{r} \overset{\bullet}{0}112 \\ 11\cancel{00}0_2 \\ -\quad 1_2 \\ \hline 10111_2 \end{array} \qquad \begin{array}{r} \overset{\bullet}{0}11\overset{\bullet}{2}02 \\ 1\cancel{000}101_2 \\ -\quad 11011_2 \\ \hline 101010_2 \end{array}$$

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):

$$\begin{array}{r} \times\ 10101_2 \\ 101_2 \\ \hline + \quad 10101_2 \\ 10101_2 \\ \hline 1101001_2 \end{array} \qquad \begin{array}{r|l} -\ 10101_2 & 111_2 \\ \cline{2-2} 111_2 & 11_2 \\ \cline{1-1} 111_2 & \\ \cline{1-1} 0 & \end{array}$$

Сложение и вычитание степеней числа 2

Задача 1. Постройте двоичную запись числа $2^{12} + 2^7 - 2^5$.

Решение. Вспомним, что число 2^N в двоичной системе записывается как единица, за которой следуют N нулей:

$$2^N = 1\underbrace{0\ldots0}_{N}.$$

Построим число $2^N - 2^M$ при $M < N$, используя правила вычитания:

$$\begin{array}{r} \overset{N}{\overbrace{100\ldots00}}\overset{M}{\overbrace{0000}}_2 \\ -\quad 1\underbrace{0\ldots0}_{M}{}_2 \\ \hline \underbrace{11\ldots11}_{N-M}\underbrace{0\ldots0}_{M}{}_2 \end{array}$$

Видим, что это значение записывается в двоичной системе как $N - M$ единиц, за которыми следуют M нулей. Поэтому в нашем примере $2^7 - 2^5 = 1100000_2$. Кроме того, слагаемое 2^{12} добавляет единицу в 12-м разряде, так что

$$2^{12} + 2^7 - 2^5 = 1000001100000_2.$$

Задача 2. Сколько единиц содержит двоичная запись числа

$$8^{12} + 4^{34} - 2^{18} + 120?$$

Решение. Сначала представим все числа как степени числа 2. Учитывая, что

$$120 = 128 - 8 = 2^7 - 2^3,$$

получаем

$$(2^3)^{12} + (2^2)^{34} - 2^{18} + 2^7 - 2^3 = 2^{68} + 2^{36} - 2^{18} + 2^7 - 2^3.$$

Как показано выше, пара $2^{36} - 2^{18}$ даёт $36 - 18 = 18$ единиц, пара $2^7 - 2^3$ добавляет четыре единицы, и ещё одну единицу добавляет слагаемое 2^{68}. Поэтому заданное число содержит $18 + 4 + 1 = 23$ единицы.

Ответ: 23 единицы.

Задача 3. Сколько значащих нулей содержит двоичная запись числа

$$8^{128} + 4^{344} - 2^{136} - 70?$$

Решение. Для того чтобы определить число значащих нулей, можно вычесть количество единиц из общей длины двоичной записи числа. Запишем все слагаемые как степени двойки, представив число 70 в виде $70 = 64 + 8 - 2 = 2^6 + 2^3 - 2^1$:

$$(2^3)^{128} + (2^2)^{344} - 2^{136} - 2^6 - 2^3 + 2^1 =$$
$$= 2^{688} + 2^{384} - 2^{136} - 2^6 - 2^3 + 2^1.$$

Старшая степень двойки равна 688, к ней добавляется некоторое число, поэтому запись числа в двоичной системе содержит 689 цифр (единиц и нулей).

Теперь рассмотрим «цепочку вычитания» $2^{384} - 2^{136} - 2^6 - 2^3$. Разность $2^{384} - 2^3$ в двоичной системе счисления запишется как $384 - 3 = 381$ единица, за которой следуют 3 нуля. При вычитании 2^{136} и 2^6 две из 381 единиц заменяются на нули, так что остаётся 379 единиц.

Слагаемые $2^{688} + 2^1$ дают ещё две единицы, так что общее число единиц равно 381, а число значащих нулей равно $689 - 381 = 308$.

Ответ: 308 значащих нулей.

Дробные числа

Для перевода дробного числа в двоичную систему используется общий подход: нужно умножать число на 2, запоминать целую часть и отбрасывать её перед следующим умножением. Например, для числа 0,8125 получаем:

Вычисления	Целая часть	Дробная часть
$0{,}8125 \cdot 2 = 1{,}625$	1	0,625
$0{,}625 \cdot 2 = 1{,}25$	1	0,25
$0{,}25 \cdot 2 = 0{,}5$	0	0,5
$0{,}5 \cdot 2 = 1$	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 для записи десятичного числа 0,6 в двоичной системе счисления требуется бесконечное число разрядов. Поскольку реальный компьютер не может иметь бесконечную память, число 0,6 в двоичном представлении хранится в памяти с ошибкой[1] (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 памяти компьютера, отведённой на хранение числа. В любой системе счисления существуют бесконечные дроби, которые не могут быть точно представлены конечным числом разрядов.

[1] Последний стандарт кодирования вещественных чисел IEEE 754-2008 позволяет записывать в память числа в десятичном виде. Однако вычисления с такими данными выполняются сложнее и медленнее, чем с двоичными. Кроме того, проблема конечного числа разрядов остаётся: десятичная дробь, равная $^1/_3$, по-прежнему не может быть точно представлена в памяти компьютера.

Обеспечение точности расчётов с дробными (вещественными) числами — это очень важная и актуальная проблема, пока до конца не решённая. Поэтому сначала надо попытаться решить задачу, используя только операции с целыми числами. Например, пусть требуется проверить, верно ли, что $A < \sqrt{B}$, где A и B — целые неотрицательные 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 всё-таки нужно обязательно использовать дробные числа и нельзя жертвовать точностью, приходится хранить их в нестандартном виде, например в виде отношения целых чисел (0,6 = 6/10). При вычислениях отдельно работают с числителями и знаменателями простых дробей, переходя к вещественным числам только при выводе конечного результата. Этот подход применяется в программных системах символьных вычислений, например, в *Maple* (**www.maplesoft.com**) и *Mathematica* (**www.wolfram.com**). Однако выполнение таких расчётов занимает очень много времени.

Двоичная система: достоинства и недостатки

Двоичная система служит основой всех расчётов 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;
- **надёжность и защита от помех** при передаче информации (для приёма двоичного кода не нужно измерять сигнал, а достаточно знать, какое из двух значений он принимает в каждый заданный момент времени);
- компьютеру **проще выполнять вычисления** с двоичными числами, нежели с десятичными; например, умножение на однозарядное двоичное число сводится либо к обнулению (при умножении на 0), либо к копированию (при умножении на 1).

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

- двоичная запись чисел получается **длинной**: например, число 1024 записывается в виде 10000000000_2 — здесь легко перепутать количество идущих подряд нулей;
- запись **однородна**, т. е. содержит только нули и единицы; поэтому при работе с двоичными числами легко запутаться и ошибиться.

Выводы

- Чтобы перевести число в двоичную систему, нужно представить его как сумму степеней числа 2.
- Чтобы перевести число из двоичной системы в десятичную, нужно записать его в развёрнутой форме как сумму степеней числа 2 и выполнить сложение.
- Последняя цифра двоичной записи целого числа — это остаток от деления этого числа на 2. Двоичная запись чётного числа заканчивается на 0, а нечётного — на 1. Если число делится на 4, то его двоичная запись заканчивается на два нуля.
- Большинство дробных чисел хранится в памяти с некоторой ошибкой. При выполнении вычислений с дробными числами ошибки накапливаются и могут существенно влиять на результат.
- Человеку неудобно работать с данными, записанными в двоичной системе счисления, потому что запись больших чисел получается длинной и однородной (содержит только 0 и 1).

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

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 (если есть такая возможность)?
3. Как можно работать с дробными числами, не теряя точности? В чём недостатки такого подхода?
4. Сравните преимущества и недостатки использования двоичной системы счисления с точки зрения человека и с точки зрения компьютера.

§ 10
Восьмеричная система счисления

Ключевые слова:

- триада

Восьмеричная система (система с основанием 8, использующая цифры от 0 до 7) применяется для краткой записи двоичных кодов.

Для перевода десятичного числа в восьмеричную используют общий алгоритм для позиционных систем (деление на 8, выписывание остатков в обратном порядке).

Для перевода числа 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

$$144_8 = 1 \cdot 8^2 + 4 \cdot 8^1 + 4 \cdot 8^0 = 64 + 4 \cdot 8 + 4 = 100.$$

Связь с двоичной системой счисления

Пусть требуется перевести число из восьмеричной системы в двоичную. Конечно, можно перевести число сначала в десятичную систему, а потом — в двоичную. Но для этого требуется выполнить две непростые 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 трёхзначного восьмеричного числа abc_8, где a, b и c — восьмеричные цифры. Запишем число abc_8 в развёрнутой форме:

$$abc_8 = a \cdot 8^2 + b \cdot 8^1 + c \cdot 8^0 = a \cdot 2^6 + b \cdot 2^3 + c \cdot 2^0.$$

Теперь переведём отдельно каждую восьмеричную цифру в двоичную систему. Поскольку три двоичных разряда позволяют записать все числа от 0 до 7 (и никаких других чисел!), каждая восьмеричная цифра может быть записана в виде **триады** (группы из трёх цифр) в двоичной системе (табл. 2.1).

$$\begin{aligned} a = (a_2a_1a_0)_2 &= a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0, \\ b = (b_2b_1b_0)_2 &= b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0, \\ c = (c_2c_1c_0)_2 &= c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0. \end{aligned}$$

Таблица 2.1

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

Здесь a_i, b_i, и c_i ($i = 0, 1, 2$) — это двоичные разряды (0 или 1). Тогда получаем:

$$abc_8 = (a_2 \cdot 2^2 + a_1 \cdot 2^1 + a_0 \cdot 2^0) \cdot 2^6 + (b_2 \cdot 2^2 + b_1 \cdot 2^1 + b_0 \cdot 2^0) \cdot 2^3 + (c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0) \cdot 2^0.$$

Раскрывая скобки, мы получим разложение исходного числа по степеням двойки, т. е. его запись в двоичной системе счисления:

$$abc_8 = \underbrace{a_2 \cdot 2^8 + a_1 \cdot 2^7 + a_0 \cdot 2^6}_{a} + \underbrace{b_2 \cdot 2^5 + b_1 \cdot 2^4 + b_0 \cdot 2^3}_{b} + \underbrace{c_2 \cdot 2^2 + c_1 \cdot 2^1 + c_0 \cdot 2^0}_{c}$$

Таким образом, $abc_8 = (a_2a_1a_0\,b_2b_1b_0\,c_2c_1c_0)_2$. Это значит, что каждая цифра восьмеричного числа при переводе в двоичную систему записывается как группа из трёх двоичных разрядов (**триада**) независимо от других цифр.

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.
2. Соединить триады в одно «длинное» двоичное число.

Например, для числа 35721_8 сразу получаем:

$$35721_8 = 11\,101\,111\,010\,001_2.$$

В этой записи триады специально отделены друг от друга пробелами. Обратите внимание, что все триады дополнены спереди нулями до трёх цифр:

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

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

А если разбить двоичное число на триады, то можно быстро перевести его в восьмеричную систему счисления. Именно благодаря этому свойству восьмеричную систему удобно использовать для сжатой записи двоичных кодов.

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 (отдельно) в восьмеричную систему счисления[1)].
3. Соединить полученные цифры в одно «длинное» число.

Переведём в восьмеричную систему число 1010011100101110111_2. Разобьём его на триады (начиная справа), в начало числа нужно добавить два нуля (они подчёркнуты):

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

Далее по табл. 2.1 переводим каждую триаду отдельно в восьмеричную систему:

$$1010011100101110111_2 = 1234567_8.$$

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

Арифметические операции

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.

Заём из старшего разряда равен $10_8 = 8$, а все «промежуточные» разряды заполняются цифрой 7 — старшей цифрой системы счисления. Приведём примеры сложения и вычитания:

$$\begin{array}{r} 111 \\ 356_8 \\ +\ 4662_8 \\ \hline 5240_8 \end{array} \qquad \begin{array}{l} 6+2=1\cdot 8+\textcircled{0} \\ 5+6+1=1\cdot 8+\textcircled{4} \\ 3+6+1=1\cdot 8+\textcircled{2} \\ 0+4+1=\textcircled{5} \end{array}$$

$$\begin{array}{r} \overset{\cdot\,\cdot}{456_8} \\ -\ 277_8 \\ \hline 157_8 \end{array} \qquad \begin{array}{l} (6+8)-7=\textcircled{7} \\ (5-1+8)-7=\textcircled{5} \\ (4-1)-2=\textcircled{1} \end{array}$$

В примере на сложение запись $1 \cdot 8 + 2$ означает, что получилась сумма, большая 7, которая не помещается в один разряд.

[1)] Заметим, что значение цифры в восьмеричной системе счисления совпадает со значением этой же цифры в десятичной системе.

Единица идёт в перенос, а двойка остаётся в этом разряде. При выполнении вычитания запись «–1» означает, что из этого разряда раньше был заём (его значение уменьшилось на 1), а запись «+8» — заём из старшего разряда.

Применение

С помощью восьмеричной системы удобно кратко записывать содержимое областей памяти, в которых количество бит кратно трём. 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 для кодирования команд в компьютерах 1950–1980-х годов, например в американской серии PDP, советских компьютерах серий ДВК, СМ ЭВМ, БЭСМ.

Сейчас восьмеричная система применяется 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.

Выводы

- Восьмеричная система счисления используется для сжатой записи двоичных кодов.
- При переводе числа из восьмеричной системы в двоичную каждая цифра отдельно представляется в виде триады — группы из трёх двоичных разрядов.
- При переводе числа из двоичной системы в восьмеричную двоичная запись разбивается на триады, начиная с конца, и каждая триада отдельно переводится в одну восьмеричную цифру.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы думаете, из каких других систем счисления возможен быстрый переход к двоичной системе и обратно?
2. Сколько существует различных двузначных восьмеричных чисел? Трёхзначных восьмеричных чисел?

§ 11
Шестнадцатеричная система счисления

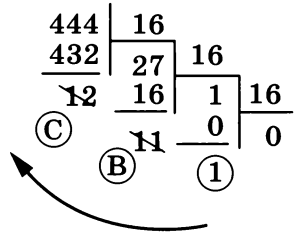
Ключевые слова:

- тетрада

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

$$A = 10,\ B = 11,\ C = 12,\ D = 13,\ E = 14 \text{ и } F = 15.$$

Для перевода чисел из десятичной системы в шестнадцатеричную используют деление на 16 и взятие остатков. Все остатки, бóльшие 9, нужно заменить на буквы:



$444 = 1BC_{16}$

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

разряды → 2 1 0

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

Можно также использовать схему Горнера:

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

Связь с двоичной системой счисления

Основания двоичной и шестнадцатеричной систем связаны соотношением $2^4 = 16$, поэтому можно переводить числа из шестнадцатеричной системы в двоичную напрямую. Алгоритмы перевода чисел из шестнадцатеричной системы в двоичную и обратно аналогичны соответствующим алгоритмам для восьмеричной системы. Каждая шестнадцатеричная цифра представляется в виде **тетрады**

(группы из четырёх двоичных цифр) в двоичной системе счисления (табл. 2.2).

Таблица 2.2

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

Переведём 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 четырёх знаков.

При обратном переводе нужно разбить двоичное число на тетрады, *начиная справа*, и каждую тетраду отдельно перевести в двоичную систему:

$$10000100001010101111100_2 =$$
$$= 10\ 0001\ 0000\ 1010\ 1011\ 1100_2 = 210ABC_{16}.$$

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

Арифметические операции

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:

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

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

$$\begin{array}{rcl} C5B_{16} & \dot{1}\dot{2}\ \dot{5}\ 11 & (11+16)-14=13=\textcircled{D} \\ A7E_{16} & 10\ 7\ 14 & (5-1+16)-7=13=\textcircled{D} \\ \hline 1DD_{16} & 1\ 13\ 13 & (12-1)-10=\textcircled{1} \end{array}$$

Если нужно работать с числами, записанными в разных системах счисления, их сначала приводят к какой-нибудь одной системе. Например, пусть 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 трудоёмок. Практика показывает, что больше всего ошибок делается при вычислениях именно в двоичной системе, поэтому лучше выбирать восьмеричную или шестнадцатеричную систему. Переведём число 53_8 в шестнадцатеричную систему через двоичную:

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

Теперь выполним сложение:

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

и переведём результат в двоичную систему:

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

Применение

Шестнадцатеричная система оказалась очень удобной для записи значений ячеек памяти. Байт в современных компьютерах представляет собой 8 бит, т. е. 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[1)].

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

Выводы

- При переводе числа из шестнадцатеричной системы в двоичную каждая цифра отдельно представляется в виде тетрады — группы из четырёх двоичных разрядов.
- При переводе числа из двоичной системы в шестнадцатеричную двоичная запись разбивается на тетрады, начиная с конца, и каждая тетрада отдельно переводится в одну шестнадцатеричную цифру.
- Перевод чисел между шестнадцатеричной и восьмеричной системами удобнее выполнять через двоичную систему.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Почему в шестнадцатеричной системе счисления появилась необходимость использовать латинские буквы?
2. Какое минимальное основание должно быть у системы счисления, чтобы в ней могли быть записаны числа 123, 4AB, 9A3 и 8455?

§ 12
Другие системы счисления

Ключевые слова:

- троичная уравновешенная система
- трит
- двоично-десятичная система

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), 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 и отрицательных чисел и число ноль. Вот, например, все двухразрядные числа в троичной системе (табл. 2.3).

Таблица 2.3

−4	$\bar{1}\ \bar{1}$	$= (-1)\cdot 3^1 + (-1)\cdot 3^0$
−3	$\bar{1}\ 0$	$= (-1)\cdot 3^1 + 0\cdot 3^0$
−2	$\bar{1}\ 1$	$= (-1)\cdot 3^1 + 1\cdot 3^0$
−1	$0\ \bar{1}$	$= 0\cdot 3^1 + (-1)\cdot 3^0$
0	0 0	$= 0\cdot 3^1 + 0\cdot 3^0$
1	0 1	$= 0\cdot 3^1 + 1\cdot 3^0$
2	$1\ \bar{1}$	$= 1\cdot 3^1 + (-1)\cdot 3^0$
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 развёрнутой 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 по сравнению с двоичным кодированием, при котором для хранения отрицательных чисел пришлось изобретать специальный код.

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

> Найти такой набор из 4 гирь, чтобы с их помощью на чашках 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 трёх состояниях:

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

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

Двоично-десятичная система счисления

Существует ещё 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:

$$\begin{matrix} 9024{,}19 = & 1001 & 0000 & 0010 & 0100, & 0001 & 1001_{ДДС}. \\ & 9 & 0 & 2 & 4 & 1 & 9 \end{matrix}$$

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

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

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

$10101{,}1_{ДДС} = 15{,}8;$

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

Использование ДДС даёт 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;
- конечные десятичные дроби записываются точно, без ошибки, так что вычисления в ДДС (вместо двоичной системы) дадут тот же результат, что и ручные расчёты 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.

Выводы

- В троичной уравновешенной системе основание равно 3, используются три цифры: «минус 1», 0 и 1. Один троичный разряд называется тритом.

- В двоично-десятичной системе счисления каждая цифра десятичного числа отдельно записывается в виде четырёх двоичных разрядов.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

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?
3. Попробуйте сформулировать правила сложения чисел в троичной уравновешенной системе.
4. Сравните троичную уравновешенную систему счисления с двоичной. Как вы думаете, почему разработчики компьютеров всё-таки выбрали двоичный код для хранения данных? Найдите сведения об этом в дополнительных источниках.
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 чём преимущества и недостатки каждой из них?

Подготовьте сообщение

а) «Троичная уравновешенная система счисления»
б) «ЭВМ "Сетунь"»
в) «Факториальная система счисления»
г) «Фибоначчиева система счисления»

Проекты

а) Программа для работы с числами в троичной уравновешенной системе счисления
б) Программа для работы с числами в факториальной системе счисления
в) Программа для работы с числами в фибоначчиевой системе счисления

§ 13
Кодирование текстов

Ключевые слова:

- текстовый файл
- шрифт
- внедрение шрифтов
- кодировка ASCII
- кодовая страница
- UNICODE

Текст в памяти компьютера хранится как последовательность двоичных кодов символов. Изображения символов хранятся в отдельных шрифтовых файлах.

В текстовых файлах (которые не содержат оформления, например, в файлах с расширением *txt*) хранятся не изображения символов, а их коды. Откуда же компьютер берёт 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[1]. Именно эти изображения и выводятся на экран. Это значит, что при изменении шрифта текст на экране может выглядеть совсем по-другому. Например, многие шрифты не содержат изображений русских букв. Поэтому, когда вы передаёте (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.

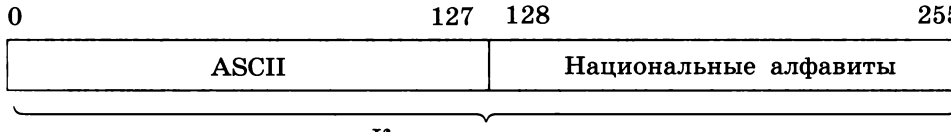
Однобайтные кодировки

Международным стандартом является 7-битная кодировка ASCII, которая определяет коды 0–127 для 128 символов, в том числе латинских букв, цифр, знаков препинания, знаков арифметических операций, скобок и др.

В современных компьютерах минимальная единица памяти, имеющая собственный адрес, — это байт (8 бит). Поэтому для хранения кодов ASCII в памяти можно добавить к ним ещё один

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

(старший) нулевой бит, таким образом, получая 8-битную кодировку. Дополнительный бит можно использовать: он даёт возможность добавить в таблицу ещё 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).



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 *Unix*-совместимых операционных системах и электронной почте;
- альтернативная кодировка **CP-866** — в системе *MS DOS*;
- кодовая страница **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 используют кодовые страницы Windows-1251 и KOI8-R. Браузер после загрузки страницы пытается автоматически определить её кодировку. Если ему это не удаётся, вы увидите странный набор букв вместо понятного русского текста. В этом случае нужно сменить кодировку вручную с помощью меню *Вид* браузера.

Главный недостаток однобайтных кодировок заключается в том, что в одном документе можно использовать только 256 символов (включая 128 символов ASCII).

Стандарт UNICODE

Для того чтобы в одном документе можно было использовать более 256 символов, в 1991 году был принят стандарт кодирования символов UNICODE, который включает знаки любых существующих (и даже некоторых мёртвых) языков, математические и музыкальные символы и др.

В современной версии UNICODE можно кодировать до 1 112 064 различных знаков, однако реально используются немногим более 137 000.

В системе Windows применяется кодировка UNICODE, называемая UTF-16 (от англ. UNICODE Transformation Format — формат преобразования UNICODE). В ней все наиболее важные символы кодируются с помощью 16 бит (2 байт), а редко используемые — с помощью 4 байт.

В Unix-подобных системах, например в Linux, чаще применяют кодировку UTF-8. В ней все символы, входящие в таблицу ASCII, кодируются с помощью 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 объём файла по сравнению с UTF-16. Текст, состоящий только из символов таблицы ASCII, кодируется точно так же, как и в кодировке ASCII. По данным поисковой системы Google, на начало 2014 года около 80% сайтов в Интернете использовали кодировку UTF-8.

Кодировки стандарта UNICODE позволяют использовать символы разных языков в одном документе, но за это приходится расплачиваться увеличением объёма файлов.

Задача 1. Автоматическое устройство осуществило перекодировку информационного сообщения на русском языке, первоначально записанного в 16-битном коде UNICODE, в 8-битную кодировку KOI8-R. При этом информационное сообщение уменьшилось на 480 бит. Какова длина сообщения в символах?

Решение. Обозначим количество символов в сообщении через L. При использовании 16-битной кодировки информационный объём сообщения равен $16 \cdot L$ бит, а после перекодирования в 8-битный код он стал равен $8 \cdot L$ бит. Таким образом, объём сообщения уменьшился на $8 \cdot L$ бит, отсюда находим: $L = 480/8 = 60$ символов.

Ответ: 60 символов.

Выводы

- В текстовых файлах содержатся только коды символов, а их изображения хранятся в памяти компьютера. Современные операционные системы загружают изображения букв из специальных шрифтовых файлов. Документы текстовых процессоров могут содержать внедрённые шрифты.
- ASCII — это 7-битная кодировка, которая является международным стандартом. Она содержит цифры, латинские буквы, скобки, знаки препинания, знаки арифметических операций и другие символы.
- Однобайтные (8-битные) кодировки (кодовые страницы) включают таблицу ASCII (символы с кодами 0..127) и дополнительную часть (символы с кодами 128...255), в которую входят символы национальных алфавитов.
- Стандарт UNICODE позволяет записывать знаки любых существующих (и даже некоторых мёртвых) языков, математические и музыкальные символы и др. (всего до 1 112 064 знаков).
- В операционной системе Linux и в Интернете часто используется кодировка UTF-8. В ней все символы, входящие в таблицу ASCII, кодируются с помощью 1 байта, а другие символы могут занимать от 2 до 4 байт.
- Для кодирования веб-страниц на русском языке используют кодировки UTF-8, Windows-1251 и KOI8-R.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Вы хотите использовать в тексте придуманный собственный символ, которого нет ни в одном шрифте. Какими путями это можно сделать?
2. Вы сами разработали шрифт и хотите переслать другу документ, в котором этот шрифт используется. Какими способами это можно сделать?
3. В чём, по вашему мнению, достоинства и недостатки внедрения шрифтов в документ?
4. Почему в современных компьютерах используются кодировки, в которых каждый символ занимает целое число байт?
5. Почему использование кодовых страниц для кодирования текста может привести к проблемам?
6. В чём достоинства и недостатки использования кодировок UNICODE?
7. Что такое UTF-16 и UTF-8? Чем различаются эти кодировки?

§ 14
Кодирование графической информации

Ключевые слова:

- пиксель
- разрешение
- цветовая модель RGB
- цветовая модель CMYK
- цветовая модель HSB
- глубина цвета
- цветовая палитра
- векторный рисунок
- кривые Безье
- 3D-графика
- рендеринг
- фрактальная графика

Изображения могут быть закодированы с помощью растрового и векторного методов.

При **растровом кодировании** рисунок разбивается на отдельные элементы (пиксели), каждый из которых закрашен одним цветом.

Пиксель — это наименьший элемент рисунка, для которого можно задать свой цвет.

Разрешение — это количество пикселей, приходящихся на единицу линейного размера изображения. Разрешение обычно измеряется в пикселях на дюйм (англ. **ppi**: *pixels per inch*).

При **векторном кодировании** рисунок представляется как множество фигур, контуры которых описываются математическими формулами.

Цветовые модели

Один из главных вопросов, которые возникают при кодировании изображений, — как хранить информацию о цвете?

Человек воспринимает свет как множество электромагнитных волн. Определённая длина волны соответствует некоторому цвету. Например, волны длиной 500–565 нм — это зелёный цвет, а волны длиной 625–740 нм — красный. 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), глаз человека содержит чувствительные элементы («колбочки») трёх типов. Каждый из них воспринимает весь поток света, но первые наиболее чувствительны в области красного цвета, вторые — в области зелёного цвета, а третьи — в области синего цвета. Цвет — это результат возбуждения всех

трёх типов рецепторов (чувствительных элементов). Поэтому считается, что любой цвет (т. е. ощущения человека, воспринимающего волны определённой длины) можно имитировать, используя только три световых луча (красный, зелёный и синий) разной яркости. Следовательно, любой цвет (в том числе и «белый») приближённо раскладывается на три составляющих — красную, зелёную и синюю. Меняя силу этих составляющих, можно составить любые цвета. Эта модель цвета получила название **RGB** по начальным буквам английских слов *Red* (красный), *Green* (зелёный) и *Blue* (синий).

В **модели RGB** (рис. 2.18 и цветной рисунок на форзаце) яркость каждой составляющей (или, как говорят, каждого канала) чаще всего кодируется целым числом[1)] от 0 до 255. При этом код цвета — это тройка чисел (R, G, B), яркости отдельных каналов.

Рис. 2.18

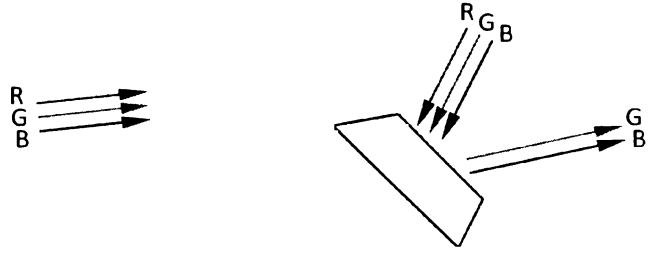
Цветовую модель RGB называют **аддитивной** (от англ. *add* — складывать), потому что нужный цвет получается «сложением» трёх базовых цветовых лучей. Так строится изображение на всех излучающих устройствах: экранах телевизоров, мониторах компьютеров и т. п. (рис. 2.19, *а* и цветной рисунок на форзаце). Модель RGB также используется в сканерах и цифровых фотоаппаратах.

Цвет (0, 0, 0) в модели RGB — это чёрный цвет, а (255, 255, 255) — белый. Если все составляющие имеют равную яркость, получаются оттенки серого цвета, от чёрного до белого. Чтобы сделать светло-красный (розовый) цвет, нужно в красном цвете (255, 0, 0) одинаково увеличить яркость зелёного и синего каналов, например цвет (255, 150, 150) — это розовый. Равномерное уменьшение яркости всех каналов делает цвет темнее, например цвет с кодом (100, 0, 0) — тёмно-красный.

[1)] Или дробным числом от 0 до 1.

При кодировании цвета на веб-странице яркости каналов записывают 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.

Когда мы смотрим на изображение, отпечатанное на бумаге (рис. 2.19, *б* и цветной рисунок на форзаце), ситуация совершенно другая. Мы видим не прямые лучи источника, попадающие в глаз, а *отражённые* от поверхности.



а *б*

Рис. 2.19

«Белый свет» от какого-то источника (солнца, лампочки), содержащий волны во всём видимом диапазоне, попадает на бумагу, на которую 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 зелёные — мы видим голубой цвет. В этом смысле красный и голубой цвета *дополняют* друг друга, так же как и пары «зелёный — пурпурный» и «синий — жёлтый». Действительно, если из белого цвета (его RGB-код #FFFFFF) «вычесть» зелёный, то получится цвет #FF00FF (пурпурный), а если «вычесть» синий, то получится цвет #FFFF00 (жёлтый).

На трёх *дополнительных цветах* — голубом, пурпурном и жёлтом — строится цветовая модель **CMY** (англ. *Cyan* — голубой, *Magenta* — пурпурный, *Yellow* — жёлтый), 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 нанести на бумагу краску голубого цвета, красные лучи поглощаются, остаются только синие и зелёные (см. рис. 2.19, *б* и цветной рисунок на форзаце). Если сверху нанести ещё жёлтую краску. которая поглощает синие

лучи, остаётся только зелёный. Такая модель называется **субтрактивной** (от англ. *subtract* — вычитать) — рис. 2.20 и цветной рисунок на форзаце.

Рис. 2.20

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

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

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, чёрного 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 трёх датчиков, принимающих лучи в красном, зелёном и синем диапазонах. Современные форматы графических файлов (например, формат PSD программы *Adobe Photoshop*) вместе с кодами пикселей хранят и профиль монитора, на котором создавался рисунок.

К сожалению, не все цвета RGB-модели могут быть напечатаны на бумаге. 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[1] 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.

Кроме цветовых моделей RGB и CMY (CMYK) существуют и другие. Наиболее интересная из них — модель **HSB**[2] (англ. ***H**ue* — тон, оттенок; ***S**aturation* — насыщенность, ***B**rightness* — яркость), которая ближе всего к естественному восприятию человека.

Тон (например, синий, зелёный или жёлтый) определяется длиной волны, он задаётся в градусах на цветовом круге (рис. 2.21 и цветной рисунок на форзаце). Насыщенность — это чистота тона, при уменьшении насыщенности до нуля получается серый цвет. Яркость определяет, насколько цвет яркий или тёмный. Любой цвет при снижении яркости до нуля превращается в чёрный. Чтобы получить белый цвет, нужно установить нулевую насыщенность и максимальную яркость, тон тут не важен.

---

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

[2] Или **HSV** (англ. ***H**ue* — тон, оттенок; ***S**aturation* — насыщенность, ***V**alue* — величина).

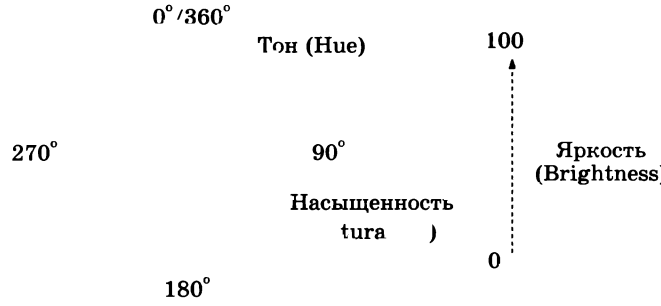


Рис. 2.21

Для кодирования «абсолютного» цвета, не зависящего от устройства, на котором он будет отображаться, применяют цветовую 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 модели RGB в модель CMYK и обратно.

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.

Растровое кодирование

Существуют два основных способа получения растровых изображений. При **вводе изображения** с помощью какого-либо устройства (сканера, цифрового фотоаппарата или веб-камеры) происходит **дискретизация** (оцифровка) — преобразование аналоговой информации в компьютерные данные. Как правило, оцифровка связана с потерей информации, поскольку требуется, чтобы каждый пиксель был залит одним цветом.

Второй способ — **создание рисунка** с помощью какой-либо компьютерной программы. В этом случае мы сразу строим цифровое изображение. Появилось даже отдельное направление в искусстве, которое называется «цифровая живопись» — создание электронных картин с помощью компьютерных технологий.

Пиксели растрового изображения обычно имеют квадратную форму, размеры рисунка (ширина и высота) также задаются в пикселях.

Для каждого пикселя в памяти хранится код его цвета. В модели RGB обычно используется по 256 вариантов яркости каждого из трёх цветов. Это позволяет закодировать $256^3 = 16\,777\,216$

оттенков, что более чем достаточно для человека. Так как $256 = 2^8$, каждая из трёх составляющих цвета занимает в памяти 8 бит (1 байт), а вся информация о каком-то цвете — 24 бита (3 байта).

Глубина цвета — это количество бит, используемых для кодирования цвета пикселя.

24-битное кодирование цвета часто называют режимом **истинного цвета** (англ. *True Color* — истинный цвет).

Задача 1. Определите информационный объём рисунка размером 20 × 30 пикселей, закодированного в режиме истинного цвета.

Решение. В режиме истинного цвета на каждый пиксель выделяется $i = 3$ байта. Количество пикселей изображения можно вычислить как произведение его размеров: $K = 20 \cdot 30 = 600$. Поэтому информационный объём рисунка равен $I = K \cdot i = 20 \cdot 30 \cdot 3$ байт = = 1800 байт.

Ответ: 1800 байт.

Конечно, при решении этой задачи мы не учитывали *сжатие* — уменьшение объёма файла с помощью специальных алгоритмов, — которое применяется во всех современных форматах графических файлов. Кроме того, в файлах всегда 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 составляющие отводится по 5 бит, а на зелёную, к которой человеческий глаз более чувствителен, — 6 бит. В режиме High Color можно закодировать $2^{16} = 65\,536$ различных цветов. Иногда применяют 12-битное кодирование цвета (4 бита на канал, 4096 цветов).

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

Если количество цветов в изображении невелико (не более 256), применяют **кодирование с палитрой. Цветовая палитра** — это таблица, в которой каждому цвету, заданному в виде составляющих в модели RGB, сопоставляется числовой код.

Палитра хранится в заголовке файла. Если рисунок содержит пиксели четырёх цветов — чёрного, красного, синего и белого, то его палитра содержит четыре трёхбайтных блока, в которых записаны RGB-коды этих цветов:

0	0	0	255	0	0	0	0	255	255	255	255
цвет $0 = 00_2$			цвет $1 = 01_2$			цвет $2 = 10_2$			цвет $3 = 11_2$		

Таким образом, размер палитры в байтах вычисляется как $3 \cdot N$, где N — количество цветов в палитре.

Для каждого пикселя изображения в памяти хранится не RGB-код цвета, а номер (код) соответствующего блока в палитре. Так как мы используем всего $2^2 = 4$ цвета, этот код занимает всего два бита, так что глубина цвета для такого рисунка составляет $i = 2$ бита на пиксель.

Задача 2. Какой минимальный объём памяти (в Кбайт) нужно зарезервировать, чтобы можно было сохранить любое растровое изображение размером 64×128 пикселей при условии, что в изображении могут использоваться 256 различных цветов?

Решение. Сначала находим количество пикселей K, перемножив ширину и высоту рисунка в пикселях. При этом удобно проводить все вычисления через степени числа 2:

$$K = 64 \cdot 128 = 2^6 \cdot 2^7 = 2^{13}.$$

Так как $256 = 2^8$, глубина цвета составляет $i = 8 = 2^3$ бит на пиксель. Поэтому информационный объём файла в битах равен $I = 2^{13} \cdot 2^3 = 2^{16}$ бит. Остаётся перевести его в килобайты:

$I = 2^{16} : 2^{13}$ Кбайт $= 2^3$ Кбайт $= 8$ Кбайт.

Ответ: 8 Кбайт.

Заметим, что мы вычислили только размер памяти, необходимый для хранения кодов пикселей (без учёта заголовка с палитрой).

Задача 3. Рисунок размером 512×256 пикселей занимает в памяти 64 Кбайт (без учёта сжатия). Найдите максимально возможное количество цветов в палитре изображения.

Решение. Максимально возможное количество цветов в палитре определяется глубиной цвета — количеством бит на пиксель. Чтобы найти глубину цвета, нужно разделить информационный объём изображения I на количество пикселей K.

Находим количество пикселей, перемножив размеры рисунка:

$$K = 512 \cdot 256 = 2^9 \cdot 2^8 = 2^{17}.$$

Информационный объём данных в битах равен

$$I = 64 \text{ Кбайт} = 2^6 \cdot 2^{13} \text{ бит} = 2^{19} \text{ бит}.$$

Теперь можно вычислить глубину цвета: $i = I : K = 2^{19} : 2^{17} = 4$ бита на пиксель. В этом случае максимально возможное количество цветов в палитре изображения равно $2^4 = 16$.

Ответ: 16 цветов.

Растровое кодирование — это универсальный метод, позволяющий кодировать любые изображения, в том числе размытые (например, фотографии). В то же время у него есть существенные **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 объём.

Форматы файлов

Существует много разных форматов хранения растровых рисунков. В большинстве из них используется сжатие, т. е. размер файла уменьшают с помощью специальных алгоритмов. В некоторых форматах применяют сжатие без потерь, при котором исходный рисунок можно в точности восстановить из сжатого состояния. Ещё бóльшую степень сжатия можно обеспечить, используя сжатие с потерями, при котором незначительная часть данных (почти не влияющая на восприятие рисунка человеком) теряется. Подробно мы изучим эти вопросы в 11 классе.

Чаще всего встречаются следующие форматы файлов:

- **BMP** (англ. *bitmap* — битовая карта, файлы с расширением *bmp*) — стандартный формат растровых изображений в операционной системе *Windows*; поддерживает кодирование с палитрой и в режиме истинного цвета;
- **JPEG** (англ. *Joint Photographic Experts Group* — объединённая группа фотографов-экспертов, файлы с расширением *jpg* или *jpeg*) — формат, разработанный специально для кодирования фотографий; поддерживает только режим истинного цвета; для уменьшения объёма файла используется сильное сжатие, при котором изображение немного размывается, поэтому не рекомендуется использовать его для рисунков с чёткими границами объектов;
- **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, т. е. на веб-странице через них будет «просвечивать» фон; в современном варианте формата GIF можно хранить анимированные изображения; используется сжатие без потерь;

- **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 четвёртый байт задаёт 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	—

Заголовок файла

Вы знаете, что все виды информации хранятся в памяти компьютера как двоичные коды, т. е. цепочки 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 для рисунка с палитрой 256 цветов, цвета 8 пикселей чёрно-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 данные на экран.

Векторное кодирование

Для чертежей, схем, карт применяется другой способ кодирования, который позволяет не терять качество при изменении размеров изображения.

Векторный рисунок — это рисунок, построенный из простейших геометрических фигур **(графических примитивов)**: линий, многоугольников, сглаженных кривых, окружностей, эллипсов. Их параметры (координаты вершин, цвет контура и заливки) хранятся в виде чисел.

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

Рис. 2.22

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 2.23 изображена линия с опорными точками *А*, *Б*, *В*, *Г* и *Д*.

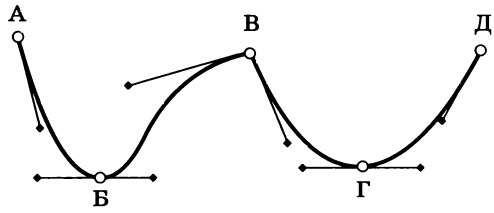


Рис. 2.23

У каждой из этих точек есть «рычаги» (*управляющие линии*), перемещая концы этих рычагов, можно регулировать наклон

касательной и кривизну всех участков кривой. Если оба рычага находятся на одной прямой, получается сглаженный узел (*Б* и *Г*), если нет — то угловой узел (*В*). Таким образом, форма этой кривой полностью задаётся координатами опорных точек и координатами рычагов. Кривые, заданные таким образом, называют *кривыми Безье* в честь их изобретателя французского инженера Пьера Безье.

Векторный способ кодирования рисунков обладает значительными **преимуществами** по сравнению с растровым:

- если изображение (например, чертёж, схема, карта, диаграмма) может быть полностью разложено на простейшие геометрические фигуры, то при векторном кодировании нет потери информации;
- объём файлов напрямую зависит от сложности рисунка — чем меньше элементов, тем меньше места занимает файл. Как правило, векторные рисунки значительно меньше по объёму, чем растровые;
- при изменении размера векторного рисунка не происходит никакого искажения формы элементов, при увеличении наклонных линий не появляются «ступеньки», как при растровом кодировании (рис. 2.24).

Растровый рисунок Векторный рисунок

Рис. 2.24

Самый главный **недостаток** этого метода — он практически непригоден для кодирования изображений, в которых объекты не имеют чётких границ, например для фотографий.

Среди форматов векторных рисунков отметим следующие:

- **WMF** (англ. *Windows Metafile* — метафайл *Windows*, файлы с расширениями *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 в Интернете.

Трёхмерная графика

Сейчас инженеры разрабатывают новые автомобили, самолёты, приборы в системах автоматизированного проектирования. В них строится трёхмерная (объёмная) модель объекта, которую затем можно просматривать с разных сторон и рассчитывать на прочность. Для создания объёмных моделей объектов (рис. 2.25) применяют **трёхмерную графику (3D-графику**, от англ. *3-dimensional* — трёхмерный).

Рис. 2.25

Трёхмерные модели хранятся в памяти компьютера как элементарные фигуры (отрезки, треугольники, четырёхугольники и др.) и поверхности, которые описываются математическими формулами. С этой точки зрения трёхмерная графика — это векторная графика.

Каркас объекта обычно строится из многоугольников (*полигонов*), потом его углы сглаживаются. На поверхности наносят рисунки, имитирующие реальные материалы, настраивают их свойства: цвет, прозрачность, блики и др. Затем устанавливают источники света, задают свойства атмосферы, в которой находится объект.

Для того чтобы построить двумерную картинку (проекцию трёхмерной модели на плоскость), нужно выбрать точку наблюдения («установить камеру») и просчитать, как выглядит модель с этой точки. Эта процедура называется *рендеринг*. Быстрая смена таких картинок позволяет строить анимацию — создавать иллюзию движения и изменения.

Фрактальная графика

Существует ещё один интересный вид графики — фрактальная графика. Слово «фрактал» образовано от латинского *fractus* и в переводе означает «состоящий из фрагментов». Фрактал обладает *самоподобием*, т. е. основная фигура состоит из нескольких таких же, только меньшего размера. Такие рисунки хранятся в памяти не как отдельные элементы (пиксели или графические примитивы), а в виде математической формулы и алгоритма построения. На рисунке 2.26 показаны два известных фрактала: дерево Пифагора (*а*) и множество Мандельброта (*б*) — см. цветной рисунок на форзаце.

а

б

Рис. 2.26

Фрактальная графика применяется для построения изображений растений, облаков, гор, водных поверхностей, а также для оформления рекламных листовок и веб-сайтов.

Выводы

- При растровом кодировании изображение разбивается на пиксели — элементы, для каждого из которых можно задать свой цвет независимо от других.
- Глубина цвета — это количество бит, используемых для кодирования цвета пикселя.
- Качество растрового кодирования зависит от разрешения и глубины цвета.
- Цвет пикселя при выводе на экран монитора кодируется в модели RGB — раскладывается на красную, зелёную и синюю составляющие.
- При выводе на печать используется цветовая модель CMYK (голубой, пурпурный, жёлтый, чёрный).
- При растровом кодировании, как правило, происходит потеря информации из-за дискретизации. Растровый рисунок искажается, когда его размеры изменяются.
- Векторный рисунок хранится в памяти как описание на специальном языке множества геометрических фигур с заданными свойствами контура и заливки внутренней области.
- При изменении размеров векторного рисунка он не искажается.
- Трёхмерные модели (3D-модели) хранятся в памяти компьютера как элементарные фигуры и поверхности, которые описываются математическими формулами.
- Фрактал — это фигура, обладающая самоподобием, т. е. основная фигура состоит из нескольких таких же, только меньшего размера. Фракталы хранятся в памяти компьютера в виде математической формулы и алгоритма построения.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы думаете, почему не удаётся придумать единый метод кодирования рисунков, пригодный во всех ситуациях?
2. Как уменьшить потерю информации при дискретизации? Что при этом ухудшается?
3. Как связаны глубина цвета и объём файла?
4. Сравните режим истинного цвета и кодирование с палитрой.
5. Сравните растровой и векторный способы кодирования изображений.

6. В каких форматах целесообразно сохранять фотографии? Рисунки с чёткими границами объектов?
7. Как можно уменьшить объём файла, в котором хранится рисунок? Чем при этом придётся пожертвовать?
8. Как компьютер определяет, что находится в файле — текст, рисунок, звук или видео?
9. Узнайте, с каких символов начинается заголовок файла в формате PNG.

Подготовьте сообщение

а) «Цветовые модели XYZ и Lab»
б) «Фракталы»
в) «Сплайны»

Проекты

а) Сравнение форматов для хранения изображений
б) Программа для построения фракталов

§ 15 Кодирование звуковой и видеоинформации

Ключевые слова:

- оцифровка
- звуковая карта
- частота дискретизации
- разрядность кодирования
- кодек
- потоковый формат
- стандарт MIDI
- синтезатор
- сэмпл
- синхронность
- артефакты

Для кодирования звука используются два метода: оцифровка и инструментальное кодирование.

Оцифровка — это преобразование аналогового сигнала в цифровой код (последовательность чисел). При **инструментальном кодировании** в памяти компьютера хранится нотная запись мелодии и коды музыкальных инструментов.

Звук — это колебания среды (воздуха, воды). С помощью микрофона звук преобразуется в **аналоговый** электрический сигнал, который в любой момент времени может принимать любое значение в некотором интервале. Этот сигнал можно подать на вход звуковой карты, где специальное устройство — аналого-цифровой преобразователь (АЦП) — преобразует его в цифровой код. Процессор компьютера может затем обработать этот код по некоторому алгоритму, сохранить в файле и т. д. (рис. 2.27).

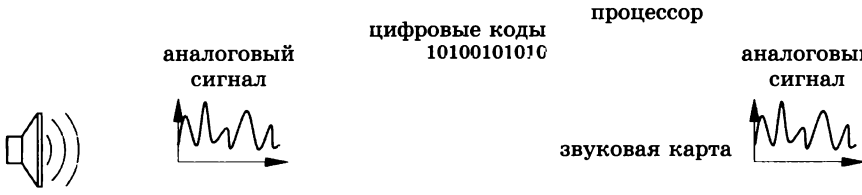


Рис. 2.27

Для проигрывания звука через наушники или звуковые колонки (это аналоговые устройства!), цифровой код из памяти компьютера (например, из файла) передаётся звуковой карте, где с помощью цифро-аналогового преобразователя (ЦАП) преобразуется в аналоговый сигнал, поступающий на устройство вывода звука.

Оцифровка звука

При оцифровке звука выполняется **дискретизация** — из всего бесконечного множества значений аналогового сигнала сохраняются в памяти только значения в отдельных точках, взятых с некоторым шагом T по времени (рис. 2.28, *а*). Это называется **дискретизацией по времени**.

Число T называется **интервалом дискретизации**, а обратная ему величина $f = 1/T$ — **частотой дискретизации**. Частота дискретизации измеряется в герцах (Гц) и килогерцах (кГц). Чем больше частота дискретизации, тем точнее мы записываем сигнал, тем меньше информации теряем. Однако при этом возрастает количество отсчётов, т. е. информационный объём закодированного звука.

Для кодирования звука в компьютерах чаще всего используются частоты дискретизации 8 кГц (минимальное качество, достаточное 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).

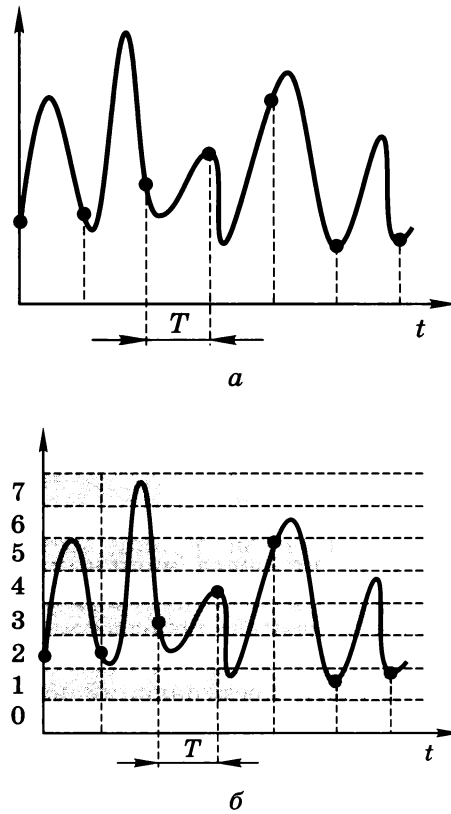


Рис. 2.28

Кроме дискретизации по времени в АЦП происходит и **дискретизация по уровню** (квантование): измеренные значения сигнала записываются в памяти как целые числа. На рис. 2.28, *б* весь диапазон значений сигнала разбит на $8 = 2^3$ одинаковых полос, что соответствует 3-битному кодированию. Все значения, попавшие в одну полосу, получают одинаковые коды.

Разрядность кодирования (глубина кодирования) — это число бит, используемое для хранения одного отсчёта.

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

Информационный объём данных, полученных в результате оцифровки звука, равен

$$I = f \cdot i \cdot t \cdot k,$$

где f — частота квантования, i — разрядность кодирования, t — время и k — число каналов, которые записываются одновременно.

Для стереофонической записи (когда отдельно записываются левый и правый каналы) нужно принять $k = 2$, а для квадрофонического звука (запись четырёх каналов одновременно) — $k = 4$.

Например, если используется 16-разрядное кодирование с частотой 44 кГц, то за 1 с выполняется 44 000 измерений сигнала, и каждое из измеренных значений занимает 16 бит (2 байта). Поэтому за 1 секунду накапливается $f \cdot i = 44000 \cdot 2 = 88\,000$ байт данных, а за 1 минуту

$$f \cdot i \cdot t = 88\,000 \cdot 60 = 5\,280\,000 \text{ байт} \approx 5 \text{ Мбайт}.$$

Если записывается стереозвук, это число нужно удвоить, а при записи квадрофонического звука — умножить на четыре.

Восстановление звукового сигнала

При проигрывании звука приходится решать сложную задачу — восстанавливать аналоговый сигнал по его дискретным значениям, взятым с некоторой частой f. С точки зрения математики, любой сигнал можно представить в виде суммы очень большого числа колебаний разных частот (гармоник). Если выбрать частоту дискретизации f больше, чем удвоенная частота самой быстрой гармоники, то теоретически по отдельным отсчётам можно точно восстановить исходный аналоговый сигнал. Этот результат известен в радиотехнике как теорема Котельникова–Шеннона.

К сожалению, на практике всё 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 объём 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 дискретизации применяют тогда, когда важно всячески уменьшать объём звуковых данных (например, для трансляции радиопередач через Интернет), даже ценой ухудшения качества.

Простейший метод восстановления сигнала по отдельным отсчётам — построить ступенчатый сигнал (рис. 2.29). В современных звуковых картах для повышения качества звука этот ступенчатый сигнал сглаживается с помощью специальных фильтров, однако восстановить точно исходный сигнал всё равно не удаётся, так как информация о значениях сигнала между моментами дискретизации была потеряна при оцифровке.

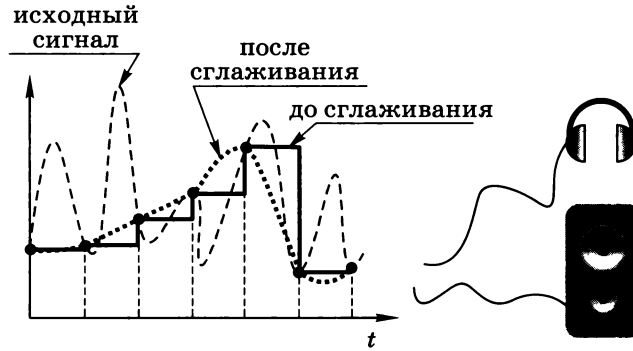


Рис. 2.29

С помощью оцифровки можно закодировать любой звук, который принимает микрофон. Однако при оцифровке звука всегда есть потеря информации (из-за дискретизации). Кроме того, звуковые файлы имеют, как правило, большой размер, поэтому в большинстве современных форматов используется сжатие. Программа, которая выполняет сжатие звуковых данных, называется **кодеком** (от англ. *coder/decoder* — кодировщик/декодировщик).

Среди форматов оцифрованных звуковых файлов наиболее известны форматы:

- **WAV** (англ. *Waveform Audio File Format*, файлы с расширением *wav*);
- **MP3** (файлы с расширением *mp3*);

- **AAC** (англ. *Advanced Audio Coding*, файлы с расширениями *aac*, *mp4*, *m4a* и др.);
- **WMA** (англ. *Windows Media Audio*, файлы с расширением *wma*);
- **Ogg Vorbis** (файлы с расширением *ogg*) — открытый формат, не требующий оплаты лицензии.

Все эти форматы — **потоковые**, т. е. можно начинать прослушивание звука до того момента, как весь файл будет получен (например, из Интернета). Как правило, в них используется *сжатие с потерями*: для значительного уменьшения объёма файла снижается качество кодирования для тех частот, которые практически неразличимы для человеческого слуха.

Инструментальное кодирование звука

Для кодирования инструментальных мелодий нередко используется стандарт **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, т. е. 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*. Для проигрывания MIDI-файла используют *синтезаторы* — электронные устройства, имитирующие звук реальных инструментов. Простейший синтезатор — звуковая карта компьютера.

Главные **достоинства** инструментального кодирования:

- кодирование мелодии (нотной записи) происходит без потери информации;
- файлы имеют значительно меньший объём 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.

Кодирование видеоинформации

Для того чтобы сохранить видео в памяти компьютера, нужно закодировать звук и изменяющееся изображение, причём при проигрывании 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.

При размере кадра 768 × 576 точек и глубине цвета 24 бита на пиксель закодированная 1 секунда видео (без звука) будет занимать примерно 32 Мбайт, а 1 минута — около 1,85 Гбайт. Это недопустимо много, поэтому в большинстве форматов видеоизображений используется сжатие. Упаковку и распаковку видеоданных выполняют программы-**кодеки**.

Основная идея сжатия видеофайлов основана на том, что за короткое время изображение изменяется очень мало, поэтому можно запомнить «базовый» кадр, а затем сохранять только изменения. Через 10–15 с изображение изменяется настолько, что необходим новый базовый кадр. Для того чтобы ещё больше уменьшить объём файла, применяют сжатие с потерями, при котором теряются некоторые детали, несущественные для восприятия человеком. При очень сильном сжатии с потерями появляются заметные искажения (**артефакты**), например становится явно видно, что изображение разбито на блоки размером 8 × 8 пикселей.

Современные цифровые видеокамеры и фотоаппараты могут записывать видео в форматах высокой чёткости с размерами изображения 1280 × 720 пикселей и 1920 × 1080 пикселей (*Full HD*).

Среди форматов видеофайлов наиболее известны:

- **AVI** (англ. *Audio Video Interleave* — чередующиеся звук и видео, файлы с расширением *avi*);

- **WMV** (англ. *Windows Media Video*, файлы с расширением *wmv*);
- **MPEG** (файлы с расширением *mpg*, *mpeg*);
- **MP4** (файлы с расширением *mp4*);
- **MOV** (англ. *Quick Time Movie*, файлы с расширением *mov*);
- **WebM** — открытый (не требующий оплаты лицензии) видеоформат, который поддерживается в современных браузерах без установки дополнительных модулей.

Выводы

- Для кодирования звука применяют оцифровку и инструментальное кодирование.
- Оцифровка — это дискретизация аналогового сигнала: в памяти сохраняются значения этого сигнала, измеряемые с некоторой частотой, которая называется частотой дискретизации.
- Качество оцифровки определяется частотой дискретизации и разрядностью кодирования (количеством бит, выделяемых для хранения одного измеренного значения).
- Инструментальное кодирование звука позволяет без искажений сохранить в памяти компьютера нотную запись, но этот метод нельзя использовать для кодирования нестандартных звуков, например речи человека.
- Кодирование видеофильмов сводится к кодированию изменяющегося изображения и звуковых дорожек. Изображение и звук должны проигрываться одновременно (синхронно).
- Для уменьшения объёма файлов при кодировании звука и видеоданных, как правило, используют сжатие с потерями.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Сравните оцифровку звука и инструментальное кодирование. Как вы думаете, почему не удаётся придумать один способ кодирования звука, который был бы лучше других во всех случаях?
2. Как связаны частота дискретизации с потерей информации и объёмом файла?
3. От чего зависит выбор частоты дискретизации? Почему частоты дискретизации более 48 кГц применяются очень редко?

4. Какие проблемы возникают при проигрывании цифрового звука?
5. Как связана разрядность кодирования звука с качеством закодированного звука и размером файла?
6. Почему мелодии, закодированные по стандарту MIDI, могут по-разному звучать на разной аппаратуре?
7. По какому принципу выбирается частота смены кадров и частота дискретизации звука при кодировании видеофильмов?
8. Какие принципы используются для сжатия видеофильмов?

Подготовьте сообщение

а) «Программы для создания MIDI-музыки»
б) «Кодирование звука в формате MP3»

Проекты

а) Сравнение форматов для хранения звука
б) Сравнение видеоформатов

ЭОР на сайте ФЦИОР (http://fcior.edu.ru)

- Представление текста в различных кодировках
- Принцип дискретного (цифрового) представления информации, системы счисления, алгоритмы
- Достоинства и недостатки двоичной системы счисления при использовании её в компьютере
- Понятие о системах счисления
- Представление числовой информации с помощью систем счисления. Алфавит, базис, основание. Свёрнутая и развёрнутая формы представления чисел
- Алгоритм перевода дробных чисел из 10-й системы счисления в P-ичную
- Алгоритм перевода целых чисел из 10-й системы счисления в P-ичную
- Арифметические операции в позиционных системах счисления
- Связь между двоичной, восьмеричной и шестнадцатеричной системами счисления
- Алгоритм перевода целых чисел из P-ичной системы счисления в 10-ю

- Аппаратное и программное обеспечение для представления изображения
- Растровая и векторная графика
- Аппаратное и программное обеспечение для представления изображения
- Аппаратное и программное обеспечение для представления звука

Практические работы к главе 2

www

Работа № 5 «Декодирование»
Работа № 6 «Необычные системы счисления»

Глава 3
ЛОГИЧЕСКИЕ ОСНОВЫ КОМПЬЮТЕРОВ

§ 16
Логические операции

Ключевые слова:

- логическое высказывание
- алгебра логики
- логическая функция
- логические операции
- базовые логические операции
- исключающее «ИЛИ»
- импликация
- эквиваленция

Логическое высказывание — это повествовательное предложение, про которое можно однозначно сказать, истинно оно или ложно.

Если обозначить истинное значение единицей, а ложное — нулём, то получится, что формальная логика представляет собой правила выполнения операций с двоичными кодами.

Алгебра логики (булева алгебра) — это математический аппарат, с помощью которого записывают, вычисляют, упрощают и преобразуют логические высказывания.

Обработку двоичных данных в компьютере можно свести к выполнению логических операций.

Логическое выражение — это символическая запись высказывания, которая может содержать логические переменные и знаки логических операций.

Логическая функция — это правило преобразования входных логических значений в выходные. Логическая функция задаётся таблицей истинности.

Любой логической функции соответствует множество логических выражений (их таблицы истинности одинаковы).

Операции НЕ, И, ИЛИ

Операция НЕ — это переход от одного логического значения к другому, от 1 к 0 или наоборот. Она называется **отрицанием** или **инверсией** (англ. *inverse* — обратный). В алгебре логики выражение «не А» записывается как $\overline{A}$ или $\neg A$, в языках программирования Паскаль и Python — как `not A`, в языках C, Java и Javascript — как `!A`. Операцию НЕ можно задать в виде таблицы истинности (рис. 3.1).

A	$\overline{A}$
0	1
1	0

Рис. 3.1

Операции, которые выполняются над одной величиной, называют *унарными* (от лат. *uno* — один) или *одноместными*. Таким образом, НЕ — это унарная логическая операция.

Операция И (в отличие от НЕ) выполняется с двумя логическими значениями, которые мы обозначим как A и B. Высказывание «A И B» истинно в том и только в том случае, когда оба высказывания, A и B, истинны одновременно.

Результат этой операции в алгебре логики записывают как $A \cdot B$, $A \wedge B$ или $A \,\&\, B$. В языках программирования используют обозначения `A` **and** `B` (Паскаль, Python) или `A && B` (C, Java, Javascript). Таблица истинности операции И приведена на рис. 3.2.

A	B	$A \cdot B$
0	0	0
0	1	0
1	0	0
1	1	1

Рис. 3.2

Легко проверить, что результат операции И можно получить «обычным» умножением A на B, поэтому операцию И называют **логическим умножением**. Кроме того, с точки зрения обычной математики эта операция выбирает *минимальное* из исходных значений. Существует ещё одно название операции И — **конъюнкция** (от лат. *conjunctio* — союз, связь).

Операция ИЛИ также выполняется с двумя логическими значениями. Высказывание *A* ИЛИ *B* истинно, когда истинно хотя бы одно из входящих в него высказываний (или оба истинны одновременно).

В алгебре логики операция ИЛИ обозначается как $A + B$ или $A \vee B$, в языках программирования — `A or B` (Паскаль, Python) или `A || B` (C, Java, Javascript).

В столбце значений в таблице истинности операции ИЛИ только один ноль — для варианта $A = B = 0$ (рис. 3.3).

A	*B*	*A + B*
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 3.3

Операцию ИЛИ называют **логическим сложением**, потому что она похожа на обычное математическое сложение. Единственное отличие — в последней строке таблицы истинности: в математике 1 + 1 равно 2, а в алгебре логики — 1. Можно считать, что в результате применения операции ИЛИ из исходных значений выбирается наибольшее. Другое название этой операции — **дизъюнкция** (от лат. *disjunctio* — разделение).

Операции И и ИЛИ, которые выполняются с двумя значениями (**операндами**), называются *бинарными* (лат. *bis* — дважды) или *двуместными*.

Доказано, что операций НЕ, И и ИЛИ достаточно для того, чтобы записать с их помощью любую логическую операцию (логическую функцию), которую только можно придумать. Поэтому эти операции иногда называют **базовыми** (основными) и говорят, что они образуют базис. Далее мы рассмотрим ещё три известные операции и покажем, как их можно представить через операции НЕ, И и ИЛИ.

Операция исключающее ИЛИ

Операция исключающее ИЛИ отличается от обычного ИЛИ только тем, что результат равен 0, если оба значения равны 1 (последняя строка в таблице истинности). То есть её результат — истина в том и только в том случае, когда два значения *не равны* (рис. 3.4).

A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 3.4

Операции исключающее ИЛИ соответствует пословица «Либо пан, либо пропал», где выполнение обоих условий одновременно невозможно.

Исключающее ИЛИ в алгебре логики обозначается знаком $\oplus$, в языке Паскаль — словом **xor** (например, `A xor B`), а в языках C, Python, Java и Javascript — знаком ^ (`A ^ B`). Эту операцию можно представить через базовые операции (НЕ, И, ИЛИ) следующим образом:

$$A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}.$$

Пока мы не можем вывести это равенство, но можем доказать его (или опровергнуть). Для этого достаточно для всех возможных комбинаций A и B вычислить значения выражения, стоящего в правой части равенства, и сравнить его со значением $A \oplus B$ для тех же исходных данных. Поскольку провести такие вычисления в уме достаточно сложно, сначала вычислим значения $\overline{A}$, $\overline{B}$, $\overline{A} \cdot B$ и $A \cdot \overline{B}$, а потом уже $\overline{A} \cdot B + A \cdot \overline{B}$. В таблице истинности появятся дополнительные столбцы для промежуточных результатов (рис. 3.5).

A	B	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot B$	$A \cdot \overline{B}$	$\overline{A} \cdot B + A \cdot \overline{B}$	$A \oplus B$
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

Рис. 3.5

Легко видеть, что выражение $\overline{A} \cdot B + A \cdot \overline{B}$ совпадает с $A \oplus B$ для всех возможных вариантов значений A и B. Это значит, что равенство доказано.

Возможен и другой вариант замены:

$$A \oplus B = (A + B) \cdot (\overline{A} + \overline{B}).$$

Докажите это равенство самостоятельно.

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 её результат равен остатку от деления арифметической суммы $A + B$ на 2:

$$A \oplus B = (A + B) \bmod 2.$$

Здесь mod обозначает операцию взятия остатка от деления. Из таблицы истинности видно, что $A \oplus B = 1$ тогда и только тогда, когда $A \neq B$.

Операция исключающее ИЛИ обладает интересными свойствами. По таблице истинности несложно проверить, что

$$A \oplus 0 = A, \quad A \oplus 1 = \overline{A}, \quad A \oplus A = 0.$$

Для доказательства этих равенств можно просто подставить в них $A = 0$ и $A = 1$.

Теперь докажем, что

$$(A \oplus B) \oplus B = A. \qquad (1)$$

Подставляя в левую часть $B = 0$, получим $(A \oplus 0) \oplus 0 = A \oplus 0 = A$. Аналогично для $B = 1$ имеем $(A \oplus 1) \oplus 1 = \overline{A} \oplus 1 = A$. Это означает, что формула (1) 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 неё обратима также операция НЕ — если применить её дважды, мы вернёмся к исходному значению).

Формулу (1) можно использовать для шифрования данных. Пусть A и B — это двоичные коды одинаковой длины. Чтобы зашифровать данные A, используя ключ B, надо применить операцию исключающее ИЛИ отдельно для каждого разряда A и B. Для расшифровки ещё раз применяется исключающее ИЛИ с тем же ключом. Нужно отметить, что такой метод шифрования очень нестойкий: для достаточно длинных текстов его легко «взломать» частотным анализом.

Импликация

Мы часто используем логическую связку «если..., то», например: «Если пойдёт дождь, то я надену плащ» или «Если все стороны прямоугольника равны, то это квадрат». В логике эта связка называется **импликацией** (*следованием*, от лат. *implicatio* — сплетение, тесная связь) и обозначается стрелкой: $A \to B$ («если A, то B», «из A следует B»). Таблица истинности операции импликации показана на рис. 3.6.

A	B	$A \to B$
0	0	1
0	1	1
1	0	0
1	1	1

Рис. 3.6

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 $A \to B$ истинно, то все, кто хорошо работают ($A = 1$), должны получать большую зарплату ($B = 1$). Если же кто-то работает хорошо ($A = 1$), а получает мало ($B = 0$), то высказывание $A \to B$ ложно.

Лодыри и бездельники ($A = 0$) могут получать как маленькую ($B = 0$), так и большую зарплату ($B = 1$), это не нарушает истинность высказывания $A \to B$. 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, т. е. именно A вызывает B. Алгебра логики не устанавливает взаимосвязь явлений; истинность высказывания $A \to B$ 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, формулировку вида «если A, то B» можно записать как $A \to B = 1$.

Для импликации (в отличие от других изученных ранее операций с двумя переменными) не действует переместительный закон: если в записи $A \to B$ поменять местами A и B, то результат изменится: $A \to B \neq B \to A$. Внешне это видно по стрелке, которая указывает «направление».

Импликацию можно заменить на выражение, использующее только базовые операции (здесь — только НЕ и ИЛИ):

$$A \to B = \overline{A} + B.$$

Доказать это равенство вы уже можете самостоятельно.

Эквиваленция

Эквиваленция (её также называют эквивалентность, равносильность, логическое равенство) — это логическая операция, которая соответствует связке «тогда и только тогда». Высказывание $A \leftrightarrow B$ (используются также обозначения $A \equiv B$ и $A \sim B$) истинно в том и только в том случае, когда A и B равны (рис. 3.7).

A	B	$A \leftrightarrow B$
0	0	1
0	1	0
1	0	0
1	1	1

Рис. 3.7

Возможно, вы заметили, что эквиваленция — это обратная операция для исключающего ИЛИ (проверьте по таблицам истинности):

$$A \leftrightarrow B = \overline{A \oplus B}.$$

Здесь черта сверху, охватывающая всё выражение в правой части равенства, означает отрицание (инверсию), которое применяется к результату вычисления выражения $A \oplus B$, а не к отдельным высказываниям.

Можно заменить эквиваленцию выражениями, которые включают только базовые логические операции:

$$A \leftrightarrow B = \overline{A} \cdot \overline{B} + A \cdot B,$$
$$A \leftrightarrow B = (A + \overline{B}) \cdot (\overline{A} + B).$$

Эти равенства вы можете доказать (или опровергнуть) самостоятельно.

Другие логические операции

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 даёт новую логическую операцию (логическую функцию). Всего их, очевидно, столько, сколько существует четырёхразрядных двоичных чисел, т. е. $16 = 2^4$. Из тех, что мы ещё не рассматривали, наиболее интересны две — **штрих Шеффера** (И-НЕ, англ. *nand* = *not and*, рис. 3.8):

$$A \mid B = \overline{A \cdot B}$$

и **стрелка Пирса** (ИЛИ-НЕ, англ. *nor*: *not or*, рис. 3.9):

$$A \downarrow B = \overline{A + B}.$$

Штрих Шеффера

A	B	$A \mid B$
0	0	1
0	1	1
1	0	1
1	1	0

Рис. 3.8

Стрелка Пирса

A	B	$A \downarrow B$
0	0	1
0	1	0
1	0	0
1	1	0

Рис. 3.9

Особенность этих операций состоит в том, что с помощью любой одной из них можно записать *произвольную* логическую операцию. Например, операции НЕ, И и ИЛИ (базовый набор) можно выразить через штрих Шеффера так:

$$\overline{A} = A|B, \quad A \cdot B = (A|B)|(A|B), \quad A + B = (A|A)|(B|B).$$

Поэтому одна операция И-НЕ тоже представляет собой базис. То же самое можно сказать и про операцию ИЛИ-НЕ.

Выводы

- Исключающее ИЛИ $A \oplus B$ принимает значение «истина» тогда и только тогда, когда значения A и B различны.
- Импликация $A \to B$ истинна во всех случаях, кроме $A = 1$ и $B = 0$.
- Эквиваленция $A \leftrightarrow B$ истинна тогда и только тогда, когда A и B имеют одинаковые значения.
- Операция исключающее ИЛИ обратима: если взять любое значение A и выполнить дважды исключающее ИЛИ с любой постоянной, то получится исходное значение A. Эту особенность можно использовать для простого шифрования.
- Логические функции $A \to B$, $A \leftrightarrow B$ и $A \oplus B$ можно выразить через базовые логические операции НЕ, И и ИЛИ:

$$A \to B = \overline{A} + B,$$

$$A \leftrightarrow B = A \cdot B + \overline{A} \cdot \overline{B} = (A + \overline{B}) \cdot (\overline{A} + B),$$

$$A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B = (A + B) \cdot (\overline{A} + \overline{B}).$$

- Логические операции И-НЕ (штрих Шеффера) и ИЛИ-НЕ («стрелка Пирса») тоже представляют собой базис: с помощью любой из них можно записать любую логическую функцию.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Дано высказывание «Винни-Пух любит мёд, и дверь в дом открыта». Как бы вы сформулировали отрицание этого высказывания?
2. Почему таблица истинности для операции НЕ содержит две строки, а таблицы для других изученных операций — четыре? Сколько строк в таблице истинности выражения с тремя переменными? С четырьмя? С пятью?
3. В чём различие арифметического и логического сложения?
4. Сколько можно определить различных логических функций с двумя переменными? С тремя переменными?
5. Чем отличается операция исключающее ИЛИ от операции ИЛИ?
6. Почему операция исключающее ИЛИ называется сложением по модулю 2?
7. Как можно доказать или опровергнуть логическое равенство?
8. Что значит выражение «обратимая операция»? Какие известные вам логические операции обратимы?
9. Какое свойство операции исключающее ИЛИ позволяет использовать её для простейшего шифрования?
10. Чем отличается смысл высказывания «если А, то В» в обычной речи и в математической логике?
11. Запишите в виде логического выражения высказывание «если утюг горячий, то лоб холодный».
12. Запишите в виде логического выражения высказывание «неверно, что если утюг горячий, то лоб холодный». Если это высказывание истинно, можно ли сразу сказать, горячий ли утюг и горячий ли лоб?
13. Чем интересны операции штрих Шеффера и стрелка Пирса?

Подготовьте сообщение

а) «Логическая операция Штрих Шеффера»
б) «Логическая операция Стрелка Пирса»
в) «Шифрование с помощью операции исключающее ИЛИ»
г) «Взлом шифров с помощью частотного анализа»

Проекты

а) Программа для частотного анализа
б) Программа для шифрования текстов

Возможно выполнение задания в парах: один пишет программу для шифрования с помощью операции «исключающее ИЛИ», напарник — программу для взлома этого шифра.

§ 17
Логические выражения

Ключевые слова:

- логическое выражение
- таблица истинности
- порядок выполнения операций
- вычислимое выражение
- тавтология
- противоречие
- равносильные выражения

Вычисление логических выражений

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) эквиваленция.

Например, в логическом выражении $(A + (B \cdot C)) \to (A \cdot C)$ можно убрать все скобки, это не изменит порядок его вычисления и результат: $A + B \cdot C \to A \cdot C$.

Операции И, ИЛИ, исключающее ИЛИ и эквиваленция обладают свойством **ассоциативности**:

$$(A * B) * C = A * (B * C),$$

где символ «*» обозначает одну из таких операций. Для импликации это свойство неверно, обычно предполагают, что цепочка импликаций выполняется *справа налево*:

$$A \to B \to C = A \to (B \to C).$$

Порядок вычисления выражения можно, так же как и для арифметических выражений, определить с помощью дерева (рис. 3.10). Вычисление начинается с листьев, корень дерева — это самая последняя операция.

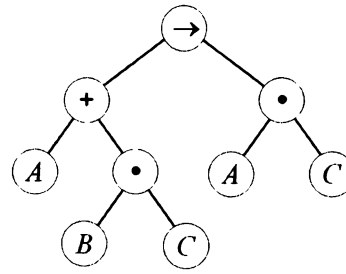


Рис. 3.10

При составлении таблицы истинности сложного логического выражения удобно разбить его на несколько простых, сначала вычислить значения этих промежуточных величин, а затем — окончательный результат. На рисунке 3.11 показано построение таблицы истинности для выражения $X = A + B \cdot C \to A \cdot C$.

A	B	C	$B \cdot C$	$A + B \cdot C$	$A \cdot C$	X
0	0	0	0	0	0	1
0	0	1	0	0	0	1
0	1	0	0	0	0	1
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	1	0	1	1	1
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Рис. 3.11

Из таблицы истинности видно, что при некоторых значениях переменных значение X истинно, а при других — ложно. Такие выражения называют **вычислимыми**.

Высказывание «Вася — программист или Вася не программист» всегда истинно (для любого Васи). Оно может быть записано в виде логического выражения $A + \overline{A}$. 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 $A \cdot \overline{A}$ 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, выражения $A \to B$ и $\overline{A} + B$ равносильны, а равенство $A \to B = \overline{A} + B$ называют **тождеством**. Равносильные выражения определяют одну и ту же логическую функцию, т. е. при одинаковых исходных данных приводят к одинаковым результатам.

Попробуем подсчитать, сколько существует логических функций от n переменных. Как мы знаем, таблица истинности однозначно определяет логическую функцию — правило преобразования исходных логических значений в результат. В таблице истинности любой функции от n переменных нужно перечислить все возможные комбинации исходных данных. Каждая переменная может принимать два значения: 0 или 1, поэтому общее число комбинаций равно 2^n. Это значит, что таблица истинности содержит 2^n строк. Поэтому столбец значений функции содержит $L = 2^n$ бит, а общее число различных комбинаций этих битов (и следовательно, количество различных функций) равно $2^L = 2^{2^n}$.

Диаграммы Эйлера–Венна

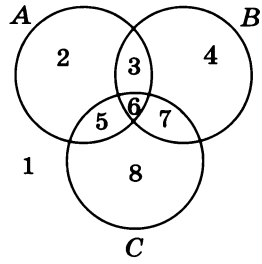
Логические выражения, зависящие от небольшого количества переменных (обычно не более четырёх), удобно изображать в виде диаграмм, которые называют **диаграммами Эйлера–Венна**. На такой диаграмме каждой переменной соответствует круг, внутри которого её значение истинно, а вне его — ложно. Круги могут пересекаться.

Области, соответствующие базовым логическим операциям, показаны на рис. 3.12. Серым цветом залиты области, где рассматриваемое выражение равно единице (истинно).

$A \cdot B$ $A + B$

Рис. 3.12

Диаграмма Эйлера–Венна для трёх переменных содержит 8 областей. Для каждой из них запишем логические выражения (рис. 3.13).



1: $\overline{A} \cdot \overline{B} \cdot \overline{C}$ 5: $A \cdot \overline{B} \cdot C$
2: $A \cdot \overline{B} \cdot \overline{C}$ 6: $A \cdot B \cdot C$
3: $A \cdot B \cdot \overline{C}$ 7: $\overline{A} \cdot B \cdot C$
4: $\overline{A} \cdot B \cdot \overline{C}$ 8: $\overline{A} \cdot \overline{B} \cdot C$

Рис. 3.13

Для того чтобы найти логическое 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: $A \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C}$.

Вместе с тем если не обращать внимания на область A, то 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 Эйлера–Венна удобно применять для решения задач, в которых используются множества, например множества страниц, полученных от поисковой системы в ответ на какой-то запрос.

Некоторые задачи

Рассмотрим ряд задач, в которых требуется исследовать таблицы истинности и логические выражения.

Задача 1. Дано логическое выражение от пяти переменных:

$$X_1 \cdot \overline{X}_2 \cdot \overline{X}_3 \cdot X_4 \cdot \overline{X}_5.$$

Сколько существует различных наборов значений переменных, при которых это логическое выражение истинно?

Решение. Для того чтобы логическое произведение было истинно, необходимо, чтобы все его «сомножители» были истинны. Поэтому существует только один подходящий набор переменных:

$$X_1 = 1,\ X_2 = 0,\ X_3 = 0,\ X_4 = 1,\ X_5 = 0,$$

при котором заданное выражение истинно. Полная таблица истинности для логического выражения с пятью переменными содержит $2^5 = 32$ строки. Для остальных наборов значений переменных (их 31!) значение этого выражения ложно.

Задача 2. Дано логическое выражение от пяти переменных:

$$X_1 + \overline{X}_2 + \overline{X}_3 + X_4 + \overline{X}_5.$$

Сколько существует различных наборов значений переменных, при которых это логическое выражение ложно?

Решение. Для того чтобы логическая сумма была равна нулю, необходимо, чтобы все её «слагаемые» были равны нулю. Поэтому существует только один подходящий набор переменных:

$$X_1 = 0,\ X_2 = 1,\ X_3 = 1,\ X_4 = 0,\ X_5 = 1,$$

при котором заданное выражение ложно. Полная таблица истинности для логического выражения с пятью переменными содержит $2^5 = 32$ строки. Для любого из остальных 31 наборов значений переменных значение этого выражения истинно.

Задача 3. Дан фрагмент таблицы истинности выражения F с тремя переменными: X, Y, Z.

X	*Y*	*Z*	*F*
1	0	0	1
0	0	0	0
1	1	1	0

Запишите любое логическое выражение, которое соответствует этой таблице истинности.

Решение. Вспомним, что полная таблица истинности для логического выражения с тремя переменными содержит $2^3 = 8$ строк, из них в нашей задаче известны три, а остальные пять неизвестны. В этих неизвестных строках функция F может принимать любые логические значения (0/1), т. е. существует $2^5 = 32$ различных логических функций, удовлетворяющих заданной неполной таблице истинности.

Заметим, что в последнем столбце таблицы одна единица и два нуля. Такой таблице может соответствовать, например, функция, которая равна 1 для комбинации аргументов в первой строке таблицы ($X = 1$, $Y = Z = 0$) и равна 0 во всех остальных строках.

Эта функция имеет вид $F = X \cdot \overline{Y} \cdot \overline{Z}$ (см. задачу 1): поскольку в первой строке $X = 1$, инверсию для X применять не нужно, а для тех аргументов, которые равны нулю ($Y = Z = 0$), — нужно.

Ответ: $F = X \cdot \overline{Y} \cdot \overline{Z}$.

Задача 4. Дан фрагмент таблицы истинности выражения F:

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	F
		0					1	1
0			0					0
	0						0	0

Пустые ячейки могут содержать как 0, так и 1. Запишите любое логическое выражение, которое соответствует этой таблице истинности.

Решение. В последнем столбце таблицы истинности мы видим одну единицу и два нуля. Такой таблице может соответствовать, например, функция, которая равна 1 для комбинации аргументов в первой строке таблицы и равна 0 во всех остальных строках. Эта функция представляет собой логическое произведение аргументов и их инверсий (см. задачу 1).

В первой строке мы знаем только два значения: $X_3 = 0$ и $X_8 = 1$. Поэтому переменная X_3 должна войти в произведение с инверсией, а переменная X_8 — без инверсии. Остальные переменные можно взять как с инверсией, так и без неё, например:

$$F = X_1 \cdot \overline{X}_2 \cdot \overline{X}_3 \cdot X_4 \cdot X_5 \cdot \overline{X}_6 \cdot \overline{X}_7 \cdot X_8,$$

предполагая, что в первой строке таблицы $X_1 = X_4 = X_5 = 1$ и $X_2 = X_6 = X_7 = 0$.

Функций, удовлетворяющих условию задачи, очень много. Полная таблица истинности для логического выражения с восемью переменными содержит $2^8 = 256$ строк, из них в нашей задаче частично известны только три.

Ограничимся только функциями, представляющими собой логическое произведение всех переменных или их инверсий. В первой строке 6 неизвестных ячеек, в каждой из которых может быть 0 или 1. Каждая из этих комбинаций даёт свою логическую функцию, таким образом получаем $2^6 = 64$ функции выбранной структуры. А если мы будем учитывать вообще все подходящие функции, их количество будет намного больше (если вам интересно, попробуйте подсчитать его самостоятельно).

Ответ: $F = X_1 \cdot \overline{X}_2 \cdot \overline{X}_3 \cdot X_4 \cdot X_5 \cdot \overline{X}_6 \cdot \overline{X}_7 \cdot X_8$.

Задача 5. Дан фрагмент таблицы истинности выражения F:

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	F
	0						1	0
1			0					1
			1				1	1

Пустые ячейки могут содержать как 0, так и 1. Запишите любое логическое выражение, которое соответствует этой таблице истинности.

Решение. В последнем столбце таблицы истинности видим один ноль и две единицы. Поэтому можно выбрать такую функцию, которая равна 0 в одной-единственной строке и равна 1 во всех остальных строках, в том числе и в тех, про которые ничего не известно. Эта функция представляет собой логическую сумму всех переменных или их инверсий (см. задачу 2).

Рассмотрим «особую» первую строку в таблице истинности, где функция равна 0. Мы знаем в ней только два значения — $X_2 = 0$ и $X_8 = 1$. Для того чтобы в результате в первой строке получить 0, необходимо, чтобы переменная X_8 входила в сумму с инверсией (тогда из 1 получится 0!), а переменная X_2 — без инверсии. Пример такой функции:

$$F = X_1 + X_2 + X_3 + \overline{X}_4 + \overline{X}_5 + \overline{X}_6 + \overline{X}_7 + \overline{X}_8,$$

при $X_1 = X_3 = 0$ и $X_4 = X_5 = X_6 = X_7 = 1$ в первой строке таблицы.

Ответ: $F = X_1 + X_2 + X_3 + \overline{X}_4 + \overline{X}_5 + \overline{X}_6 + \overline{X}_7 + \overline{X}_8$.

Задача 6. Даны две логические функции F и G от пяти одинаковых переменных. В их таблицах истинности 5 одинаковых строк, причём в трёх из них обе функции равны 1, а в двух остальных равны 0. При скольких комбинациях переменных:

а) функция $F \cdot G$ равна 0;
б) функция $F \cdot G$ равна 1;
в) функция $F + G$ равна 0;
г) функция $F + G$ равна 1?

Решение. В таблице истинности любой функции от пяти переменных всего $2^5 = 32$ строки. В трёх из них обе функции равны 1, поэтому для этих строк $F \cdot G = 1$ и $F + G = 1$.

Как сказано в условии, в таблицах истинности есть ещё две одинаковые строки, и в них обе функции равны 0. Поэтому для этих двух строк $F \cdot G = 0$ и $F + G = 0$.

Остальные $32 - 5 = 27$ строк в таблицах истинности разные, это значит, что для каждого из этих наборов данных одна функция равна 0, а вторая — 1. Для этих 27 строк получаем $F \cdot G = 0$ и $F + G = 1$.

Ответ: а) 29, б) 3, в) 2, г) 30.

Задача 7. Логические выражения A и B зависят от одного и того же набора из семи переменных. В таблицах истинности каждого из этих выражений в столбце значений стоит ровно по 4 единицы. Каково максимально возможное число единиц может быть в столбце значений таблицы истинности выражения $A + B$?

Решение. В таблице истинности любой функции от семи переменных всего $2^7 = 128$ строк. Выражение $A + B$ равно 1 тогда, когда хотя бы одно из слагаемых равно 1. Максимально возможное число единиц получается тогда, когда выражения A и B равны 1 для разных наборов переменных. В этом случае в столбце значений таблицы истинности выражения $A + B$ будет $4 + 4 = 8$ единиц.

Ответ: 8 единиц.

Задача 8. Логическая функция F задаётся выражением $F = \overline{Z} \cdot X + X \cdot Y$. Определите, какому столбцу таблицы истинности функции F соответствует каждая из переменных X, Y, Z.

?	?	?	*F*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Решение. Составим таблицу истинности для логической функции $F = \overline{Z} \cdot X + X \cdot Y$ и выделим в ней те строки, где $F = 1$:

Теперь сравним строки, выделенные фоном, и строки таблицы из условия, в которых функция равна 1:

?	?	?	*F*
0	0	1	1
0	1	1	1
1	1	1	1

Сравнивая столбцы интересующих нас строк, определяем, что в третьем столбце заданной таблицы — значение X (три единицы), во втором — Y (один ноль и две единицы) и в первом — Z (два ноля и единица).

Задача 8. В таблице приведены запросы[1)] и количество страниц, которые нашел поисковый сервер по этим запросам в некотором сегменте Интернета:

№	Запрос	Найдено страниц, тыс.
1	*муравей*	100
2	*куропатка*	130
3	*ноосфера*	90
4	*муравей* | *куропатка*	180
5	*муравей* | *ноосфера*	160
6	*ноосфера* & (*муравей* | *куропатка*)	30

Сколько страниц (в тысячах) будет найдено по запросу

куропатка & (*муравей* | *ноосфера*)

[1)] В поисковых запросах операция И обозначается знаком &, а операция ИЛИ — знаком |.

Количество страниц, соответствующих запросу $A|B$, которое мы обозначим как $N_{A|B}$, определяется формулой

$$N_{A|B} = N_A + N_B - N_{A\&B},$$

где N_A, N_B и $N_{A\&B}$ — количество страниц, полученных по запросам A, B и A & B, соответственно.

Решение. Обозначим области *муравей*, *куропатка* и *ноосфера* буквами M, K и H. Нас интересует область K & $(M|H)$, выделенная серым фоном на рис. 3.14, *а*.

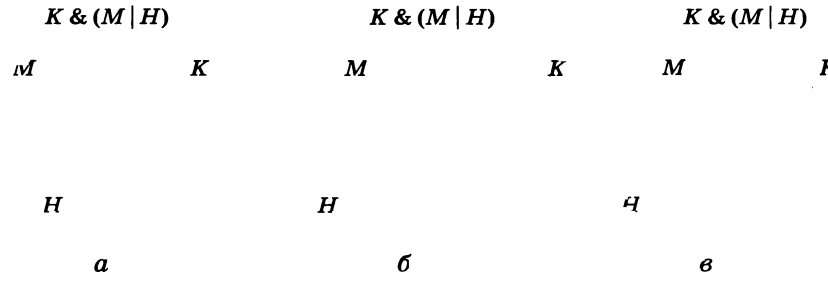


Рис. 3.14

Найдём размер областей K & M и H & M, используя общую формулу для двух областей

$$N_{A|B} = N_A + N_B - N_{A\&B}.$$

Получаем, подставляя результаты запросов 1–5:

$$N_{K\&M} = N_K + N_M - N_{K|M} = 130 + 100 - 180 = 50,$$
$$N_{H\&M} = N_H + N_M - N_{H|M} = 90 + 100 - 160 = 30.$$

Теперь используем запрос 6: размер области H & $(M|K)$ равен 30. Но, как мы только что получили, размер области H & M тоже равен 30. Это может быть только тогда, когда область H & K пустая (и диаграмму можно перерисовать так, как показано на рис. 3.14, *б*) или входит целиком в область H & M (рис. 3.14, *в*). В обоих случаях область K & $(M|H)$ совпадает с областью K & M, размер которой нам известен, он равен 50.

Заметим, что задача с трёмя областями в общем виде (когда каждая область пересекается с каждой) достаточно сложна. Она сильно упрощается, если какие-то две области не пересекаются.

Ответ: 50.

Выводы

- Порядок действий при вычислении логических выражений:
  1) действия в скобках;
  2) отрицание (НЕ);
  3) логическое умножение (И);
  4) логическое сложение (ИЛИ) и исключающее ИЛИ;
  5) импликация;
  6) эквиваленция.
- Таблица истинности логического выражения от n переменных содержит 2^n строк.
- Количество различных логических функций от n переменных равно 2^{2^n}.
- Логическое произведение переменных или их инверсий равно 1 при единственной комбинации значений этих переменных.
- Логическая сумма переменных или их инверсий равна 0 при единственной комбинации значений этих переменных.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Сравните понятия «логическая функция» и «логическое выражение».
2. Можно ли сказать, что таблица истинности полностью определяет логическое выражение? Ответ обоснуйте.
3. Сравните представления логического выражения в виде формулы и в виде дерева. Укажите достоинства и недостатки каждой из форм.
4. При каких значениях переменных логическое выражение $X_1 \cdot \overline{X}_2 \cdot \overline{X}_3 \cdot X_4$ истинно? При каких оно ложно?
5. При каких значениях переменных логическое выражение $X_1 + X_2 + \overline{X}_3 + \overline{X}_4$ ложно? При каких оно истинно?
6. Задано 5 строк таблицы истинности некоторого логического выражения с тремя переменными. Сколько различных логических функций соответствуют этой неполной таблице истинности?

*7. Приведите пример логического выражения с пятью переменными, у которого в таблице истинности:
   а) две единицы, а остальные — нули;
   б) два нуля, а остальные — единицы.

Подготовьте сообщение

а) «Диаграммы Эйлера–Венна и теория множеств»
б) «Язык запросов поисковых систем»

Проект

«Сравнение поисковых систем»

§ 18
Упрощение логических выражений

Ключевые слова:

- эквивалентные преобразования
- законы алгебры логики
- законы де Моргана

Законы алгебры логики

Для упрощения логических выражений применяют **эквивалентные преобразования.** Это означает, что исходное и упрощённое выражения определяют одну и ту же логическую функцию (имеют одинаковые таблицы истинности).

Законы алгебры логики, с помощью которых выполняют эквивалентные преобразования, формулируются для базовых логических операций — НЕ, И и ИЛИ (рис. 3.15).

Законы	Для И	Для ИЛИ
Двойного отрицания	$\overline{\overline{A}} = A$	
Исключённого 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	$\overline{A \cdot B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \cdot \overline{B}$

Рис. 3.15

Закон двойного отрицания означает, что операция НЕ обратима: если применить её два раза, логическое значение не изменится. Закон исключённого третьего основан на том, что в классической (двузначной) логике 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 так же, как и в школь-

ной математике. 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 алгебре 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

$$A + \overline{A} \cdot B = (A + \overline{A}) \cdot (A + B) = A + B.$$

Правила, позволяющие раскрывать отрицание сложных выражений, названы в честь шотландского математика и логика Августа 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.

А. де Морган (1806–1871)

Теперь с помощью приведённых законов алгебры логики упростим полученное ранее логическое выражение для объединения областей 3 и 4 на диаграмме с тремя переменными (см. рис. 3.13):

$$A \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot \overline{C} = (A + \overline{A}) \cdot B \cdot \overline{C} = B \cdot \overline{C}.$$

Здесь мы сначала вынесли общий множитель двух слагаемых за скобки, а затем применили закон исключённого третьего.

В общем случае можно рекомендовать такую последовательность действий:

1. Заменить все «небазовые» операции (исключающее ИЛИ, импликацию, эквиваленцию 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

$$(A + \overline{B}) \cdot \overline{A + B} \cdot (\overline{A} + C) = (A + \overline{B}) \cdot \overline{A} \cdot \overline{B} \cdot (\overline{A} + C) =$$
$$= (A \cdot \overline{A} + \overline{B} \cdot \overline{A}) \cdot \overline{B} \cdot (\overline{A} + C) = \overline{B} \cdot \overline{A} \cdot \overline{B} \cdot (\overline{A} + C) =$$
$$= \overline{A} \cdot \overline{B} \cdot \overline{B} \cdot (\overline{A} + C) = \overline{A} \cdot \overline{B} \cdot (\overline{A} + C) = \overline{B} \cdot \overline{A} \cdot (\overline{A} + C) = \overline{B} \cdot \overline{A}.$$

Здесь последовательно использованы закон де Моргана, распределительный закон, закон исключённого третьего, переместительный закон, закон повторения, снова переместительный закон и закон поглощения.

Выводы

- Эквивалентные преобразования логических выражений — это такие преобразования, при которых исходное и преобразованное выражения определяют одну и ту же логическую функцию.
- Эквивалентные преобразования выполняются с помощью законов алгебры логики.
- Перед применением законов логики необходимо представить все логические операции через базовые операции НЕ, И и ИЛИ.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Запишите несколько различных логических выражений, которые тождественно равны

   а) 0; б) 1; в) $A + B$; г) $A \cdot B$; д) $A \to B$.
2. Докажите законы де Моргана с помощью таблиц истинности.
3. Сравните законы алгебры логики и правила преобразования выражений в алгебре. Найдите сходства и различия.
4. Объясните различие между логическими выражениями $\overline{A \cdot B}$ и $\overline{A} \cdot \overline{B}$.
5. Запишите законы де Моргана для выражений

   а) $\overline{A \cdot B \cdot C \cdot D}$; б) $\overline{A + B + C + D}$; в) $\overline{A \cdot B + C \cdot D}$.

Проекты

а) Преобразование логических выражений к базису И-НЕ
б) Преобразование логических выражений к базису ИЛИ-НЕ

§ 19
Логические уравнения

Ключевые слова:

- логическое уравнение
- количество решений
- битовая цепочка
- замена переменных

Если приравнять два логических выражения, мы получим уравнение. Его решением будут значения переменных, при которых уравнение превращается в истинное равенство, т. е. когда значения левой и правой частей совпадают. Например, уравнение $A \cdot B = 1$ имеет единственное решение: $A = B = 1$, для остальных комбинаций значений переменных левая часть равна нулю. В то же время уравнение $A + B = 1$ имеет три решения: ($A = 0$, $B = 1$), ($A = 1$, $B = 0$) и $A = B = 1$.

Логическое уравнение всегда имеет конечное количество решений. Для уравнения с n переменными количество решений не может быть больше, чем количество комбинаций n двоичных значений, равное 2^n.

Все решения уравнения

Пример 1. Требуется найти все решения уравнения

$$(\overline{(B + C)} \cdot A) \to (\overline{A} \cdot \overline{C} + D) = 0.$$

Способ 1. Вспоминаем, что импликация равна нулю только тогда, когда первое выражение равно 1, а второе — 0. Поэтому исходное уравнение сразу разбивается на два:

$$(\overline{(B + C)} \cdot A) = 1, \qquad \overline{A} \cdot \overline{C} + D = 0.$$

Первое уравнение с помощью закона де Моргана можно преобразовать к виду $\overline{B} \cdot \overline{C} \cdot A = 1$, откуда сразу следует, что все три сомножителя должны быть равны 1. Это значит, что $A = 1$, $B = 0$ и $C = 0$. Кроме того, из второго уравнения следует, что $D = 0$. Решение найдено, причем оно единственное.

Способ 2. Заменяя импликацию по формуле $A \to B = \overline{A} + B$, получаем

$$\overline{(\overline{(B + C)} \cdot A)} + \overline{A} \cdot \overline{C} + D = 0.$$

Используем закон де Моргана:

$$B + C + \overline{A} + \overline{A} \cdot \overline{C} + D = 0$$

и закон поглощения

$$B + C + \overline{A} + D = 0.$$

Для того, чтобы логическая сумма была равна нулю, каждое слагаемое должно быть равно нулю, поэтому $A = 1$, $B = C = D = 0$.

Способ 3. Можно 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 трудоёмок.

Пример 2. Требуется найти все решения уравнения

$$(A + \overline{B}) \to (B \cdot C \cdot D) = 1.$$

Преобразуем выражение, раскрыв импликацию через НЕ и ИЛИ и применив закон де Моргана:

$$\overline{A + \overline{B}} + B \cdot C \cdot D = \overline{A} \cdot B + B \cdot C \cdot D.$$

Вынося за скобку B, получаем

$$B \cdot (\overline{A} + C \cdot D) = 1,$$

откуда сразу следует, что $B = 1$. Таким образом, количество решений исходного уравнения равно количеству решений уравнения $\overline{A} + C \cdot D = 1$.

Если логическая сумма равна 1, то хотя бы одно слагаемое равно 1 (или оба одновременно). Во-первых, уравнение $\overline{A} + C \cdot D = 1$ обращается в тождество при $\overline{A} = 1$, т. е. при $A = 0$. При этом получаем четыре решения, для всех возможных комбинаций C и D. Во-вторых, это равенство выполняется при $C \cdot D = 1$, т. е. при $C = D = 1$ и любом A. Это даёт ещё два решения. Но из этих шести решения два совпадают: $A = 0$, $C = 1$, $D = 1$. Поэтому уравнение имеет всего пять разных решений.

Количество решений уравнения

Пример 3. Требуется найти число решений уравнения

$$A \cdot B \cdot C + \overline{B} \cdot \overline{C} \cdot D = 0.$$

Здесь, в отличие от предыдущих задач, не нужно находить сами решения, нас интересует только их количество. Уравнение распадается на два:

$$A \cdot B \cdot C = 0 \quad \text{и} \quad \overline{B} \cdot \overline{C} \cdot D = 0.$$

Каждое из них имеет достаточно много решений. Можно поступить следующим образом: сначала найти количество решений «обратного» уравнения, с единицей в правой части:

$$A \cdot B \cdot C + \overline{B} \cdot \overline{C} \cdot D = 1,$$

и затем вычесть его из 16 (общего количества различных комбинаций четырёх переменных). Уравнение $A \cdot B \cdot C = 1$ имеет два решения: $A = B = C = 1$ и любое D (0 или 1). Второе уравнение, $\overline{B} \cdot \overline{C} \cdot D = 1$, тоже имеет два решения: A — любое, $B = C = 0$, $D = 1$. Среди этих четырёх решений нет повторяющихся, поэтому исходное уравнение имеет $16 - 4 = 12$ решений.

Пример 4. Требуется найти число решений уравнения

$$(X_1 \sim X_2) \cdot (X_2 \sim X_3) \cdot (X_3 \sim X_4) \cdot (X_4 \sim X_5) \cdot (X_5 \sim X_6) = 1.$$

Так как каждая логическая переменная может принимать только значения 0 и 1, можно представить решение в виде 6-битной цепочки. Например, цепочка 010110 означает, что $X_1 = X_3 = X_6 = 0$ и $X_2 = X_4 = X_5 = 1$. Количество различных битовых цепочек длины 6 равно $2^6 = 64$, поэтому число решений уравнения — не более 64.

Для того чтобы логическое произведение было равно 1, необходимо, чтобы каждый сомножитель был равен 1. Это значит, что значения X_i и X_{i+1} равны для всех i, т. е. соседние биты в цепочке должны быть равны. Это условие определяет всего две разрешённые цепочки, одна из которых состоит из одних нулей, вторая — из одних единиц. Поэтому уравнение имеет два решения: 000000 и 111111.

Пример 5. Требуется найти число решений уравнения[1)]

$$(X_1 \neq X_2) \cdot (X_2 \neq X_3) \cdot (X_3 \neq X_4) \cdot (X_4 \neq X_5) \cdot (X_5 \neq X_6) = 1.$$

Так же как и в предыдущем примере, необходимо, чтобы каждый сомножитель был равен 1. Это значит, что значения X_i и X_{i+1} не равны для всех i, т. е. соседние биты в цепочке должны быть разными. Это условие определяет всего две разрешённые цепочки, в которых биты чередуются: одна из них начинается с нуля, а вторая — с единицы. Поэтому уравнение имеет два решения: 010101 и 101010.

Пример 6. Требуется найти число решений уравнения

$$(X_1 \to X_2) \cdot (X_2 \to X_3) \cdot (X_3 \to X_4) \cdot (X_4 \to X_5) \cdot (X_5 \to X_6) = 1.$$

Импликация $A \to B$ ложна только тогда, когда $A = 1$ и $B = 0$. Поэтому в битовой цепочке, которая является решением уравнения, не может встречаться последовательность 10, иначе какая-то

[1)] Здесь и далее знаком $\neq$ будем обозначать логическое неравенство («неэквивалентность»). Результат этой операции совпадает с результатом операции исключающее ИЛИ.

импликация окажется ложной, и всё выражение в левой части будет равно 0. Поэтому существует всего 7 решений, все они имеют структуру «сначала нули, потом единицы»:

000000 000001 000011 000111 001111 011111 111111

Обратите внимание, что количество решений логических уравнений, в отличие от количества решений «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.

Системы логических уравнений

Пример 7. Требуется найти число решений системы уравнений

$$\begin{cases} (X_1 \to X_2) \cdot (X_2 \to X_3) \cdot (X_3 \to X_4) \cdot (X_4 \to X_5) = 1; \\ (Y_1 \to Y_2) \cdot (Y_2 \to Y_3) \cdot (Y_3 \to Y_4) = 1. \end{cases}$$

В эту систему входит 9 логических переменных, поэтому число решений не может быть более $2^9 = 512$.

Как мы знаем из примера 6, первое уравнение имеет 6 различных решений (назовём их X-решениями):

00000 00001 00011 00111 01111 11111,

а второе — 5 различных решений (Y-решений):

0000 0001 0011 0111 1111.

Два уравнения независимы, так как во второе уравнение не входят переменные из первого, и наоборот. Поэтому каждому из шести X-решений может соответствовать любое из пяти Y-решений, так что общее количество решений определяется как произведение $6 \cdot 5 = 30$.

Пример 8. Требуется найти число решений системы уравнений

$$\begin{cases} (X_1 \to X_2) \cdot (X_2 \to X_3) \cdot (X_3 \to X_4) \cdot (X_4 \to X_5) = 1; \\ (Y_1 \to Y_2) \cdot (Y_2 \to Y_3) \cdot (Y_3 \to Y_4) = 1; \\ X_2 \to Y_3 = 1. \end{cases}$$

Здесь к двум независимым уравнениям из предыдущего примера добавлено третье уравнение, которое можно рассматривать как уравнение связи. Оно ограничивает количество решений,

поскольку теперь нужно исключить все пары цепочек X–Y, для которых $X_2 \to Y_3 = 0$ или, что то же самое, $X_2 = 1$ и $Y_3 = 0$.

Если $X_2 = 0$ (это справедливо для первых четырёх X-решений), никаких дополнительных ограничений нет ($X_2 \to Y_3 = 0 \to Y_3 = 1$ при любом Y_3). Таким образом, получаем $4 \cdot 5 = 20$ решений.

Теперь рассмотрим два X-решения, для которых $X_2 = 1$: это 01111 и 11111. Каждое из них может комбинироваться только с теми Y-решениями, где $Y_3 = 1$, т. е. с 0011, 0111 и 1111. Это даёт еще $2 \cdot 3 = 6$ решений. Поэтому исходная система имеет $20 + 6 = 26$ решений.

Пример 9. Требуется найти число решений системы уравнений

$$\begin{cases} (X_1 \neq Y_1) \sim (X_2 \sim Y_2); \\ (X_2 \neq Y_2) \sim (X_3 \sim Y_3); \\ \ldots \\ (X_8 \neq Y_8) \sim (X_9 \sim Y_9). \end{cases}$$

В этой системе 9 однотипных уравнений. Заметим, что переменная X_i встречается только в паре с Y_i, поэтому можно выполнить замену переменных: $Z_i = (X_i \sim Y_i)$. Тогда исходная система преобразуется к виду

$$\overline{Z} \sim Z_2,\ \overline{Z}_2 \sim Z_3,\ \ldots,\ \overline{Z}_8 \sim Z_9$$

и даже может быть «свёрнута» в одно уравнение

$$(Z_1 \neq Z_2) \cdot (Z_2 \neq Z_3) \cdot \ldots (Z_8 \neq Z_9) = 1.$$

Такое уравнение мы рассмотрели в примере 5, оно имеет два решения: Z-цепочки с чередованием битов: 010101010 и 101010101.

Теперь нужно перейти обратно к исходным переменным X_i и Y_i. Предположим, что $Z_i = (X_i \sim Y_i) = 0$ для некоторого i. В этом случае возможны две комбинации (X_i, Y_i): (0, 1) и (1, 0). Это означает, что каждый ноль в Z-цепочке удваивает количество решений при переходе к исходным переменным.

Для случая $Z_i = 1$ также возможны две комбинации (X_i, Y_i): (0, 0) и (1, 1). Поэтому каждая единица в Z-цепочке тоже удваивает количество решений. Так как каждая из двух Z-цепочек состоит из 9 бит и каждый из них удваивает количество решений, общее число решений исходной системы равно $2^9 + 2^9 = 1024$.

Пример 10. Требуется найти число решений системы уравнений

$$\begin{cases} (X_1 + Y_1) \cdot (X_2 \cdot Y_2 \to X_1 \cdot Y_1) = 1; \\ (X_2 + Y_2) \cdot (X_3 \cdot Y_3 \to X_2 \cdot Y_2) = 1; \\ \ldots \\ (X_6 + Y_6) \cdot (X_7 \cdot Y_7 \to X_6 \cdot Y_6) = 1; \\ X_7 + Y_7 = 1. \end{cases}$$

Из системы уравнений следует, что должны выполняться два условия:

1) $X_i + Y_i = 1$ для всех $i = 1, \ldots, 7$;
2) $X_{i+1} \cdot Y_{i+1} \to X_i \cdot Y_i = 1$ для всех $i = 1, \ldots, 6$.

Поэтому можно записать систему из двух уравнений, которая будет эквивалентна исходной:

$$\begin{cases} (X_2 \cdot Y_2 \to X_1 \cdot Y_1) \cdot (X_3 \cdot Y_3 \to X_2 \cdot Y_2) \cdot \ldots \cdot (X_7 \cdot Y_7 \to X_6 \cdot Y_6) = 1; \\ (X_1 + Y_1) \cdot (X_2 + Y_2) \cdot \ldots \cdot (X_7 + Y_7) = 1. \end{cases}$$

Решим сначала первое из этих уравнений, а потом «подключим» второе в качестве дополнительного ограничения. Выполнив замену переменных $Z_i = X_i \cdot Y_i$, перепишем первое уравнение в виде

$$(Z_2 \to Z_1) \cdot (Z_3 \to Z_2) \cdot \ldots \cdot (Z_7 \to Z_6) = 1.$$

Это уравнение имеет 8 решений (Z-цепочек):

0000000, 1000000, 1100000, 1110000, 1111000, 1111100, 1111110, 1111111.

Теперь выполним обратный переход к исходным переменным X_i и Y_i, добавляя дополнительное ограничение $X_i + Y_i = 1$, которое следует из второго уравнения.

Если $Z_i = X_i \cdot Y_i = 0$, то при условии $X_i + Y_i = 1$ мы получаем две возможные комбинации (X_i, Y_i): (0, 1) и (1, 0). Это означает, что каждый ноль в Z-цепочке удваивает количество решений при переходе к исходным переменным. Если $Z_i = X_i \cdot Y_i = 1$, то при условии $X_i + Y_i = 1$ мы получаем единственную пару $(X_i, Y_i) = (1, 1)$, т. е. единица в Z-цепочке не изменяет количество решений. Поэтому количество решений, которые даёт каждая Z-цепочка, определяется количеством нулей в ней: Z-цепочка с n нулями даёт 2^n решений в исходных переменных.

Таким образом, первая Z-цепочка — 0000000, содержащая семь нулей, даст $2^7 = 128$ решений, вторая цепочка (с шестью нулями) — $2^6 = 64$ решения и т. д. Общее количество решений равно

$$128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255.$$

Выводы

- Логическое уравнение всегда имеет конечное количество решений. Для уравнения с n переменными оно не может быть больше, чем 2^n.
- Любое логическое уравнение (и любую систему логических уравнений) можно решить с помощью таблицы истинности, но этот способ при большом количестве переменных очень трудоёмок.
- Решение системы логических уравнений часто удобно представлять в виде битовой цепочки. Сами уравнения задают ограничения на свойства этой цепочки.
- При решении систем логических уравнений можно использовать замену переменных. В этом случае, получив решение в новых переменных, необходимо выяснить, как оно изменится при переходе к исходным переменным.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Сравните алгебраические и логические уравнения. В чём их сходство и различие?
2. Почему логическое уравнение не может иметь бесконечно много решений?

*3. В каких случаях можно выполнять замену переменных в логических уравнениях? Когда такой приём может привести к неверному результату?

4. При решении системы логических уравнений была сделана замена переменных $Z_i = X_i \cdot Y_i$ ($i = 1, ..., 6$) и получены три решения — Z-цепочки 101010, 111101 и 100000. Определите количество решений системы в исходных переменных (X_i, Y_i).
5. При решении системы логических уравнений была сделана замена переменных $Z_i = X_i \to Y_i$ ($i = 1, ..., 5$) и получены три решения — Z-цепочки 10110, 11000 и 00001. Определите количество решений системы в исходных переменных (X_i, Y_i).

*6. При решении системы логических уравнений была сделана замена переменных $Q_i = X_i \cdot Y_i \cdot Z_i$ ($i = 1, ..., 4$) и получены три решения — Z-цепочки 1010, 1111 и 1000. Определите количество решений системы в исходных переменных (X_i, Y_i, Z_i).

Подготовьте сообщение

а) «Решение логических уравнений с помощью таблиц истинности»
б) «Методы решения систем логических уравнений»

Проект

Сравнение методов решения логических уравнений

§ 20
Синтез логических выражений

Ключевые слова:

- синтез
- таблица истинности
- дизъюнктивная нормальная форма
- конъюнктивная нормальная форма

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 греч. αναλυσις — разложение), в них требуется исследовать заданное 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 греч. συνθεσις — совмещение).

В качестве простейшего примера построим логическое выражение, тождественное операции импликации $X = A \to B$, по её таблице истинности (рис. 3.16).

A	*B*	*X*	
0	0	1	$\overline{A} \cdot \overline{B}$
0	1	1	$\overline{A} \cdot B$
1	0	0	
1	1	1	$A \cdot B$

Рис. 3.16

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 $B = 0$, т. е. только в первой строке таблицы. Выражение $\overline{A} \cdot B$ истинно только во второй строке, а $A \cdot B$ — только в последней. Существует 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: $X = \overline{A} \cdot \overline{B} + \overline{A} \cdot B + A \cdot B$.

Такая запись — дизъюнкция простых конъюнкций — называется **дизъюнктивной нормальной формой** (ДНФ).

Упрощаем это выражение:

$$X = \overline{A} \cdot (\overline{B} + B) + A \cdot B = \overline{A} + A \cdot B = (\overline{A} + A) \cdot (\overline{A} + B) = \overline{A} + B.$$

Таким образом, мы вывели формулу, которая позволяет представить импликацию через операции НЕ и ИЛИ.

Способ 2. Если в таблице истинности нулей меньше, чем единиц, удобнее сначала найти формулу для обратного выражения, $\overline{X}$, а потом применить операцию НЕ. В данном случае выражение равно нулю в единственной строке, при $A = 1$ и $B = 0$, и только в этой строке $\overline{X} = 1$, поэтому, используя предыдущий способ, получаем: $\overline{X} = A \cdot \overline{B}$. Теперь остается применить операцию НЕ и закон де Моргана:

$$X = \overline{A \cdot \overline{B}} = \overline{A} + B.$$

Рассмотрим более сложный пример, когда выражение зависит от трёх переменных. В этом случае в таблице истинности будет 8 строк. Отметим все строки, где $X = 1$, и для каждой из них построим выражение, истинное только для этой комбинации переменных (рис. 3.17).

A	*B*	*C*	*X*	
0	0	0	1	$\overline{A} \cdot \overline{B} \cdot \overline{C}$
0	0	1	1	$\overline{A} \cdot \overline{B} \cdot C$
0	1	0	1	$\overline{A} \cdot B \cdot \overline{C}$
0	1	1	1	$\overline{A} \cdot B \cdot C$
1	0	0	0	
1	0	1	1	$A \cdot \overline{B} \cdot C$
1	1	0	0	
1	1	1	1	$A \cdot B \cdot C$

Рис. 3.17

Теперь выполним логическое сложение — построим ДНФ:

$$X = \overline{A} \cdot \overline{B} \cdot \overline{C} + \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + \overline{A} \cdot B \cdot C + A \cdot \overline{B} \cdot C + A \cdot B \cdot C.$$

Упрощение этого выражения даёт

$$X = \overline{A}\cdot\overline{B}\cdot(\overline{C}+C)+\overline{A}\cdot B\cdot(\overline{C}+C)+A\cdot C\cdot(\overline{B}+B)=$$
$$=\overline{A}\cdot\overline{B}+\overline{A}\cdot B+A\cdot C=\overline{A}\cdot(\overline{B}+B)+A\cdot C=\overline{A}+A\cdot C=$$
$$=(\overline{A}+A)\cdot(\overline{A}+C)=\overline{A}+C.$$

Используя способ 2, получаем

$$\overline{X} = A\cdot\overline{B}\cdot\overline{C}+A\cdot B\cdot\overline{C}=A\cdot\overline{C}\cdot(\overline{B}+B)=A\cdot\overline{C}.$$

Тогда $X=\overline{A\cdot\overline{C}}=\overline{A}+C$. В данном случае второй способ оказался проще, потому что в последнем столбце таблицы истинности нулей меньше, чем единиц.

Способ 3. При небольшом количестве нулей можно использовать ещё один метод. Попробуем применить операцию НЕ к исходному выражению для $\overline{X}$ из предыдущего примера, без предварительного упрощения:

$$X=\overline{A\cdot\overline{B}\cdot\overline{C}+A\cdot B\cdot\overline{C}}.$$

Применяя закон де Моргана, получим:

$$X=\overline{(A\cdot\overline{B}\cdot\overline{C})}\cdot\overline{(A\cdot B\cdot\overline{C})}.$$

Используя закон де Моргана ещё два раза (для обеих скобок), находим:

$$X=(\overline{A}+B+C)\cdot(\overline{A}+\overline{B}+C).$$

Заметим, что выражение в каждой скобке *ложно* только для одной комбинации исходных данных, при которых $X=0$. Таким образом, для каждой строки таблицы истинности, где выражение равно 0, нужно построить логическую сумму, в которую переменные, равные в этой строке единице, входят с отрицанием, а равные нулю — без отрицания. Выражение для X — это произведение полученных сумм.

Такая запись — конъюнкция простых дизъюнкций — называется **конъюнктивной нормальной формой** (КНФ).

В нашем примере выражение упрощается с помощью распределительного закона для операции И и закона исключённого третьего:

$$X=(\overline{A}+B+C)\cdot(\overline{A}+\overline{B}+C)=(\overline{A}+C)+B\cdot\overline{B}=\overline{A}+C.$$

Неудивительно, что мы получили тот же ответ, что и раньше.

Иногда при упрощении выражений может потребоваться искусственный приём, который сначала вроде бы усложняет запись, но затем позволяет получить более простую форму. Например, рассмотрим выражение

$$X = \overline{A} \cdot B + \overline{A} \cdot C + \overline{B} \cdot C.$$

Учитывая, что $B + \overline{B} = 1$, можно представить второе слагаемое в виде

$$\overline{A} \cdot C = \overline{A} \cdot (B + \overline{B}) \cdot C = \overline{A} \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot C.$$

Тогда получаем

$$X = \overline{A} \cdot B + \overline{A} \cdot B \cdot C + \overline{A} \cdot \overline{B} \cdot C + \overline{B} \cdot C =$$
$$= \overline{A} \cdot B \cdot (1 + C) + (\overline{A} + 1) \cdot \overline{B} \cdot C = \overline{A} \cdot B + \overline{B} \cdot C.$$

Выводы

- Синтез логического выражения — это построение логического выражения по таблице истинности.
- Любой логической функции соответствует множество эквивалентных логических выражений.
- Для решения задачи синтеза нужно для каждой строки таблицы истинности, где функция равна 1, записать логическое выражение, истинное только для этой строки, и эти выражения сложить. Таким образом будет построено выражение в дизъюнктивной нормальной форме.
- Другой метод решения задачи синтеза — для каждой строки таблицы истинности, где функция равна 0, записать логическое выражение, ложное только для этой строки, и эти выражения перемножить. Таким образом будет построено выражение в конъюнктивной нормальной форме.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Сравните методы синтеза логических выражений с помощью ДНФ и КНФ. Когда лучше использовать каждый из них?
2. Предложите простой метод перехода от КНФ к ДНФ. Продемонстрируйте его на примере выражения $X = (A + \overline{B}) \cdot (\overline{A} + B)$. Как называется эта логическая операция?
3. Используя дополнительные источники, выясните, как перейти от ДНФ к КНФ, используя законы де Моргана. Продемонстрируйте этот метод на примере выражения $X = A \cdot \overline{B} + \overline{A} \cdot B$. Как называется эта логическая операция?

Подготовьте сообщение

а) «Совершенные нормальные формы»
б) «Карты Карно»

Проект

Сравнение методов упрощения логических выражений

§ 21
Множества и логика

Ключевые слова:

- множество
- универсальное множество
- логическое выражение
- дополнение
- пересечение
- объединение
- задача дополнения

Множества и логические выражения

Любое множество можно задать с помощью некоторого логического выражения (условия), которое истинно для каждого элемента множества и ложно для всех объектов, не входящих в это множество. Например, множество чётных чисел задаётся условием «число делится на 2 без остатка». Поэтому для выполнения операций со множествами можно успешно применять аппарат алгебры логики.

Пусть A — это логическое выражение, определяющее некоторое множество. Для того чтобы не вводить лишние обозначения, мы будем говорить «множество A», имея в виду «множество элементов, для которых выражение A истинно».

Множество $\overline{A}$ — это множество всех объектов, которые не входят в A (для них выражение A ложно). Такое множество называется **дополнением** множества A до некоторого **универсального множества** U, из которого мы выделяем множество A. Например, если A — это множество чётных чисел, то в качестве универсального множества U можно взять множество всех целых чисел. Тогда $\overline{A}$ — это множество всех нечётных чисел.

Множество $A \cdot B$ включает общие элементы множеств A и B и называется их **пересечением**. Множество $A + B$ — это **объединение** множеств A и B, т. е. все элементы, которые входят хотя бы в одно из двух множеств.

Все перечисленные множества удобно изображать с помощью диаграмм Эйлера–Венна (рис. 3.18).

$A \cdot \underline{B}$ $A + B$

Рис. 3.18

Задача дополнения

В практических задачах программирования и разработки систем управления часто необходимо построить условие (логическое выражение), которое будет выполняться при всех возможных исходных данных. Например, нужно предусмотреть все случаи неверного ввода данных и сделать так, чтобы программа правильно работала (не завершалась аварийно!) при любых действиях пользователя. Система управления ядерным реактором не должна отказывать при любых, даже неверных, действиях персонала.

Как правило, в таких задачах условие частично известно, его нужно только дополнить так, чтобы соответствующее логическое выражение было всегда истинно.

Задача 1. Построить логическое выражение A такое, что логическое выражение $A + B$ истинно для всех элементов универсального множества.

Эту задачу можно сформулировать несколько иначе, через множества: найти множество A, такое что множество $A + B$ совпадает с универсальным множеством.

Фактически выбор множества A должен обеспечить выполнение логического равенства

$$A + B = 1.$$

В логике нет операции вычитания, поэтому «простое» решение в виде $A = 1 - B$ не имеет смысла. Мы должны построить выражение A так, чтобы оно было истинным для всех случаев, когда $B = 0$, т. е. для всех объектов, не входящих в множество B. Кроме того, выражение A может быть истинно и для некоторых (или даже для всех!) элементов множества B, это не нарушит условия $A + B = 1$. Эти рассуждения приводят к формуле

$$A_{\min} = \overline{B},$$

которая определяет минимальное множество, удовлетворяющее условию задачи.

Возможен и второй вариант, при котором в логическую сумму входит отрицание логического выражения A.

Задача 2. Построить логическое выражение A, такое что логическое выражение $\overline{A} + B$ истинно для всех элементов универсального множества.

В этом случае, используя решение задачи 1, получаем $\overline{A}_{\min} = \overline{B}$. Выполняя отрицание для обеих частей этого равенства, приходим к такому результату (при отрицании вместо минимального множества получаем, наоборот, максимальное):

$$A_{\max} = B.$$

Далее мы рассмотрим несколько примеров задач, которые можно свести к задачам 1 и 2, и сразу применить готовые решения.

Задачи с отрезками

Пример 1. На числовой прямой даны два отрезка: $p = [37; 60]$ и $q = [40; 77]$. Укажите наименьшую возможную длину такого отрезка a, что выражение

$$(x \in p) \to (((x \in q) \cdot (x \notin a)) \to (x \notin p))$$

истинно при любом значении переменной x.

Решение. Обозначим логические выражения, показывающие принадлежность числа x к отрезкам:

$$P = (x \in p), \quad Q = (x \in q), \quad A = (x \in a).$$

Тогда логическое выражение, соответствующее условию задачи, может быть записано так:

$$P \to (Q \cdot \overline{A} \to \overline{P}).$$

Используя свойство импликации $A \to B = \overline{A} + B$ и закон де Моргана $\overline{A \cdot B} = \overline{A} + \overline{B}$, получаем:

$$P \to (Q \cdot \overline{A} \to \overline{P}) = \overline{P} + (Q \cdot \overline{A} \to \overline{P}) = \overline{P} + \overline{Q} + A + \overline{P} = A + \overline{P} + \overline{Q}.$$

В результате мы свели задачу к задаче 1, где $B = \overline{P} + \overline{Q}$. Её решение — условие, которое определяет минимальное множество A — получаем с помощью закона де Моргана:

$$A_{\min} = \overline{\overline{P} + \overline{Q}} = P \cdot Q.$$

Результат — это пересечение множеств P и Q, т. е. общая часть двух отрезков p и q. В нашей задаче это отрезок [40; 60], его длина — 20 (рис. 3.19).

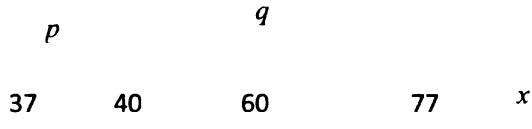


Рис. 3.19

Ответ: 20.

Пример 2. На числовой прямой даны два отрезка: $p = [10; 20]$ и $q = [25; 55]$. Укажите наибольшую возможную длину такого отрезка a, что выражение

$$(x \in a) \to ((x \in p) + (x \in q))$$

истинно при любом значении переменной x.

Решение. Будем использовать те же обозначения, что и в примере 1. Логическое выражение, соответствующее условию задачи, может быть записано в виде:

$$A \to (P + Q).$$

Используя свойство импликации $A \to B = \overline{A} + B$, получаем:

$$A \to (P + Q) = \overline{A} + P + Q.$$

Мы свели задачу к задаче 2, где $B = P + Q$. Её решение:

$$A_{\max} = B = P + Q.$$

Это условие, которое определяет максимальное множество A. Полученный результат — это объединение множеств P и Q, включающее оба отрезка p и q (рис. 3.20).

Рис. 3.20

Нужно учесть, что множество A — это один отрезок, а множество $P + Q$ — это объединение двух непересекающихся отрезков. Отрезок не может состоять из непересекающихся отрезков, поэтому обеспечить точное выполнение условия $A = P + Q$ невозможно. Самое лучшее, что можно сделать, — это выбрать

наибольший из двух отрезков, в данном случае — отрезок q. Длина этого отрезка — 30.

Ответ: 30.

Множества чисел

Пример 3. Элементами множеств a, p и q являются натуральные числа, причём

$p = \{2, 4, 6, 8, 10, 12\}$ и $q = \{4, 8, 12, 16\}$.

Известно, что выражение

$$(x \in p) \to (((x \in q) \cdot (x \notin a)) \to (x \notin p))$$

истинно при любом значении переменной x. Определите наименьшее возможное значение суммы элементов множества a.

Решение. Обозначим логические высказывания:

$$P = (x \in p), \quad Q = (x \in q), \quad A = (x \in a).$$

Тогда логическое выражение, соответствующее условию задачи, может быть записано так:

$$P \to (Q \cdot \overline{A} \to \overline{P}).$$

Заметим, что это выражение совпадает с аналогичным выражением из примера 1. Применяя те же преобразования, находим, что минимальное множество a определяется условием

$$A_{\min} = \overline{\overline{P} + \overline{Q}} = P \cdot Q.$$

Это пересечение множеств p и q, т. е. множество общих элементов p и q. Поэтому

$$a_{\min} = \{4, 8, 12\}.$$

Сумма элементов этого множества равна 24.

Ответ: 24.

Делимость чисел

В следующих задачах $ДЕЛ(n, m)$ обозначает утверждение «натуральное число n делится без остатка на натуральное число m».

Пример 4. Для какого наибольшего натурального числа a выражение[1)]

$$\neg ДЕЛ(x, a) \to (ДЕЛ(x, 6) \to \neg ДЕЛ(x, 4))$$

истинно при любом натуральном значении переменной x?

---

1) Здесь знак $\neg$ означает отрицание логического выражения, перед которым он записан.

Решение. Через D_m мы будем для краткости обозначать высказывание *ДЕЛ*(x, m): «число x делится без остатка на m». Кроме того, чтобы выделить неизвестный элемент, обозначим $A = ДЕЛ(x, a)$.

Истинным для всех x должно быть выражение

$$\overline{A} \to (D_6 \to \overline{D}_4).$$

Упростим это выражение, дважды раскрыв импликацию по правилу $A \to B = \overline{A} + B$:

$$\overline{A} \to (D_6 \to \overline{D}_4) = A + \overline{D}_6 + \overline{D}_4.$$

Мы свели задачу к задаче 1, где $B = \overline{D}_6 + \overline{D}_4$. Её решение — условие, определяющее минимальное множество A, — запишется в виде

$$A_{\min} = B = \overline{\overline{D}_6 + \overline{D}_4} = D_6 \cdot D_4.$$

Это множество всех чисел, которые делятся одновременно на 4 и 6, т. е. делятся на наименьшее общее кратное чисел 4 и 6 — число 12. Поэтому 12 — это и есть наибольшее возможное значение a, и $A_{min} = D_{12}$.

Почему 12 — это именно *наибольшее* возможное значение, хотя в результате решения задачи 1 мы получили *минимальное* множество $A_{\min}$? Дело в том, что в качестве a можно выбрать любой делитель 12: 1, 2, 3, 4, 6 или 12. Но при уменьшении a соответствующее множество D_a расширяется. Например, при $a = 1$ множество D_a включает все натуральные числа, а не только кратные 12. Поэтому максимальному a соответствует минимальное множество D_a. Брать значение a больше, чем 12, нельзя, потому что в соответствующее множество D_a войдут не все элементы множества (например, не войдёт число 12). На рисунке 3.21 показаны множества D_6, D_{12} и D_{24}.

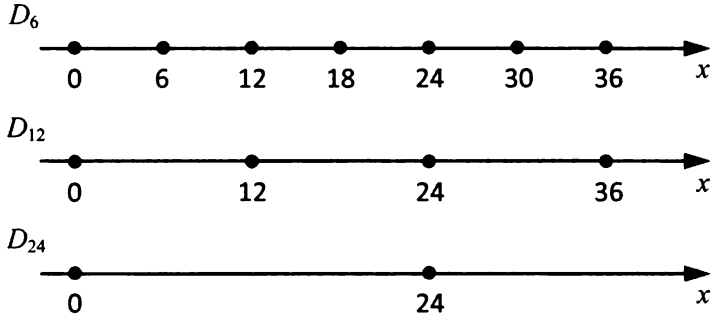


Рис. 3.21

Ответ: 12.

Пример 5. Для какого наибольшего натурального числа a выражение

$$\neg ДЕЛ(x, a) \rightarrow (\neg ДЕЛ(x, 21) \cdot \neg ДЕЛ(x, 35))$$

истинно при любом натуральном значении переменной x?

Истинным для всех x должно быть выражение

$$\overline{A} \rightarrow (\overline{D}_{21} \cdot \overline{D}_{35}).$$

Раскроем импликацию по правилу $A \rightarrow B = \overline{A} + B$:

$$\overline{A} \rightarrow (\overline{D}_{21} \cdot \overline{D}_{35}) = A + \overline{D}_{21} \cdot \overline{D}_{35}.$$

Мы свели задачу к базовой задаче 1, где $B = \overline{D}_{21} \cdot \overline{D}_{35}$. Её решение:

$$A_{\min} = \overline{\overline{D}_{21} \cdot \overline{D}_{35}} = D_{21} + D_{35}$$

определяет множество чисел, которые делятся на 21 или на 35.

Получить множество $A_{\min}$ с помощью одной операции *ДЕЛ* невозможно. Заметим, что любое множество D_a, где a — какой-нибудь общий делитель чисел 21 и 35, включает все числа, делящиеся на 21 или на 35, т. е. содержит $A_{\min}$. Так как 7 — это наибольший общий делитель этих чисел, множество D_7 — это минимальное множество, включающее $A_{\min}$, которое можно получить с помощью одной операции *ДЕЛ* (напомним, что чем больше a, тем меньше множество D_a). Поэтому наибольшее значение a равно 7.

Ответ: 7.

Поразрядные логические операции

Пример 6. Пусть выражение $n \,\&\, m$ обозначает поразрядную конъюнкцию n и m (логическое И между соответствующими битами двоичной записи этих чисел). Определите наименьшее натуральное число a, такое что выражение

$$(x \,\&\, 49 \neq 0) \rightarrow ((x \,\&\, 44 = 0) \rightarrow (x \,\&\, a \neq 0))$$

истинно при любом натуральном значении переменной x.

Решение. Поразрядная операция — это операция с отдельными битами числа. Например, пусть двоичный код числа x записывается как 1101011_2. Тогда его поразрядная конъюнкция с числом $49 = 110001_2$ выглядит так:

$$\begin{array}{rl} \text{разряды:} & 6\,5\,4\,3\,2\,1\,0 \\ x = & 1101011_2 \\ 49 = & 0110001_2 \\ \hline x \,\&\, 49 = & 0100001_2 \end{array}$$

Между соответствующими битами двух операндов (чисел x и 49) выполняется операция И (конъюнкция). Поэтому если в обоих двоичных кодах какой-то бит равен 1, то соответствующий бит результата тоже равен 1, а во всех остальных случаях этот бит результата будет равен 0. В нашем примере единичные биты — нулевой и пятый, потому что и в числе x, и в числе 49 эти биты равны 1.

Через K_m будем для краткости записи обозначать высказывание $x \,\&\, m \neq 0$ (поразрядная конъюнкция чисел x и m не равна нулю), обозначим также $A = K_a$. Тогда заданное в условии выражение можно записать в виде

$$K_{49} \to (\overline{K}_{44} \to A).$$

Дважды используя формулу $A \to B = \overline{A} + B$, преобразуем выражение к виду:

$$K_{49} \to (\overline{K}_{44} \to A) = \overline{K}_{49} + K_{44} + A.$$

Таким образом, мы получили задачу 1, где $A = \overline{K}_{49} + K_{44}$. Её решение:

$$A_{\min} = \overline{\overline{K}_{49} + K_{44}} = K_{49} \cdot \overline{K}_{44}$$

определяет множество чисел x, для которых поразрядная конъюнкция с числом 49 даёт не ноль, а та же операция с числом 44 даёт ноль. Это множество нам нужно описать с помощью одной поразрядной конъюнкции.

Пусть двоичный код числа записывается в виде $x = b_5b_4b_3b_2b_1b_0$, где b_i $(i = 1, \ldots, 5)$ — отдельные биты. Тогда поразрядные конъюнкции K_{49} и K_{44} дают следующие результаты:

K_{49}: разряды:	5	4	3	2	1	0
$x =$	b_5	b_4	b_3	b_2	b_1	b_0
$49 =$	1	1	0	0	0	1_2
$x \,\&\, 49 =$	b_5	b_4	0	0	0	b_0

K_{44}: разряды:	5	4	3	2	1	0
$x =$	b_5	b_4	b_3	b_2	b_1	b_0
$44 =$	1	0	1	1	0	0_2
$x \,\&\, 44 =$	b_5	0	b_3	b_2	0	0_2

Поскольку $K_{49} = 1$, среди битов b_5, b_4 и b_0 числа x есть ненулевые. Кроме того, поскольку $K_{44} = 0$, биты b_5, b_3 и b_2 — нулевые. Эти два условия выполняются одновременно, поэтому, объединяя их, делаем вывод, что среди битов b_4 и b_0 числа x есть ненулевые. Это утверждение может быть записано в виде поразрядной конъюнкции $K_a = (x \,\&\, a \neq 0)$, где в числе a биты с номерами 0 и 4 равны 1, а остальные равны нулю. Это число $a = 10001_2 = 17$.

Ответ: 17.

Пример 7. Определите наибольшее натуральное число a, такое что выражение

$$(x \,\&\, a \neq 0) \to ((x \,\&\, 42 = 0) \to (x \,\&\, 28 \neq 0))$$

истинно при любом натуральном значении переменной x.

Решение. Будем использовать те же обозначения, что и в предыдущем примере. Заданное логическое выражение можно переписать в виде

$$A \to (\overline{K}_{42} \to K_{28})$$

и упростить

$$A \to (\overline{K}_{42} \to K_{28}) = \overline{A} + K_{42} + K_{28},$$

таким образом сводя его к форме задачи 2, где $B = K_{42} + K_{28}$. Решение задачи:

$$A_{\max} = K_{42} + K_{28}$$

определяет множество чисел x, для которых поразрядная конъюнкция с числом 42 даёт не ноль или та же операция с числом 28 даёт не ноль. Выполним поразрядные конъюнкции:

K_{42}: разряды:	5	4	3	2	1	0
$x =$	b_5	b_4	b_3	b_2	b_1	b_0
$42 =$	1	0	1	0	1	0_2
$x \,\&\, 42 =$	b_5	0	b_3	0	b_2	0_2

K_{28}: разряды:	5	4	3	2	1	0
$x =$	b_5	b_4	b_3	b_2	b_1	b_0
$28 =$	0	1	1	1	0	0_2
$x \,\&\, 44 =$	0	b_4	b_3	b_2	0	0_2

Нам нужно описать множество чисел, для которых выполняется K_{42} (среди битов b_5, b_3 и b_1 есть ненулевые) или выполняется K_{28} (среди битов b_4, b_3 и b_2 есть ненулевые). Для всех этих чисел будет выполняться условие K_a, где в числе a все эти биты (с номерами 1, 2, 3, 4 и 5) единичные. Это число $a = 111110_2 = 62$.

Ответ: 62.

Выводы

- Любое множество можно задать с помощью некоторого логического выражения (условия), которое истинно для каждого элемента множества и ложно для всех объектов, не входящих в это множество.
- Если логическое выражение A определяет некоторое множество, то выражение $\overline{A}$ определяет дополнение этого множества до универсального множества.
- Задача 1: построить логическое выражение A, такое что логическое выражение $A + B$ истинно для всех элементов универсального множества. Минимальное множество, которое является решением задачи 1, определяется условием $A_{\min} = B$.

- Задача 2: построить логическое выражение A, такое что логическое выражение $\overline{A} + B$ истинно для всех элементов универсального множества. Максимальное множество, которое является решением задачи 2, определяется условием $A_{max} = B$.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Для каждого из следующих множеств выберите некоторое универсальное множество и определите дополнение заданного множества до универсального множества:
   а) множество натуральных чисел;
   б) множество иррациональных чисел;
   в) множество автомобилей с бензиновыми двигателями;
   г) множество двоичных кодов длиной 8 бит.
2. Нарисуйте в тетради диаграммы Эйлера–Венна для областей, заданных логическими выражениями
   а) $(A + B) \cdot \overline{C}$; б) $A \cdot B + \overline{C}$; в) $(A \leftrightarrow B) \leftrightarrow C$.

Проект

Битовые логические операции

§ 22
Предикаты и кванторы

Ключевые слова:

- предикат
- квантор
- квантор всеобщности
- квантор существования

В предыдущих параграфах 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 этих высказывания истинны, то Сократ тоже смертен. Однако алгебра логики не позволяет это доказать. В таких случаях приходится использовать

более сложный математический аппарат, с которым мы познакомимся в этом параграфе.

Рассмотрим утверждение «В городе N живёт более 2 миллионов человек». Его нельзя считать логическим высказыванием, поскольку непонятно, о каком городе идёт 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 живёт более 2 миллионов человек».

Если мы задаём конкретные значения переменных, предикат превращается в логическое высказывание. Например, для предиката $P(N)$ мы получим истинное высказывание для N = «Москва» и ложное 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»;
Живёт(x, y) = «x живёт 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: $P(x) = (x > 0)$. Множество пар чисел, сумма которых равна 1, задаётся предикатом $P(x, y) = (x + y = 1)$, который зависит от двух переменных.

Существуют предикаты, которые справедливы для всех допустимых значений переменных. Например, $P(x) = (x^2 \geq 0)$, опре-

делённый на множестве всех вещественных чисел. В таком случае используют запись $\forall x P(x)$; это означает: «при любом x предикат $P(x)$ справедлив». Знак $\forall$ — это буква «А», развёрнутая вверх ногами (от англ. *all* — всё); 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:

$$\forall n: \quad 1 + 2 + \ldots + n = \frac{n(n+1)}{2}.$$

Часто используют ещё один квантор — **квантор существования** $\exists$ (зеркальное отражение буквы E, от англ. *exist* — существовать). Знак $\exists$ означает «существует», «хотя бы один». Например, если $P(x) = (x - 5 > 0)$, то можно записать $\exists x P(x)$, что означает «существует x, такой что $x - 5 > 0$». Это уже высказывание, а не предикат, потому что можно сразу установить его истинность. Высказывание $\exists x P(x)$ истинно, так как существует x, удовлетворяющий этому условию, например $x = 6$. Запись $\forall x P(x)$ — это тоже высказывание, но оно ложно, потому что неравенство $x - 5 > 0$ 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 $\forall x \exists y (x + y = 0)$. 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 $\exists y \forall x (x + y = 0)$ означает «существует такое значение y, что для любого x выполняется равенство $x + y = 0$», это ложное высказывание.

Теперь давайте вернемся к Сократу, точнее, к двум высказываниям, приведённым в начале параграфа. Как записать утверждение «Все люди смертны»? Можно сказать иначе: для любого x верно: «если x — человек, то x смертен». Вспоминаем, что связка «если..., то» записывается как импликация, а выражение «для любого x» — в виде квантора $\forall x$. Поэтому получаем:

$$\forall x (P(x) \to Q(x)),$$

где $P(x)$ = «x — человек», $Q(x)$ = «x — смертен». Так как утверждение $P(x) \to Q(x)$ верно для любого x, оно также верно при подстановке x = Сократ:

$$P(\text{Сократ}) \to Q(\text{Сократ}) = 1.$$

Поскольку Сократ — человек, P(Сократ) = 1. Поэтому с помощью таблицы истинности для импликации мы находим, что Q(Сократ) = 1, т. е. «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 как «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. В таких системах часто применяется язык логического программирования Пролог, в котором программа представляет собой набор данных и правила вывода новых результатов из этих данных.

Выводы

- Предикат — это высказывание, содержащее переменные. Предикат можно рассматривать как логическую функцию.
- Квантор — это знак или выражение, обозначающее количество.
- В математике и логике широко используются квантор существования $\exists$ («существует», «хотя бы один») и квантор всеобщности $\forall$ («любой», «для любого», «для всех»).

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как превратить предикат «В городе N живёт более 2 миллионов человек» в логическое высказывание? Предложите несколько способов.
2. Сравните логические высказывания $\forall x \exists y\, P(x, y)$ и $\exists y \forall x P(x, y)$, где $P(x, y)$ — предикат от x и y.
3. Какие из этих выражений — предикаты, а какие — логические высказывания?
   а) $\forall x\, P(x, y)$; б) $\exists y\, P(x, y)$;
   в) $\forall x \exists y\, P(x, y)$; г) $\exists y \forall x\, P(x, y)$.

Подготовьте сообщение

а) «Что такое предикаты?»
б) «Кванторы в математике и логике»
в) «Логические языки программирования»

Проект

Решение задач с помощью языка Пролог

§ 23
Логические элементы компьютера

Ключевые слова:

- логический элемент
- логическая функция
- триггер
- регистр
- полусумматор
- сумматор

Простейшие элементы

Современный компьютер состоит из большого количества *логических элементов* — электронных схем, выполняющих логические операции. Обозначения простейших элементов приводятся на рис. 3.22 (ГОСТ 2.743-91). Заметьте, что небольшой кружок на выходе (или на входе) обозначает операцию «НЕ» (отрицание, инверсию).

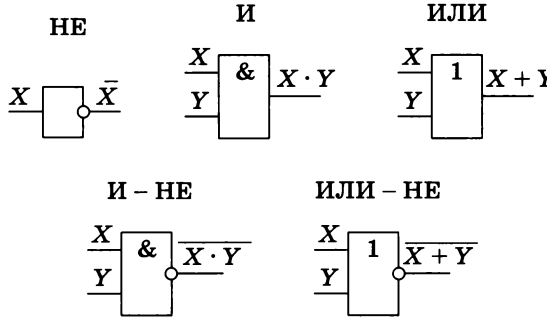


Рис. 3.22

Может показаться, что для реализации сложных логических функций нужно много разных логических элементов. Однако, как вы знаете, 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 И-НЕ. Эта функция (штрих Шеффера) позволяет реализовать НЕ, И и ИЛИ, а значит и любую другую операцию.

Если нужно составить схему по известному логическому выражению, её 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 (рис. 3.23).

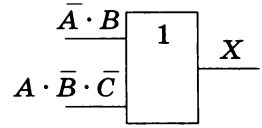


Рис. 3.23

Для того чтобы получить на первом входе $\overline{A} \cdot B$, нужно умножить $\overline{A}$ на B, поэтому добавляем элемент И (рис. 3.24).

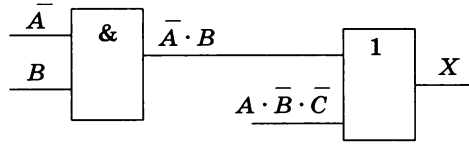


Рис. 3.24

Чтобы получить $\overline{A}$, ставим элемент НЕ (рис. 3.25).

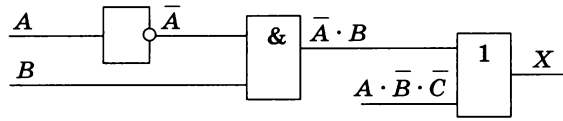


Рис. 3.25

Аналогично разбираем вторую ветку, которая поступает на второй вход элемента ИЛИ (рис. 3.26).

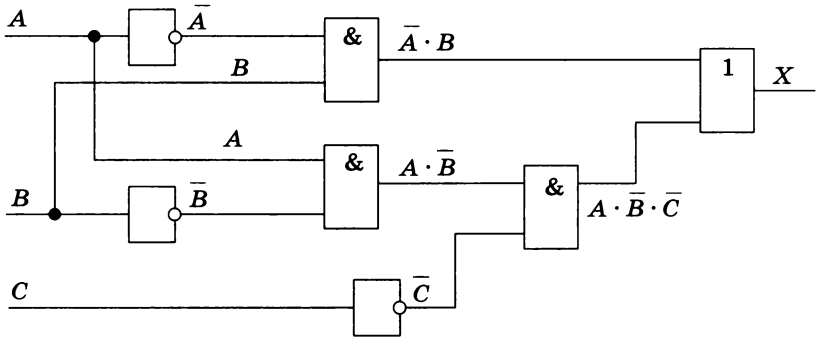


Рис. 3.26

Схема составлена, её входами являются исходные сигналы A, B и C, а выходом — X.

Триггер

Слово «триггер» происходит от английского слова *trigger* — «защёлка» или спусковой крючок[1]. 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 современных компьютерах на основе триггеров строится быстродействующая оперативная память. Один триггер способен хранить один бит данных. Соответственно, для того чтобы запомнить 1 байт данных, требуется 8 триггеров, а для хранения 1 Кбайта данных — $8 \cdot 1024 = 8192$ триггера.

Триггеры бывают разных типов. Самый распространённый — это ***RS*-триггер**. Он имеет два входа, которые обозначаются как S (англ. *set* — установить) и R (англ. *reset* — сброс), и два выхода — Q и $\overline{Q}$, причём выходной сигнал $\overline{Q}$ является логическим отрицанием сигнала Q (если $Q = 1$, то $\overline{Q} = 0$, и наоборот). *RS*-триггер можно построить на двух элементах И-НЕ или на двух элементах ИЛИ-НЕ. На рисунке 3.27 показано условное обозначение RS-триггера, внутреннее устройство триггера на элементах ИЛИ—НЕ и его таблица истинности.

[1] В английском языке триггер называется *flip-flop*.

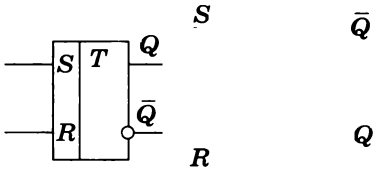


Рис. 3.27

Триггер использует так называемые **обратные связи** — сигналы с выхода каждой схемы ИЛИ-НЕ поступают на вход соседней схемы. Именно это позволяет хранить информацию.

Построим таблицу истинности триггера. Начнём с варианта, когда $S = 0$ и $R = 1$. 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 $Q = 0$ (рис. 3.28).

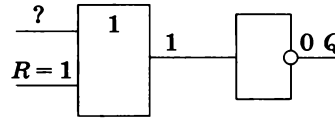


Рис. 3.28

Тогда на входе другого элемента ИЛИ-НЕ будут два нуля, а на выходе $\bar{Q}$ — единица (рис. 3.29).

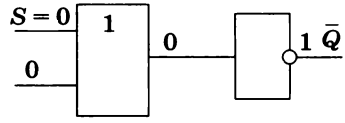


Рис. 3.29

Поскольку основным выходом считается Q, мы записали в триггер значение 0. Схема симметрична, поэтому легко догадаться, что при $S = 1$ и R = 0 мы запишем в триггер 1 ($Q = 1$).

Теперь рассмотрим случай, когда $S = 0$ и $R = 0$. На выходе первого элемента ИЛИ будет сигнал $Q + 0 = Q$, поэтому на выходе $\bar{Q}$ останется его предыдущее значение (рис. 3.30).

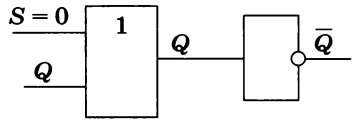


Рис. 3.30

Аналогично легко показать, что на выходе Q тоже остаётся 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 запрещён.

Для хранения многоразрядных данных триггеры объединяются в единый блок, который называется **регистром.** Регистры (размером от 8 до 64 бит) используются во всех процессорах для временного хранения промежуточных результатов.

Над регистром, как над единым целым, можно выполнять ряд стандартных операций: сбрасывать (обнулять), заносить в него код и т. д. Часто регистры способны не просто хранить данные, но и обрабатывать их. Например, существуют регистры-счётчики, которые подсчитывают количество импульсов, поступающих на вход.

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 учётом 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 рис. 3.31.

A	*B*	*P*	*S*
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Рис. 3.31

Легко увидеть, что столбец P — это результат применения операции И ко входам A и B, а столбец S — результат исключающего ИЛИ:

$$P = A \cdot B, \quad S = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}.$$

Формулу для S можно также записать в таком виде

$$S = \overline{A} \cdot B + A \cdot \overline{B} = (A + B) \cdot (\overline{A} + \overline{B}) =$$
$$= (A + B) \cdot \overline{(A \cdot B)} = (A + B) \cdot \overline{P},$$

что позволяет построить полусумматор, используя всего 4 простейших элемента (рис. 3.32).

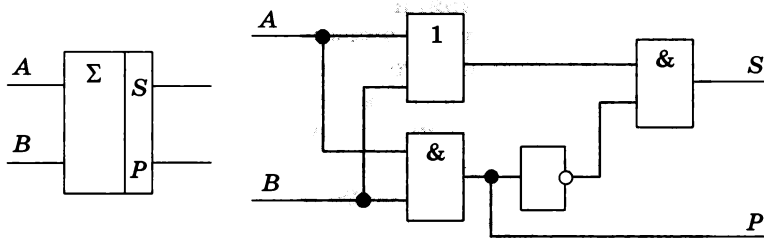


Рис. 3.32

Слева показано условное обозначение полусумматора, греческая буква Σ здесь (и в математике) обозначает сумму.

Полный **одноразрядный сумматор** учитывает также и третий бит C — перенос из предыдущего разряда. Сумматор имеет три входа и два выхода. Таблица истинности и обозначение сумматора показаны на рис. 3.33 и 3.34.

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

Рис. 3.33

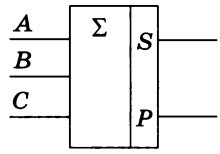


Рис. 3.34

Логические функции для выходов сумматора вы можете найти самостоятельно.

Сумматор можно построить с помощью двух полусумматоров и одного элемента ИЛИ (рис. 3.35).

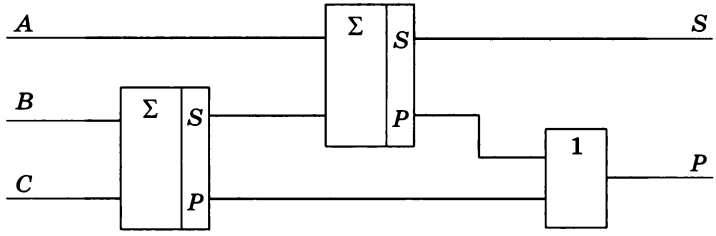


Рис. 3.35

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 даёт 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 3.36 показано, как складываются два трёхразрядных числа: $X = 110_2$ и $Y = 011_2$. Сумма $Z = 1001_2$ состоит из четырёх бит, поэтому на выходе последнего сумматора бит переноса будет равен 1 (см. рис. 3.36).

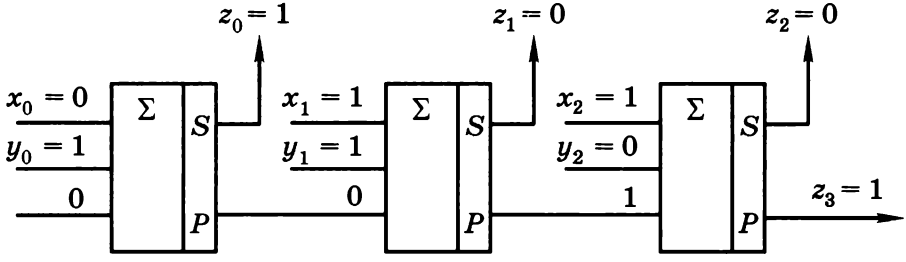


Рис. 3.36

Сложение начинается с самого младшего (нулевого) разряда. На входы первого сумматора подаются младшие биты исходных чисел, x_0 и y_0, а на третий вход — ноль (нет переноса из предыдущего разряда). Выход S первого сумматора — это младший бит результата, z_0, а его выход P (перенос) передаётся на вход второго сумматора и т. д. Выход P последнего из сумматоров представляет собой дополнительный разряд суммы, т. е. z_3.

Сумматор с последовательным переносом, показанный на рис. 3.36, работает слишком медленно. Поэтому в реальных процессорах применяют схемы с ускоренным переносом, которые выполняют сложение намного быстрее, но используют дополнительные логические элементы.

Сумматор играет важную роль не только при сложении чисел, но и при выполнении других арифметических действий. Фактически он является основой арифметического устройства современного компьютера.

Выводы

- Современный компьютер состоит из большого количества логических элементов — электронных схем, выполняющих логические операции.
- Для любой логической функции можно построить схему, использующую только элементы НЕ, И и ИЛИ.
- Триггер — это электронная схема, которая может находиться только в двух состояниях (их можно обозначить как 0 и 1) и способна почти мгновенно переходить из одного состояния в другое.
- Сумматор — это электронная схема, предназначенная для сложения двух битов с учётом переноса из предыдущего разряда.
- Для сложения двоичных данных используют многоразрядные сумматоры.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Почему для RS-триггера комбинация входов $S = 1$ и $R = 1$ запрещена?
2. Чем отличается одноразрядный сумматор от полусумматора?
3. Как можно построить сумматор с помощью двух полусумматоров?
4. Постройте логические выражения для выходов сумматора и нарисуйте соответствующие им схемы.
5. Как используется перенос в многоразрядном сумматоре?

Подготовьте сообщение

а) «Обозначения логических элементов в России и за рубежом»
б) «Типы триггеров»
в) «Что такое регистр?»
г) «Что такое мультиплексор?»
д) «Что такое демультиплексор?»
е) «Шифратор и дешифратор»

Проекты

а) Логические схемы в составе компьютера
б) Логические устройства в системах автоматики
в) Сравнение различных типов триггеров

ЭОР к главе 3 на сайте ФЦИОР (http://fcior.edu.ru)

- Высказывание. Простые и сложные высказывания
- Основные логические операции
- Теория множеств
- Логические законы и правила преобразования логических выражений
- Построение отрицания к простым высказываниям, записанным на русском языке
- Построение отрицания к сложным высказываниям, записанным на русском языке
- Решение логических задач
- Сумматор двоичных чисел

Практические работы к главе 3

Работа № 7 «Тренажёр "Логика"»
Работа № 8 «Исследование запросов для поисковых систем»
Работа № 9 «Логические элементы компьютера»

Глава 4
КОМПЬЮТЕРНАЯ АРИФМЕТИКА

§ 24
Особенности представления чисел в компьютере

Ключевые слова:

- разрядная сетка
- переполнение
- антипереполнение

На уроках математики вы никогда не обсуждали, как хранятся числа. Математика — это теоретическая наука, для которой неважно, записаны ли числа на маленьком или большом листе бумаги, зафиксированы ли с помощью счётных палочек, счётов или внутри полупроводниковой схемы. Поэтому число в математике может состоять из любого количества цифр, которое требуется в решаемой задаче.

В то же время инженеры, разрабатывающие компьютер, должны спроектировать реальное устройство из определённого количества деталей. Поэтому количество разрядов, отведённых для хранения каждого числа, ограничено, и точность вычислений тоже ограничена. Из-за этого при компьютерных расчётах могут возникать достаточно серьёзные проблемы. Например, сумма двух положительных чисел может получиться отрицательной, а выражение $A + B$ может совпадать с A при ненулевом B. В этой главе мы рассмотрим важные особенности компьютерной арифметики, которые нужно учитывать при обработке данных. В первую очередь они связаны с тем, как размещаются целые и вещественные числа в памяти компьютера.

Предельные значения чисел

Как вы уже поняли, числа, хранящиеся в компьютере, не могут быть сколь угодно большими и имеют некоторые предельные

значения. Представим себе некоторое вычислительное устройство, которое работает с четырёхразрядными неотрицательными целыми десятичными числами (рис. 4.1).

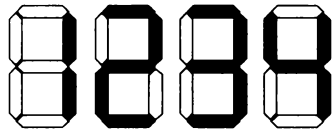


Рис. 4.1

Для вывода чисел используется четырёхразрядный индикатор, на котором можно отобразить числа от 0 (все разряды числа минимальны) до 9999 (все разряды максимальны) — рис. 4.2.

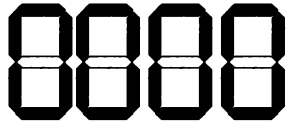
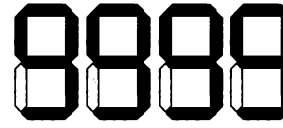
Диапазон от  до 

Рис. 4.2

Вывести на такой индикатор число 10 000 невозможно: не хватает технического устройства для пятого разряда. Такая «аварийная» ситуация называется *переполнением разрядной сетки* или просто *переполнением* (англ. *overflow* — переполнение «сверху»).

Переполнение разрядной сетки — это ситуация, когда число, которое требуется сохранить, не умещается в имеющемся количестве разрядов вычислительного устройства.

В нашем примере переполнение возникает при значениях, бóльших $9999 = 10^4 - 1$, где 4 — это количество разрядов. В общем случае, если в системе счисления с основанием B для записи числа используется K разрядов, максимальное допустимое число вычисляется по аналогичной формуле[1)]

$$C_{\max} = B^K - 1.$$

Вспомните, что именно эта формула для $B = 2$ неоднократно применялась в главе 2.

Подчеркнём, что переполнение никак не связано с системой счисления: оно вызвано *ограниченным количеством разрядов устройства* и не зависит от количества возможных значений в каждом из этих разрядов.

---

1) Докажите эту формулу самостоятельно, например подсчитав количество всех возможных комбинаций значений цифр в K разрядах.

Посмотрим теперь, что будет, если наше устройство работает не только с целыми, но и с дробными числами. Пусть, например, один из четырёх разрядов относится к целой части числа, а остальные три — к дробной (рис. 4.3).

Рис. 4.3

Конечно, эффект переполнения сохранится и здесь: максимально допустимое число равно 9,999. Кроме того, дробная часть числа тоже ограничена, поэтому любое число, имеющее более трёх цифр после запятой, не может быть представлено точно: младшие цифры придётся отбрасывать (или округлять).

Не все вещественные числа могут быть представлены в компьютере точно.

При ограниченном числе разрядов дробной части существует некоторое минимальное ненулевое значение C_{min}, которое можно записать на данном индикаторе (в нашем примере это 0,001, рис. 4.4).

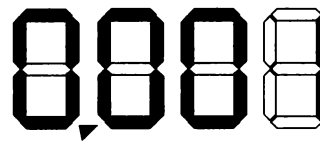


Рис. 4.4

В общем случае, если число записано в системе счисления с основанием B и для хранения дробной части числа используется F разрядов, имеем:

$$C_{min} = B^{-F}.$$

Любое значение, меньшее чем C_{min}, неотличимо от нуля. Такой эффект принято называть *антипереполнением* (англ. *underflow* — переполнение «снизу»).

Кроме того, два дробных числа, отличающиеся менее чем на C_{min}, для компьютера неразличимы. Например, 1,3212 и 1,3214 на нашем индикаторе выглядят совершенно одинаково (рис. 4.5).

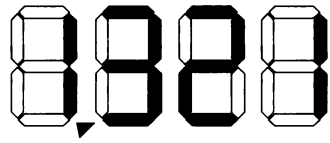


Рис. 4.5

Дополнительная погрешность появляется при переводе дробных чисел из десятичной системы счисления в двоичную. При этом даже некоторые «круглые» числа (например, 0,2) в памяти компьютера представлены неточно, потому что в двоичной системе они записываются как бесконечные дроби и их приходится округлять до заданного числа разрядов.

Так как вещественные числа хранятся в памяти приближённо, сравнивать их (особенно если они являются результатами сложных расчётов) необходимо с большой осторожностью. Пусть при вычислениях на компьютере получили $X = 10^{-6}$ и $Y = 10^{6}$. Дробное значение X будет неточным, и произведение $X \cdot Y$ может незначительно отличаться от 1. Поэтому при сравнении вещественных чисел в компьютере условие «равно» использовать нельзя. В таких случаях числа считаются равными, если их разность достаточно мала по модулю. В данном примере нужно проверять условие $|1 - X \cdot Y| < \varepsilon$, где ε — малая величина, которая задаёт нужную точность вычислений. К счастью, для большинства практических задач достаточно взять ε порядка $10^{-2}...10^{-4}$, а ошибка компьютерных расчётов обычно значительно меньше[1)] (не более 10^{-7}).

Введение разряда для знака числа не меняет сделанных выше выводов, только вместо нулевого минимального значения появляется отрицательное, которое зависит от разрядности (оно равно –9999 в обсуждаемом выше примере).

Различие между вещественными и целыми числами

Существуют величины, которые по своей природе могут принимать только целые значения, например счётчики повторений каких-то действий, количество людей и предметов, координаты пикселей на экране и т. п. Кроме того, как показано в главе 2, кодирование нечисловых видов данных (текста, изображений, звука) сводится именно к целым числам.

---

1) Тем не менее встречаются ситуации, когда вычислительные трудности всё же возникают: классический пример — разность близких по значению десятичных дробей, отличающихся в последних значащих цифрах.

Чтобы сразу исключить все возможные проблемы, связанные с неточностью представления в памяти вещественных чисел, целочисленные данные кодируются в компьютерах особым образом.

Целые и вещественные числа в компьютере хранятся и обрабатываются по-разному.

Операции с целыми числами, как правило, выполняются значительно быстрее, чем с вещественными. Не случайно в ядре современных процессоров реализованы только целочисленные арифметические действия, а для вещественной арифметики используется специализированный встроенный блок — *математический сопроцессор*.

Кроме того, использование целых типов данных позволяет экономить компьютерную память. Например, целые числа в диапазоне от 0 до 255 в языке Паскаль можно хранить в переменных типа `byte`, которые занимают всего один байт в памяти. В то же время самое «короткое» вещественное число (типа `single`) занимает четыре байта памяти.

Наконец, только для целых чисел определены операции деления нацело и нахождения остатка от деления. В некоторых задачах они удобнее, чем простое деление с получением дробного (к тому же не совсем точного) результата: например, без них не обойтись при вычислении суммы цифр какого-то числа.

Таким образом, для всех величин, которые не могут иметь дробных значений, нужно использовать целочисленные типы данных.

Дискретность представления чисел

Из главы 2 вы знаете, что существует непрерывное и дискретное представление информации. Их принципиальное различие состоит в том, что дискретная величина может принимать конечное количество различных значений в заданном интервале, а непрерывная имеет бесконечно много возможных значений. Для нашего обсуждения важно, что:

- целые числа дискретны;
- вещественные (действительные, дробные) числа непрерывны;
- современный компьютер работает только с дискретными данными.

Таким образом, для хранения вещественных чисел в памяти компьютере нужно выполнить *дискретизацию* — записать непрерывную величину в дискретной форме. При этом может происходить искажение данных, поэтому большинство трудностей в

компьютерной арифметике (антипереполнение, приближённость представления дробной части и др.) связано именно с кодированием дробных чисел.

Программное повышение точности вычислений

Современные модели процессоров *Intel* «умеют» обрабатывать 8-, 16-, 32- и 64-разрядные двоичные целые числа, а также (в математическом сопроцессоре) 32-, 64- и 80-разрядные вещественные числа. Для большинства практических задач такой разрядности вполне достаточно. Если для каких-либо особо точных расчётов требуется повысить разрядность вычислений, это можно сделать программно. Например, можно считать, что 4 последовательно хранящихся целых 64-разрядных числа — это единое «длинное» число, и написать программу обработки таких «удлинённых» чисел. Очень удобно хранить числа в виде последовательности десятичных цифр[1)], правда, программы, выполняющие обработку таких чисел, получаются сложными и медленными.

Использование этих и других программных методов позволяет увеличить разрядность обрабатываемых чисел по сравнению с аппаратной разрядностью компьютера. Однако ограничение разрядности (и связанный с ним эффект переполнения) всё равно остаётся: в программу заложено *конкретное* число разрядов, да и объём памяти компьютера конечен.

Выводы

- Для хранения чисел в памяти компьютера используется конечное число разрядов. Из-за этого числа в компьютере имеют ограниченный диапазон, а результаты вычислений могут быть неточными.
- Переполнение разрядной сетки — это ситуация, когда число, которое требуется сохранить, не умещается в имеющемся количестве разрядов вычислительного устройства.
- Целые и вещественные числа в компьютере хранятся и обрабатываются по-разному.
- Не все вещественные числа могут быть представлены в компьютере точно. При вычислениях с вещественными числами может накапливаться ошибка.

Нарисуйте в тетради интеллект-карту этого параграфа.

---

1) Такие задачи часто даются на школьных олимпиадах по информатике; для них даже придумано специальное название — *«длинная» арифметика*.

Вопросы и задания

1. Чем отличается компьютерная арифметика от «обычной»?
2. Почему диапазон чисел в компьютере ограничен? Связано ли это с двоичностью компьютерной арифметики?
3. Что такое переполнение разрядной сетки?
4. Какие проблемы появляются при ограниченном числе разрядов в дробной части?
5. Что, по-вашему, опаснее для вычислений — переполнение или антипереполнение?

*6. Может ли антипереполнение сделать невозможными дальнейшие вычисления?

7. Сколько бит информации несёт знаковый разряд?
8. Приведите примеры величин, которые по своему смыслу могут иметь только целые значения.
9. Какие преимущества даёт разделение в компьютере целых и вещественных (дробных) чисел?
10. Какая математическая операция между двумя целыми числами может дать в результате нецелое число?
11. Чем отличается деление для целых и вещественных чисел?
12. Вспомните определение дискретных и непрерывных величин. Какие множества чисел в математике дискретны, а какие — нет? Ответ обоснуйте.
13. Объясните, почему ограниченность разрядов дробной части приводит к нарушению свойства непрерывности.
14. Можно ли организовать вычисления с разрядностью, превышающей аппаратную разрядность компьютера? Попробуйте предложить способы решения этой задачи.

Проект

Повышение точности вычислений

§ 25
Хранение в памяти целых чисел

Ключевые слова:

- беззнаковые данные
- данные со знаком
- знаковый разряд
- прямой код
- дополнительный код

Целые числа без знака

Беззнаковые (англ. *unsigned*) типы данных, т. е. величины, не имеющие отрицательных значений, широко используются в вычислительной технике. Дело в том, что в задачах, решаемых на компьютерах, есть много таких значений: всевозможные счётчики (количество повторений циклов, число параметров в списке или символов в тексте), количество людей или предметов и др.

Чтобы закодировать целое число без знака, достаточно перевести его в двоичную систему счисления и дополнить слева нулями до нужной разрядности. Например, число 28 записывается в 8-разрядную ячейку так:

0001 1100

Это же число в 16-разрядном представлении будет иметь слева ещё 8 нулей. Восьмиразрядные коды некоторых характерных чисел приведены в табл. 4.1.

Таблица 4.1

X_{10}	0	1	...	127	128	129	...	255
X_{16}	00	01	...	7F	80	81	...	FF
X_2	0000 0000	0000 0001	...	0111 1111	1000 0000	1000 0001	...	1111 1111

Минимальное значение для беззнаковых целых чисел всегда равно 0 (все разряды нулевые), а максимальное число $X_{max} = 2^{K-1}$ состоит из всех единиц и определяется разрядностью (количеством бит) K (табл. 4.2).

Таблица 4.2

K, бит	8	16	32	64
$X_{max} = 2^{K-1}$	255	65 535	4 294 967 295	18 446 744 073 709 551 615

Возникает вопрос: что будет, если увеличить максимальное число в K-битной ячейке на единицу? Рассмотрим случай $K = 8$ и попытаемся прибавить единицу к числу $255_{10} = 1111\ 1111_2$. Добавляя дополнительный бит слева, получим:

$$\begin{array}{rr} & 0\,|\,1111\ 1111 \\ + & 0\,|\,0000\ 0001 \\ \hline & 1\,|\,0000\ 0000 \end{array}$$

Отбросив несуществующий дополнительный разряд[1)], получаем: 255 + 1 = 0. Именно это произойдёт в реальном компьютере.

[1)] На самом деле для того, чтобы обнаружить факт переполнения, этот разряд сохраняется в специальном управляющем бите процессора, который называется *битом переноса*.

Говорят, что при K разрядах арифметика выполняется «по модулю 2^K», т. е. при $K = 8$ имеем[1]:

$$(255 + 1) \bmod 256 = 256 \bmod 256 = 0.$$

Вместе с тем, вычитая единицу из минимального значения 0, к которому добавлен старший единичный разряд за пределами 8-битной ячейки, получим $1111\,1111_2 = 255_{10}$ (проверьте это самостоятельно).

Можно заметить, что при многократном увеличении числа на единицу мы доходим до максимального значения и *скачком возвращаемся к минимальному*. При вычитании единицы получается обратная картина — дойдя до минимума (нуля), мы сразу перескакиваем на максимум (255). Поэтому для изображения допустимого диапазона чисел лучше подходит не отрезок числовой оси (как в математике), а окружность (рис. 4.6).

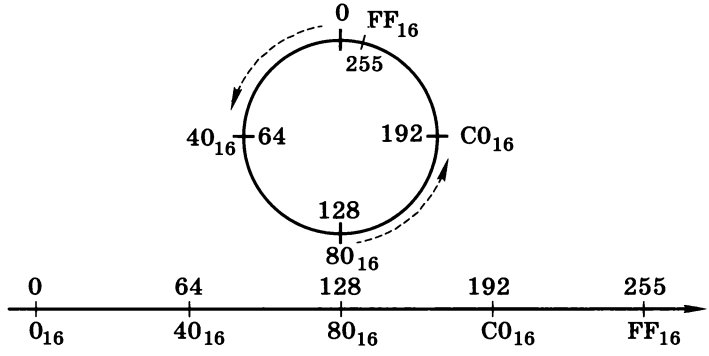


Рис. 4.6

Факт переполнения всегда фиксируется процессором, но выполнение программы не прерывается. Программе (точнее, программисту) предоставляется возможность как-то реагировать на переполнение или «не заметить» его.

Целые числа со знаком

Теперь рассмотрим числа со знаком (англ. *signed*). Для того чтобы различать положительные и отрицательные числа, в двоичном коде выделяется один бит для хранения знака числа — **знаковый разряд**. По традиции для этого используют самый старший бит, причём нулевое значение в нём соответствует знаку «плюс», а единичное — знаку «минус». Ноль формально является положительным числом, так как все его разряды, включая знаковый, нулевые.

[1] Здесь запись a **mod** b обозначает остаток от деления a на b.

Поскольку один бит выделяется для хранения информации о знаке, ровно половина из всех 2^K чисел будут отрицательными. Учитывая, что одно значение — нулевое, положительных чисел будет на единицу меньше, т. е. допустимый диапазон значений оказывается несимметричным.

Положительные числа записываются в знаковой форме так же, как и в беззнаковой, но для значения остаётся на один разряд меньше. А как поступить с отрицательными числами? Первое, что приходит в голову, это кодировать отрицательные значения точно так же, как и положительные, только записывать в старший бит единицу. Такой способ кодирования называется **прямым кодом**. Несмотря на свою простоту и наглядность, он не применяется в компьютерах для представления целых чисел[1]. Это неудобно, потому что действия над числами, записанными в прямом коде, выполняются по-разному для разных сочетаний знаков чисел. Поэтому в современных компьютерах отрицательные числа кодируются с помощью другого метода, который менее нагляден, но позволяет выполнять арифметические действия с положительными и отрицательными числами по одному и тому же алгоритму.

Как же представить целые числа, чтобы арифметика выглядела максимально просто? Попробуем, например, вычислить код, соответствующий числу –1. Для этого просто вычтем из нуля единицу:

$$\begin{array}{rcl} & 1 & 0000\ 0000 \\ - & 0 & 0000\ 0001 \\ \hline & 0 & 1111\ 1111 \end{array}$$

Чтобы вычитание «состоялось», придётся занять из несуществующего старшего бита единицу, что не очень естественно, но зато быстро приводит к правильному результату[2]. Заметим, что фактически мы вычитали не из 0, а из 256. В общем случае вычисление происходит по формуле $2^K - X$, где для данного примера $K = 8$, а $X = 1$.

Однако предложенный способ перевода не слишком хорош, поскольку мы использовали дополнительный «несуществующий» разряд. Вместо этого можно применить равносильный алгоритм:

$$256 - X = (255 - X) + 1 = \mathbf{not}\ X + 1.$$

---

[1] Тем не менее прямой код используется в представлении вещественных чисел.

[2] Для проверки можно прибавить к полученному коду единицу, в результате должен получиться ноль.

Здесь **not** обозначает логическую операцию НЕ (инверсию), применяемую к каждому биту числа отдельно (все нули заменяются на единицы и наоборот).

Итак, для получения кода целого числа $(-X)$ нужно:

Алгоритм А1

1) Выполнить инверсию каждого разряда двоичного представления числа X. Такой код называется **обратным**.
2) К полученному результату прибавить единицу.

В результате получается **дополнительный код** — он *дополняет* число X до 2^K (если сложить его с числом X, мы получим 2^K).

Алгоритм А1 приводится в большинстве учебников, но его можно немного изменить так, чтобы облегчить человеку «ручные» вычисления:

Алгоритм А2

1) Вычислить число $X - 1$ и перевести его в двоичную систему.
2) Выполнить инверсию каждого разряда результата.

Оба алгоритма дают одинаковые результаты, но А2 для человека существенно проще, потому что ему легче вычесть единицу в «родной» десятичной системе, чем прибавлять её в двоичной (при использовании алгоритма А1).

Наконец, оба пункта алгоритма А1 можно объединить, получив ещё один вариант:

Алгоритм А3

Выполнить инверсию всех старших битов числа, кроме последней (младшей) единицы и тех нулей, которые стоят после неё.

Например, определим дополнительный код числа «–16», которое хранится в 8-разрядной ячейке. Здесь $X = 16$. Используя алгоритмы А1 и А2, получаем:

	А1
X_2	$0001\ 0000_2$
not	$1110\ 1111_2$
+1	$1111\ 0000_2$

	А2
$X - 1$	15
$(X - 1)_2$	$0000\ 1111_2$
not	$1111\ 0000_2$

Применение алгоритма А3 к числу $16 = 00010000_2$ сводится к замене первых трёх нулей единицами: $1111\ 0000_2$.

Для проверки можно сложить полученный результат с исходным числом и убедиться, что сумма будет равна нулю (перенос из старшего разряда не учитываем). Повторное применение любого из алгоритмов А1–А3 всегда приводит к восстановлению первоначального числа (убедитесь в этом самостоятельно). Это свойство также удобно использовать для проверки.

В таблице 4.3 показаны шестнадцатеричные и двоичные коды некоторых характерных 8-разрядных чисел.

Таблица 4.3

X_{10}	–128	–127	...	–1	0	1	...	127
X_{16}	80	81	...	FF	00	01	...	7F
X_2	1000 0000	1000 0001	...	1111 1111	0000 0000	0000 0001	...	0111 1111

Обратите внимание на скачок при переходе от –1 к 0 и на два граничных значения: 127 и –128. «Кольцо» для чисел со знаком выглядит так, как показано на рис. 4.7.

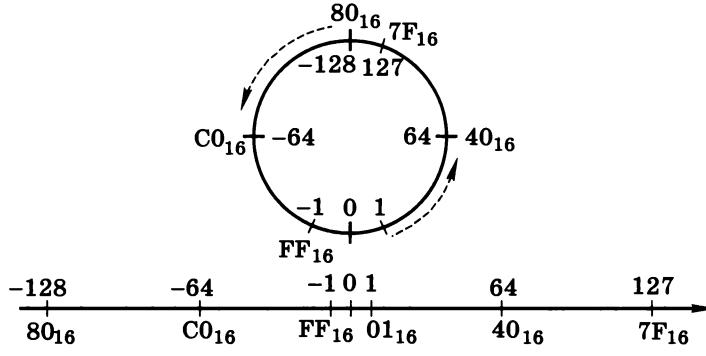


Рис. 4.7

Чтобы сравнить коды целых чисел без знака и со знаком, объединим обе таблицы (4.1 и 4.3) — получим табл. 4.4.

Таблица 4.4

Код	0	1	2	...	7F	80	81	...	FE	FF
Без знака	0	1	2	...	127	128	129	...	254	255
Со знаком	0	1	2	...	127	–128	–127	...	–2	–1

Общее количество значений со знаком и без него одинаково, но их диапазоны сдвинуты друг относительно друга на числовой оси (рис. 4.8).

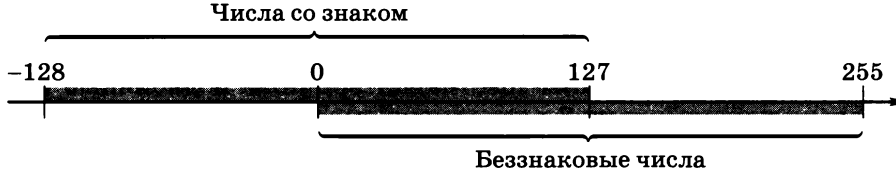


Рис. 4.8

В наших рассуждениях использовались 8-разрядные числа, но все выводы справедливы для чисел любой разрядности. От числа разрядов K зависят только граничные значения X_{max} и X_{min}, приведённые в табл. 4.5.

Таблица 4.5

K	8	16	32	64
X_{max}	127	32 767	2 147 483 647	9 223 372 036 854 775 807
X_{min}	−128	−32 768	−2 147 483 648	−9 223 372 036 854 775 808

Хотя дополнительный код гораздо менее нагляден, чем прямой, он значительно упрощает выполнение арифметических операций в компьютере. Например, вместо вычитания используется сложение с дополнительным кодом вычитаемого, поэтому не нужно проектировать специальное устройство для вычитания чисел.

Выводы

- Для хранения целого числа может быть использовано 8, 16, 32 или 64 бита памяти. Каждый дополнительный бит расширяет диапазон допустимых чисел в 2 раза.
- Отрицательные целые числа хранятся в дополнительном двоичном коде, который позволяет выполнять вычисления с положительными и отрицательными числами по одному и тому же алгоритму.
- Двоичный дополнительный код числа (−X) совпадает с двоичным кодом числа $2^K - X$, где K — число двоичных разрядов, отведённых для его хранения.

- Для получения дополнительного двоичного кода числа нужно выполнить инверсию всех старших битов числа, кроме последней (младшей) единицы и тех нулей, которые стоят после неё.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Чем отличается представление в компьютере целых чисел со знаком и без знака?
2. Приведите примеры величин, которые всегда имеют целые неотрицательные значения.
3. Как представлены в компьютере целые числа без знака?
4. Как изменится диапазон представления чисел, если увеличить количество разрядов на 1? На 2? На n?
5. Какое максимальное целое беззнаковое число можно записать с помощью K двоичных разрядов? Что произойдёт, если прибавить единицу к этому максимальному значению?
6. Как действует процессор при переполнении?
7. Почему максимальное положительное и минимальное отрицательное значения целых двоичных чисел со знаком имеют разные абсолютные значения?
8. Верно ли, что положительные числа кодируются одинаково в знаковом и беззнаковом форматах?

*9. Докажите, что алгоритмы А1, А2 и А3 всегда дают один и тот же результат.

10. Что получится, если правила перевода в дополнительный код применить к отрицательному числу?
11. Как можно проверить правильность перевода в дополнительный код?
12. Какое минимальное отрицательное значение можно записать с помощью K двоичных разрядов?

*13. Может ли быть переполнение при сложении двух отрицательных чисел? Какой знак будет у результата?

14. В чём главное преимущество дополнительного кода при кодировании отрицательных чисел?
15. Почему компьютер может обойтись без вычитания?

Проект

«Программа для перевода чисел в двоичный дополнительный код»

§ 26
Операции с целыми числами

Ключевые слова:

- знаковый бит
- дополнительный код
- переполнение
- поразрядные логические операции
- логический сдвиг
- арифметический сдвиг
- циклический сдвиг

Сложение и вычитание

Сложение и вычитание требуются не только для расчётов по формулам, но и для организации вычислений. Например, для того чтобы повторить какое-то действие R раз, используют переменную-счётчик, к которой после каждого выполнения этого действия прибавляют единицу, а затем результат сравнивают с R. Вместо этого можно сразу записать в счётчик значение R и после каждого повторения вычитать из него единицу, пока не получится ноль[1].

Благодаря тому что отрицательные числа кодируются в дополнительном коде, при сложении можно не обращать внимания на знаки слагаемых, т. е. *со знаковым разрядом обращаются точно так же, как и со всеми остальными.*

Например, сложим числа 5_{10} ($0000\,0101_2$) и -9_{10} ($1111\,0111_2$), используя 8-разрядную двоичную арифметику. Применим сложение столбиком, не задумываясь о знаках чисел:

$$\begin{array}{rr} & 0000\ 0101 \\ + & 1111\ 0111 \\ \hline & 1111\ 1100 \end{array}$$

Для расшифровки получившегося отрицательного числа применим к нему схему получения дополнительного кода: $1111\,1100 \to$ $\to 0000\ \ 0100_2 = 4_{10}$. Таким образом, результат равен -4_{10}, что совпадает с правилами «обычной» арифметики.

При сложении двух чисел с одинаковыми знаками может случиться переполнение — сумма будет содержать слишком большое количество разрядов. Покажем, как это выглядит для положительных и для отрицательных чисел.

---

[1] Второй вариант более эффективен, потому что процессор автоматически сравнивает результат очередного действия с нулём.

Сложим десятичные числа 96 и 33. Их сумма 129 выходит за 8-битную сетку. Для того чтобы обнаружить переполнение, добавим к обоим слагаемым ещё один старший бит, совпадающий со знаковым (рис. 4.9).

	0	0	1	1	0	0	0	0	0
+	0	0	0	1	0	0	0	0	1
	0	1	0	0	0	0	0	0	1
	S'	S							

Рис. 4.9

Знаковый разряд S результата равен 1, т. е. сумма получилась отрицательной, хотя оба слагаемых положительны! Процессор определяет переполнение, сравнивая биты S и S': если они различны, то произошло переполнение и результат неверный (см. рис. 4.9).

То же самое получается, если сложить два достаточно больших по модулю отрицательных числа, например –96 и –33. Добавим к кодам обоих чисел один старший разряд, равный знаковому разряду (рис. 4.10).

	1	1	0	1	0	0	0	0	0
+	1	1	1	0	1	1	1	1	1
	1	0	1	1	1	1	1	1	1
	S'	S							

Рис. 4.10

Получается, что в результате бит S равен нулю, хотя ответ должен быть отрицательным. Биты S' и S не совпадают, это говорит о том, что произошло переполнение. Несложно проверить (сделайте это самостоятельно), что, если переполнения нет, значения битов S и S' всегда одинаковы независимо от знаков слагаемых.

Сложение многоразрядных двоичных чисел в компьютере выполняет специальное устройство — **сумматор** (см. главу 3). Как мы уже говорили, вычитание сводится к сложению с дополнительным кодом вычитаемого, поэтому отдельного «блока вычитания» в компьютере нет.

Умножение и деление

Умножение и деление выполнять труднее, чем сложение и вычитание. Вспомните, например, что в математике умножение часто

заменяют многократным сложением, а деление — многократным вычитанием.

К двоичным числам можно применять обычную схему умножения «столбиком». Перемножим, например, числа 9_{10} ($0000\,1001_2$) и 5_{10} ($0000\,0101_2$):

$$\begin{array}{rr} & 00001001 \\ \times & 00000101 \\ \hline & 00001001 \\ + & 00000000 \\ & 00001001 \\ \hline & 0000101101 \end{array}$$

Легко проверить, что это число равно 45_{10}.

По сравнению с десятичной системой, здесь есть серьёзное упрощение: первый сомножитель умножается на единицу (в этом случае результат равен ему самому) или на ноль (результат — 0). Поэтому компьютерное умножение целых чисел состоит из следующих элементарных действий:

1) вычисление очередного произведения в зависимости от младшего бита второго сомножителя: оно равно нулю (если этот бит нулевой) или первому множителю (если бит равен единице);
2) сложение содержимого сумматора, накапливающего сумму, с очередным произведением;
3) сдвиг первого сомножителя влево на 1 разряд;
4) сдвиг второго сомножителя вправо на 1 разряд (при этом следующий бит попадет в младший разряд).

Таким образом, удаётся построить схему умножения без использования таблицы умножения. Заметим, что умножение — это довольно трудоёмкая операция, и для её ускорения конструкторы используют различные «хитрые» приёмы. Поэтому в реальных компьютерах всё может выглядеть значительно сложнее, чем в учебном примере.

Умножение, как и сложение, выполняется одинаково для положительных и отрицательных чисел (в дополнительном коде). Если в нашем примере вместо числа 9 подставить –9, то получится:

$$\begin{array}{rr} & 11110111 \\ \times & 00000101 \\ \hline & 11110111 \\ + & 00000000 \\ & 11110111 \\ \hline & 10011010011 \end{array}$$

Оставив только 8 младших бит, можно убедиться (применяя алгоритмы А1–А3), что результат — это дополнительный код числа –45.

Теория деления нацело оказывается намного сложнее, чем для умножения, поэтому мы её обсуждать не будем.

Сравнение

В отличие от арифметических действий операция сравнения *по-разному* выполняется для чисел со знаком и без него. Ещё раз внимательно посмотрим на таблицы кодов 8-битных чисел, приведённые в § 25. Если сравниваемые коды не превышают $7F_{16}$, то оба числа положительны и сравнение однозначно. Если это не так, то сравнение чисел с учетом и без учёта знака даёт разные результаты. Например, для беззнаковых чисел 81_{16} (129_{10}) больше, чем $7F_{16}$ (127_{10}). Для чисел со знаком, наоборот, отрицательное значение 81_{16} (-127_{10}) будет меньше, чем $7F_{16}$ (127_{10}). Поэтому современные процессоры имеют разные команды для сравнения чисел со знаком и без знака. Чтобы не путаться, в первом случае (при сравнении с учётом знака) обычно используют термины «больше»/«меньше», а во втором (без учёта знака) — «выше»/«ниже».

Поразрядные логические операции

В главе 3, изучая основы математической логики, мы увидели, что обработка истинности и ложности высказываний может быть представлена как набор операций с двоичными кодами. Оказывается, что логические операции, введённые первоначально для обработки логических данных, можно формально применить к битам двоичного числа, и такой подход широко используется в современных компьютерах.

Рассмотрим электронное устройство для управления гирляндой лампочек. Состояние каждой из них будет задаваться отдельным битом в некотором управляющем регистре: если бит равен нулю, лампочка выключена, если единице — включена. Для получения различных световых эффектов (типа «бегущих огней») требуется зажигать или гасить отдельные лампочки, менять их состояние на противоположное и т. д. (рис. 4.11). Точно так же биты регистров используются для управления внешними устройствами.

Будем применять логические операции *к каждому* биту числа, как обычно считая, что 1 соответствует значению «истина», а 0 — «ложь». Эти операции часто называют **поразрядными** или

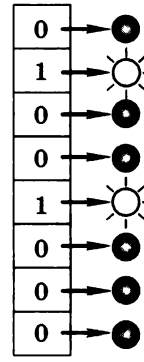


Рис. 4.11

битовыми, поскольку действия совершаются над каждым разрядом в отдельности, независимо друг от друга[1].

Введём несколько терминов, которые используются в литературе по вычислительной технике. **Сброс** — это запись в бит нулевого значения, а **установка** — запись единицы. Таким образом, если бит в результате какой-то операции становится равным нулю, то говорят, что он сбрасывается. Аналогично, когда в него записана единица, говорят, что бит установлен.

Маска — это константа (постоянная), которая определяет область применения логической операции к битам многоразрядного числа. С помощью маски можно скрывать (защищать) или открывать для выполнения операции отдельные биты[2].

Основные логические операции в современных процессорах — это НЕ (**not**), И (**and**), ИЛИ (**or**) и исключающее ИЛИ (**xor**).

Логическое НЕ (инвертирование, инверсия, **not**) — это замена всех битов числа на обратные значения: 0 на 1, а 1 — на 0. Эта операция используется, например, для получения дополнительного кода отрицательных чисел (см. алгоритм А1 в § 25). НЕ — это *унарная* операция, т. е. она действует на **все** биты *одного* числа. Маска здесь не используется.

Логическое И (and). Обозначим через D содержимое некоторого бита данных (англ. *data* — данные), а через M — значение соответствующего ему бита маски (англ. *mask* — маска). Операция И между ними задаётся таблицей, показанной на рис. 4.12.

---

1) Для сравнения, сложение (как и другие арифметические действия) не является поразрядной операцией, поскольку возможен перенос из младшего разряда в старший.

2) Использование маски аналогично выделению области рисунка в графическом редакторе — для выделенных пикселей маска равна 1, для остальных — нулю.

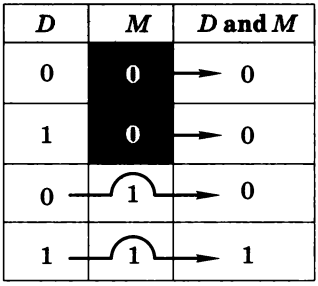


Рис. 4.12

Из таблицы видно, что при выполнении логического И нулевой бит в маске всегда сбрасывает (делает равным нулю) соответствующий бит результата, а единичный бит позволяет сохранить значение D (как бы пропускает его, открывая «окошко»).

С помощью логической операции И можно *сбросить* отдельные биты числа (те, для которых маска нулевая), не меняя значения остальных битов (для которых в маске стоят единицы).

Например, операция X **and** 1 сбросит у любого числа X все биты, кроме самого младшего. С помощью этого приёма легко узнать, является ли число чётным: остаток от деления на 2 равен последнему биту!

Логическое ИЛИ. Вспомнив таблицу истинности логической операции ИЛИ (**or**), можно обнаружить, что ноль в маске сохраняет бит (D **or** 0 = D), а единица — устанавливает соответствующий бит результата (D **or** 1 = 1) (рис. 4.13).

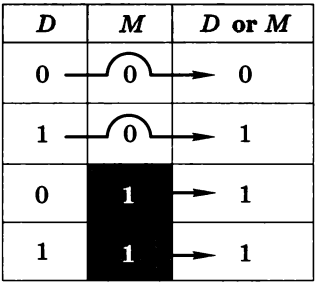


Рис. 4.13

С помощью логической операции ИЛИ можно *установить* отдельные биты числа (те, для которых маска единичная), не меняя значения остальных битов (для которых в маске стоят нули).

Например, операция X **or** 80_{16} установит старший бит восьмиразрядного числа X, тем самым формально сделав число отрицательным.

Таким образом, используя операции И и ИЛИ, можно сбрасывать и устанавливать любые биты числа, т. е. строить любой нужный двоичный код. Где это может пригодиться? Рассмотрим примеры решения конкретных задач.

Пример 1. На клавиатуре набраны 3 цифры, образующие значение целого числа без знака. Определить, какое число было введено.

При нажатии клавиши на клавиатуре в компьютер поступает код нажатой клавиши. Выпишем десятичные и шестнадцатеричные коды всех символов, обозначающих цифры:

Символ	'0'	'1'	'2'	'3'	'4'	'5'	'6'	'7'	'8'	'9'
X_{10}	48	49	50	51	52	53	54	55	56	57
X_{16}	30	31	32	33	34	35	36	37	38	39

Будем пользоваться шестнадцатеричными кодами: как видно из таблицы, их связь с цифрами числа гораздо нагляднее. Чтобы получить числовое значение цифры из кода символа X, достаточно сбросить его старшие 4 бита, не изменяя значение 4-х младших. Для этого нужно использовать операцию X **and** $0F_{16}$.

Пусть S_1 — код первого введённого символа, S_2 — второго, S_3 — третьего, а N обозначает искомое число. Тогда алгоритм перевода кодов символов в число выглядит так:

1. $N = 0$.
2. $W = S_1$ **and** $0F_{16}$ (выделяем первую цифру).
3. $N = 10 \cdot N + W$ (добавляем её к числу).
4. $W = S_2$ **and** $0F_{16}$ (выделяем вторую цифру).
5. $N = 10 \cdot N + W$ (добавляем её к числу).
6. $W = S_3$ **and** $0F_{16}$ (выделяем третью цифру).
7. $N = 10 \cdot N + W$ (добавляем её к числу).

Пусть, например, набраны символы '1', '2' и '3'. Тогда по таблице находим, что $S_1 = 31_{16}$, $S_2 = 32_{16}$ и $S_3 = 33_{16}$. Значение W на втором шаге вычисляется так:

$$\begin{array}{rr} & 0011\ \ 0001 \\ \textbf{and} & 0000\ \ 1111 \\ \hline & 0000\ \ 0001 \end{array}$$

Так как $W = 1$, на третьем шаге получаем $N = 1$. Следующая пара шагов — четвёртый и пятый, дают результаты $W = 2$ и $N = 12$ соответственно. Наконец, результат завершающих шагов — $W = 3$ и $N = 123$.

Такая процедура используется в каждом компьютере: именно так коды цифровых символов, набранные на клавиатуре, преобразуются в числа, с которыми компьютер выполняет арифметические действия.

Пример 2. Создадим структуру данных S, которая отражает, есть или нет в некотором числе каждая из цифр от 0 до 9. В математике такая структура называется *множеством*. Для хранения S будем использовать 16-разрядное целое число (рис. 4.14).

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Рис. 4.14

Договоримся, что младший бит числа имеет номер 0 и хранит информацию о том, есть во множестве цифра 0 (если этот бит равен 0, такой цифры нет, если равен 1, то есть). Аналогично первый бит (второй по счёту справа) показывает, есть ли во множестве цифра 1, и т. д. Старшие биты 10–15 при этом не используются. Например, во множестве, изображённом на рис. 4.14, есть только цифры 0, 3, 6 и 9.

Для записи элементов во множество и проверки их наличия удобно использовать логические операции. Рассмотрим для примера бит 5. Маска, которая потребуется для обращения к нему, — это единица в пятом разряде и нули во всех остальных, т. е. $M = 0020_{16}$. С её помощью можно добавить элемент ко множеству с помощью операции ИЛИ: $S = S$ **or** M. А узнать, есть ли во множестве интересующая нас цифра, можно, выделив соответствующий бит с помощью логического И ($P = S$ **and** M) и проверив результат на равенство нулю.

Исключающее ИЛИ. Как видно из таблицы истинности, операция исключающее ИЛИ (**xor**) не изменяет биты, когда маска нулевая, и меняет на противоположные при единичной маске (рис. 4.15).

Рис. 4.15

Например, команда $X = X$ **xor** FF_{16} выполняет инверсию всех битов 8-разрядного целого числа X. Напомним, что это один из этапов получения дополнительного кода отрицательных чисел.

С помощью логической операции исключающее ИЛИ можно выполнить *инверсию* отдельных битов числа (тех, для которых маска единичная), не меняя значения остальных битов (для которых в маске стоят нули).

Пример 3. Пусть X — это результат выполнения некоторого вычислительного теста, а Y — то, что ожидалось получить («правильное» значение). Нужно определить, в каких разрядах различаются эти числа (для инженера это очень полезная подсказка, где искать неисправность).

Предположим, что $X = 7$ и $Y = 3$. В результате операции X xor Y устанавливаются в единицу только те разряды, которые в этих числах не совпали, а остальные сбрасываются[1]. В данном случае находим, что числа отличаются только одним битом:

$$\begin{array}{rr} & 0000\ 0111 \\ \textbf{xor} & 0000\ 0011 \\ \hline & 0000\ 0100 \end{array}$$

Пример 4. Используя логическую операцию исключающее ИЛИ, можно шифровать любые данные. Покажем это на примере простого текста '2*2=4'.

Выберем любую маску, например $17_{16} = 0001\ 0111_2$. Эта маска представляет собой *ключ шифра* — зная ключ, можно расшифровать сообщение. Возьмём первый символ — цифру '2', которая имеет код $50_{10} = 0011\ 0010_2$, и применим исключающее ИЛИ с выбранной маской:

$$\begin{array}{rr} & 0011\ 0010 \\ \textbf{xor} & 0001\ 0111 \\ \hline & 0010\ 0101 \end{array}$$

Полученное значение $0010\ 0101_2 = 37_{10}$ — это код символа '%'. Для расшифровки применим к этому коду «исключающее ИЛИ» с той же маской:

[1] Профессиональные программисты часто используют операцию xor для обнуления переменной: команда `R:=R xor R;` в языке Паскаль запишет в переменную R ноль, независимо от её начального значения.

	0010 0101
xor	0001 0111
	0011 0010

В результате получили число 50 — код исходной цифры '2'.

Повторное применение операции исключающее ИЛИ с той же маской восстанавливает исходное значение, т. е. эта логическая операция *обратима*.

Если применить такую процедуру шифрования ко всем символам текста '2*2=4', то получится зашифрованный текст '%=%*#'.

Обратимость операции исключающее ИЛИ часто используется в компьютерной графике для временного наложения одного изображения на другое. Это может потребоваться, например, для выделения области с помощью инвертирования её цвета.

Сдвиги

Об операции сдвига вспоминают гораздо реже, чем она того заслуживает. Перечитайте ещё раз алгоритм умножения, описанный в § 26, и вы поймёте, что он весь построен на сдвигах. Сдвиги незаменимы тогда, когда требуется проделать ту или иную обработку *каждого* бита, входящего в число. Наконец, сдвиги двоичного числа позволяют быстро умножить или разделить число на степени двойки: 2, 4, 8 и т. д. Поэтому программисты очень ценят и широко применяют всевозможные разновидности сдвигов.

Идея операции сдвига довольно проста: все биты кода одновременно сдвигаются в соседние разряды[1] влево или вправо (рис. 4.16).

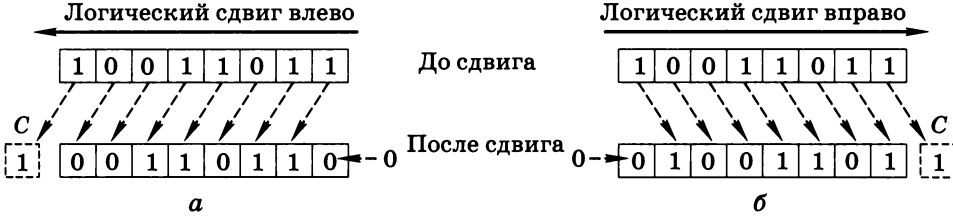


Рис. 4.16

[1] Аппаратно сдвиг реализуется необычайно просто и изящно: регистр, содержащий число, сбрасывается в ноль, при этом из тех разрядов, где исчезла единица, электрический импульс проходит в соседние и устанавливает их в единицу. При этом важно, что все разряды обрабатываются одновременно.

Отдельно надо поговорить о двух крайних битах, у которых «нет соседей». Для определённости обсудим сдвиг влево. Для самого младшего бита (на рис. 4.16, *а* он крайний справа) данные взять неоткуда, поэтому в него просто заносится ноль. Самый старший (крайний слева) бит должен потеряться, так как его некуда сохранить. Чтобы данные не пропали, содержимое этого разряда копируется в специальную ячейку процессора — бит переноса *C* (от англ. *carry* — перенос).

Рассмотренный тип сдвига обычно называется **логическим сдвигом**. Его можно использовать для быстрого умножения и деления. Рассмотрим, например, 8-разрядный двоичный код 0000 1100, который представляет число 12_{10}. Выполнив логический сдвиг влево, получим 0001 1000, т. е. число 24_{10}, которое вдвое больше! Это не случайность: вспомните, что происходит, если к десятичному числу справа приписать дополнительный ноль, например 34 → 340.

При сдвиге вправо любое чётное число уменьшается ровно в 2 раза. В случае нечётного значения происходит деление нацело, при котором остаток отбрасывается. Например, из 0001 0001 = $= 17_{10}$ при сдвиге вправо получается 0000 1000 = 8_{10}.

Логический сдвиг влево на 1 разряд увеличивает целое положительное двоичное число вдвое, а сдвиг вправо — делит на 2 нацело.

Пример. Для умножения числа, находящегося в ячейке *Z*, на 10 можно использовать такой алгоритм:

1. Сдвиг влево *Z* (в ячейке *Z* получаем $2Z_0$, где Z_0 — исходное число).
2. $X = Z$ (сохраняем Z_0).
3. Сдвиг на 2 бита влево *X* (вычисляем $8Z_0$).
4. $X = X + Z$ ($X = 8Z_0 + 2Z_0 = 10Z_0$).

Для некоторых компьютеров такая последовательность выполняется быстрее, чем стандартная операция умножения.

Посмотрим, что получится для отрицательных чисел. Сдвинем влево код 1111 1000 (8-разрядное представление числа –8): получится 1111 0000. Легко проверить, что это дополнительный код числа –16, т. е. значение удвоилось! Но сдвиг вправо работает по-другому: из 1111 1000 получаем 0111 1100 — это код положительного числа! Дело в том, что при сдвиге вправо отрицательных чисел, в отличие от положительных, старший разряд надо заполнять не нулем, а единицей! Чтобы исправить положе-

ние, вводится ещё одна разновидность сдвига — **арифметический сдвиг**. Его единственное отличие от логического состоит в том, что старший (знаковый) бит не меняется, т. е. знак числа остается прежним (рис. 4.17).

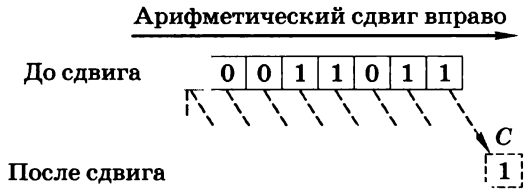


Рис. 4.17

Если применить арифметический сдвиг к коду 1111 1000, получается 1111 1100 — дополнительный код числа –4, т. е. произошло деление на 2. В качестве упражнения проверьте, как ведёт себя отрицательное нечётное число при арифметическом сдвиге вправо.

Арифметический сдвиг влево не требуется, поскольку он ничем не отличается от обычного логического сдвига.

То, что в результате логических сдвигов содержимое крайних разрядов теряется, не всегда удобно. Поэтому в компьютере предусмотрен **циклический сдвиг**, при котором бит из одного крайнего разряда переносится в другой («по циклу», рис. 4.18).

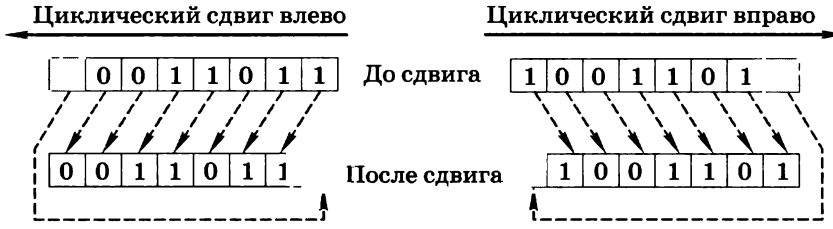


Рис. 4.18

Циклический сдвиг позволяет «просмотреть» все биты и вернуться к исходному значению. Если сделать последовательно 8 циклических сдвигов 8-битного числа, каждый его бит на каком-то шаге окажется на месте младшего разряда, где его можно выделить с помощью логической операции И с маской 1. Так можно «просматривать» не только младший, но и любой другой разряд (например, для выделения старшего разряда нужно использовать маску 80_{16}).

Выводы

- Операции с положительными и отрицательными числами выполняются в процессоре по одним и тем же алгоритмам.
- Переполнение — это ситуация, когда результат суммирования содержит слишком большое число разрядов и при переносе изменяется знаковый разряд числа.
- С помощью поразрядных логических операций можно управлять отдельными битами регистров процессора и внешних устройств.
- Логический сдвиг влево на 1 разряд увеличивает целое положительное число вдвое, а сдвиг вправо – делит на 2 нацело.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Покажите на примере, как складываются два положительных целых числа, записанные в 8-разрядные ячейки. Что изменится, если числа будут отрицательными?
2. При каких комбинациях знаков слагаемых в результате сложения может возникнуть переполнение?
3. Какое устройство выполняет в компьютере сложение? Вспомните, что вы о нём знаете.
4. Почему не нужно разрабатывать специальное устройство для вычитания целых чисел?
5. Перемножьте столбиком два положительных целых числа в двоичной системе счисления. Изменится ли алгоритм выполнения операции, если у одного из сомножителей поменять знак?
6. Почему коды чисел со знаком и без знака нужно сравнивать по-разному?
7. Почему арифметические операции нельзя отнести к поразрядным?
8. Как, используя маску, сбросить определённый бит (записать в него 0)?
9. Напишите значение маски для того, чтобы сбросить в 16-разрядном числе 2 младших бита, не изменяя все остальные. Какую логическую операцию нужно для этого использовать?
10. Напишите значение маски для того, чтобы установить в 16-разрядном числе 2 старших бита, не изменяя все остальные. Какую логическую операцию нужно для этого использовать?
11. Как, используя логические операции, определить, делится ли число на 4? На 8?

12. В каких практических задачах можно применять установку или сброс битов двоичного кода?
13. Каковы возможности операции исключающее ИЛИ?
*14. Попробуйте придумать алгоритм шифрования кода с помощью операции исключающее ИЛИ. Постарайтесь предложить простой алгоритм изменения маски, а не просто использовать константу.
15. Прочитайте ещё раз материал, связанный с переполнением при сложении. Какой логической операцией можно определить, совпадают или нет биты S' и S?
16. Какую роль играет операция НЕ при получении отрицательных чисел?
17. Как выполнить инверсию всех битов, не используя логическую операцию НЕ?
18. Как обрабатываются самый старший и самый младший биты при различных типах сдвига?
19. Покажите на примерах, что сдвиг влево двоичного кода удваивает число, а сдвиг вправо — уменьшает вдвое.
20. Почему логический сдвиг не годится для уменьшения в два раза отрицательных чисел? Как работает арифметический сдвиг?
*21. Выведите правило вычисления результата арифметического сдвига отрицательного нечётного числа на один разряд вправо. Проверьте, применимо ли это правило к положительным нечётным числам. Как упрощается формула для чётных исходных значений?

Проекты

а) Программа для выполнения поразрядных логических операций
б) Шифрование данных с помощью логических операций
в) Управление лампочками с помощью Arduino

§ 27
Хранение в памяти вещественных чисел

Ключевые слова:

- фиксированная запятая
- плавающая запятая
- значащая часть
- порядок числа
- скрытая единица

В начале главы мы отмечали принципиальное различие между вещественными и целыми числами: целые числа дискретны, а вещественные, напротив, непрерывны, а значит, не могут быть полностью корректно перенесены в дискретную по своей природе вычислительную машину. Как же всё-таки кодируются в компьютерах вещественные числа?

В первых ЭВМ использовалось кодирование **с фиксированной запятой**. Это значит, что положение запятой, отделяющей целую часть от дробной, было жёстко закреплено в разрядной сетке конкретной ЭВМ — раз и навсегда для всех чисел и для всех технических устройств этой машины. Все вычислительные алгоритмы были заранее «настроены» на это фиксированное размещение. Но в задачах, которые решаются на компьютерах, встречаются самые разнообразные по величине числа, от размера атома до астрономических расстояний. Чтобы согласовать их с таким жёстким представлением, программист, подготавливая задачу к решению на ЭВМ, выполнял большую предварительную работу по *масштабированию* данных: маленькие числа умножались на определённые коэффициенты, а большие, напротив, делились. Масштабы подбирались так, чтобы результаты всех операций, включая промежуточные, не выходили за пределы разрядной сетки и, в то же время обеспечивалась максимально возможная точность (все разряды данных по возможности находились в пределах сетки). Эта работа требовала много времени и часто являлась источником ошибок.

Тем не менее работа с фиксированным размещением запятой не только показала недостатки метода, но и наметила путь их устранения. В самом деле, если наиболее сложным и трудоёмким местом является масштабирование данных, надо его автоматизировать. Иными словами, надо научить машину самостоятельно размещать запятую так, чтобы числа при счёте не выходили за разрядную сетку и по возможности сохранялись с максимальной точностью. Конечно, для этого нужно разрешить компьютеру «перемещать» запятую, а значит, дополнительно как-то сохранять в двоичном коде числа информацию о её текущем положении. В этом и заключается главная идея представления чисел **с плавающей запятой**[1].

Удобное представление вещественных чисел не пришлось специально придумывать. В математике уже существовал подходя-

[1] В англоязычных странах используется термин *floating point* — плавающая точка, поскольку в этих странах традиционно целая часть отделяется от дробной не запятой, как у нас, а точкой.

щий способ записи, основанный на том, что любое число A в системе счисления с основанием B можно записать в виде

$$A = \pm Z \cdot B^P,$$

где Z называют **значащей частью**, а показатель степени P — **порядком** числа (рис. 4.19). Для десятичной системы это выглядит привычно, например заряд электрона равен $-1{,}6 \cdot 10^{-19}$ кулона, а скорость света в вакууме составляет $3 \cdot 10^8$ м/с.

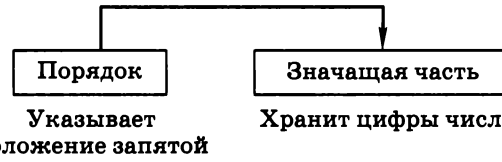


Рис. 4.19

Однако представление числа с плавающей запятой не единственно. Например, число 23,4 можно записать следующими способами:

$$2340 \cdot 10^{-2} = 234 \cdot 10^{-1} = 23{,}4 \cdot 10^0 = 2{,}34 \cdot 10^1 = 0{,}234 \cdot 10^2 = 0{,}0234 \cdot 10^3 = \ldots$$

На первый взгляд выбор очень широкий, однако большинство вариантов обладают серьезными недостатками. В частности, все представления, в которых значащая часть содержит нули непосредственно после запятой (0,0234, 0,00234 и т. п.) или перед ней (2340, 23400 и т. п.), не подходят, поскольку, сохраняя эти незначащие нули, мы напрасно увеличиваем разрядность чисел. Согласно математической теории, для обеспечения максимальной точности при сохранении цифр числа в фиксированном количестве разрядов надо выбирать такой метод, при котором значащие цифры числа следует поместить как можно ближе к запятой. С этой точки зрения оптимальным будет вариант, когда целая часть равна нулю, а первая ненулевая цифра находится сразу после запятой (в нашем примере 0,234). При этом вместо двух частей (целой и дробной) остаётся только дробная, что фактически делает ненужной «разделительную» запятую.

Но взгляните на рис. 4.20, *а*, изображающий такое число на индикаторе: первый разряд всегда равен нулю, что делает его практически бесполезным. Поэтому с точки зрения экономии разрядов лучше взять другой вариант, в котором значащая часть равна 2,34 (рис. 4.20, *б*). Именно такой выбор закреплён в стандарте IEEE 754[1)], на котором основана арифметика вещественных чисел в современных компьютерах.

[1)] Последняя версия стандарта называется IEEE 754-2008.

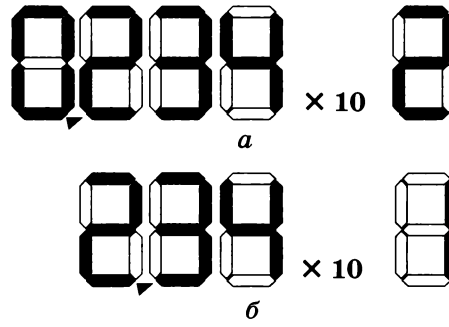


Рис. 4.20

Итак, существует два приблизительно равноценных способа представления чисел с плавающей запятой:

- оптимальный с теоретической точки зрения, в котором целая часть нулевая, а первая цифра дробной части ненулевая ($0{,}234 \cdot 10^2$);
- более удобный с практической точки зрения, в котором целая часть состоит из единственной ненулевой цифры ($2{,}34 \cdot 10^1$).

К сожалению, эта «двойственность» порождает некоторую путаницу. В теоретической литературе, как правило, используется первый способ[1]. Все описания конкретных компьютерных систем, напротив, базируются на втором. Причём в обоих случаях обычно используется один и тот же термин — **мантисса**. Зато в англоязычной компьютерной литературе приняты два разных термина: в первом случае значащая часть называется *mantissa* (слово «мантисса» для математиков однозначно связано с дробной частью числа), а во втором — *significand* (значащая часть).

Далее мы будем использовать второй вариант, поскольку именно он даёт возможность решать задачи, связанные с практическим кодированием вещественных чисел. В связи с этим мы будем применять термин «значащая часть», а не «мантисса».

В компьютере используется такое представление вещественных чисел с плавающей запятой, при котором значащая часть Z удовлетворяет условию $1 \le Z < B$, где B — основание системы счисления. Такое представление называется **нормализованным**.

Нормализованное представление числа единственно — в нашем примере это $2{,}34 \cdot 10^1$. Любое число может быть легко нормализовано. Единственное, но важное исключение из правила

[1] К этой группе относится большинство отечественных книг по основам вычислительной техники.

составляет нуль — для него невозможно получить $Z \geq 1$. Ради такого важного случая было введено дополнительное соглашение: число 0, в котором все биты нулевые, в качестве исключения считается нормализованным.

Всё сказанное выше можно применить и к двоичной системе:

$$A = \pm Z \cdot 2^P, \quad \text{причём } 1 \leq Z < 2.$$

Например: $-7_{10} = -111 \cdot 2^0 = -1{,}11 \cdot 2^{10}$ (не забывайте, что значащая часть и порядок записаны в двоичной системе!); отсюда $Z = 1{,}11$ и $P = 10$.

Двоичная значащая часть *всегда* (исключая, разумеется, ноль!) *начинается с единицы*, так как $1 \leq Z < 2$. Поэтому во многих компьютерах (в том числе в компьютерах на базе процессоров *Intel*), эта так называемая **скрытая единица** не хранится в ОЗУ, что позволяет сэкономить ещё один дополнительный разряд значащей части[1].

Идея «скрытой единицы» раньше действительно давала заметное увеличение точности представления чисел. Количество разрядов в устройствах ЭВМ того времени было невелико и поэтому усложнение метода кодирования было оправданно. Сейчас, когда процессоры работают с 64-битными данными, это скорее дань традиции, чем практически полезная мера[2].

Таким образом, при кодировании вещественного числа с плавающей запятой фактически хранится две величины: его значащая часть (*significand*) и порядок. От разрядности значащей части зависит точность вычислений, а от разрядности порядка — диапазон представления чисел. В таблице 4.6 приведены характеристики стандартных вещественных типов данных, используемых в математическом сопроцессоре *Intel*.

Таблица 4.6

Тип	Диапазон	Число десятичных значащих цифр	Размер (байт)
single	$1{,}4 \cdot 10^{-45}$ — $3{,}4 \cdot 10^{38}$	7–8	4
double	$4{,}9 \cdot 10^{-324}$ — $1{,}8 \cdot 10^{308}$	15–16	8
extended	$3{,}6 \cdot 10^{-4951}$ — $1{,}2 \cdot 10^{4932}$	19–20	10

Рассмотрим, как «распланированы» 4 байта, отводимые под простейший тип *single*. Тип *double* устроен аналогично, а тип

[1] В результате то, что осталось после «скрытия» единичной целой части, можно вполне обоснованно называть мантиссой.

[2] Оценим величину добавки для математического сопроцессора *Intel*. Значащая часть чисел двойной точности вместо «скрытой единицы» приобретает дополнительный 53-й (!) бит, что прибавляет к значению числа поправку $2^{-53} \approx 1{,}1 \cdot 10^{-16}$, влияющую на 16-й десятичный знак; согласно IEEE 754-2008, в 128-битных числах эта поправка будет и того меньше: $2^{-113} \approx 9{,}6 \cdot 10^{-35}$.

extended, который является основным форматом для вычислений в математическом сопроцессоре, отличается только тем, что в нём единица в целой части не «скрывается».

В данных типа *single* 23 младших бита (с номерами от 0 до 22) хранят значащую часть числа, следующие 8 (с 23 по 30) — порядок, а старший (31-й) бит отведён под знак числа (рис. 4.21).

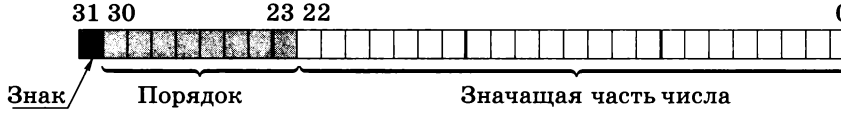


Рис. 4.21

Правила двоичного кодирования вещественных чисел во многом отличаются от правил кодирования целых чисел. Для того чтобы в них разобраться, рассмотрим конкретный пример — закодируем число –17,25 в формате *single*. Прежде всего, переведём модуль числа в двоичную систему, отдельно целую и дробную части (см. главу 2):

$$17{,}25 = 10001{,}01_2.$$

Для нормализации нужно передвинуть запятую на $4_{10} = 100_2$ разряда влево:

$$10001{,}01 \cdot 2^0 = 1{,}000101 \cdot 2^{100}.$$

Построим значащую часть, «скрыв» единицу в целой части:

$$M = Z - 1 = \mathbf{0{,}0001010...0}.$$

Так как число отрицательное, знаковый разряд нужно установить в 1, т. е. $S = 1$.

В отличие от целых чисел значащая часть вещественных чисел хранится в прямом коде.

Таким образом, значащие части положительного и равного по модулю отрицательного чисел одинаковы, а отличаются числа только старшим (знаковым) битом.

Теперь остается закодировать двоичный порядок 100. Порядок — это целое число со знаком, для него используется **кодирование со смещением**: чтобы вообще избавиться от знака порядка, к нему добавляют некоторое положительное смещение d:

$$P_d = P + d.$$

Величина смещения подбирается так, чтобы число P_d было всегда положительным. В этом случае оказывается легче скон-

струировать математический сопроцессор для обработки вещественных чисел.

Для кодирования порядка в данных типа *single* используют смещение $d = 127_{10} = 7F_{16}$. Таким образом, для нашего примера

$$P_d = 100 + 111\ 1111 = 1000\ 0011.$$

Собирая теперь S, P_d и M в единое 32-разрядное число, получаем:

$$1\ 10000011\ 00010100000000000000000$$

или более компактно в шестнадцатеричной системе — C1 8A 00 00. Этот код и будет записан в память[1)].

Не все двоичные комбинации для вещественных чисел соответствуют «правильным» числам: некоторые из них кодируют бесконечные значения, а некоторые — нечисловые данные (англ. NaN: *not a number* — «не число»). Они отличаются от остальных чисел тем, что имеют максимально возможный порядок[2)] (например, для типа *single* это смещённый порядок $P_d = 255$, а для типа *double* — 2047). Подобные «неправильные» данные возникают только в результате ошибок в вычислениях.

Таким образом, мы увидели, что целые и вещественные числа хранятся в памяти компьютера по-разному. Поэтому неудивительно, что свойства значений, скажем 3 и 3,0, в компьютерной арифметике совершенно различные.

Выводы

- Вещественные числа хранятся в памяти компьютера в формате с плавающей запятой: отдельно значащая часть и порядок.
- В компьютере используется такое представление вещественных чисел с плавающей запятой, при котором значащая часть Z удовлетворяет условию $1 \le Z < B$, где B — основание системы счисления. Такое представление называется *нормализованным*.
- В отличие от целых чисел значащая часть вещественных чисел хранится в прямом коде.
- Из-за ограниченности числа разрядов вещественное число, как правило, не удаётся точно представить в памяти.

Нарисуйте в тетради интеллект-карту этого параграфа.

---

1) На самом деле, в IBM-совместимых персональных компьютерах байты будут сохранены в памяти в обратном порядке: 00 00 8A C1.

2) Это вполне логично, поскольку для вещественных чисел переполнение (получение «бесконечного» значения) наступает именно при больших порядках.

Вопросы и задания

1. Чем вызваны трудности, возникающие при представлении вещественных чисел в компьютере? Как они связаны с непрерывностью вещественных чисел в математике?
2. Объясните, как хранятся вещественные числа с фиксированной запятой. Почему этот метод не используется в современных компьютерах?
3. Что такое плавающая запятая? Из каких частей состоит число при кодировании с плавающей запятой?
4. Приведите примеры физических величин, которые обычно записывают в форме с плавающей запятой.
5. Почему метод представления чисел с плавающей запятой неоднозначен? Как изменится порядок, если запятую сместить на один разряд влево (вправо)?
6. Что такое нормализованная форма записи числа?
7. Как требования нормализации связаны с точностью представления вещественных чисел?
8. Единственно ли нормализованное представление числа? Все ли числа имеют нормализованное представление?
9. Почему старший бит значащей части нормализованного двоичного числа всегда равен единице? Как этот факт используется на практике?
10. Какие числа сохраняются в памяти с нулевой значащей частью?
11. На что влияет разрядность значащей части и разрядность порядка?
12. Почему задание разрядности для целых чисел однозначно определяет их свойства, а для вещественных — нет?
13. Что вы знаете о типах *single*, *double* и *extended*?
14. Как хранится порядок во всех рассмотренных форматах вещественных чисел? Почему не хранится знак порядка?
15. Сравните методы хранения отрицательных целых и вещественных чисел.
16. Как по двоичному представлению вещественного числа определить, положительно оно или отрицательно? Подходит ли этот метод для целых чисел?
17. В каком из вещественных форматов не используется «скрытая единица» и почему?
18. Какие логические операции и с какой маской надо применить, чтобы в переменной типа *single*: а) выделить значащую часть, сбросив порядок и знаковый бит; б) восстановить в полученной знаковой части «скрытую единицу»?

19. Как можно выделить смещённый порядок из числа типа *single*? Как получить истинное значение порядка?
20. С помощью какой маски можно выделить знаковый бит числа, хранящегося в формате *single*?
21. Что такое NaN?
22. Чем различаются представление в памяти целого числа и равного ему вещественного с нулевой дробной частью (например, 12 и 12,0).

Проект

Программа для построения двоичных кодов вещественных чисел

§ 28
Операции с вещественными числами

Ключевые слова:

- значащая часть
- выравнивание порядков

Сложение и вычитание

Рассмотрим принципы вещественной компьютерной арифметики на простых примерах. Сложим $7{,}25_{10} = 111{,}01$ и $1{,}75_{10} = 1{,}11$ (здесь и далее будем явно указывать систему счисления только для десятичных чисел). Представим эти числа в нормализованном виде: $111{,}01 \cdot 2^0 = 1{,}1101 \cdot 2^{10}$ и $1{,}11 \cdot 2^0$ (ещё раз подчеркнём, что значащие части и порядки чисел указываются в двоичной системе!). Не будем сейчас использовать «скрытую» единицу: это нужно только при сохранении чисел в памяти, а при изучении арифметических действий удобнее иметь «развёрнутые» значения.

Числа, записанные в форме с плавающей запятой, нельзя непосредственно сложить. Дело в том, что, когда числа имеют разный порядок, их значащие части оказываются сдвинутыми друг относительно друга. Поэтому первое, что делает процессор перед сложением вещественных чисел, — **выравнивает их порядки** до большего. Число, имеющее меньший порядок p_2 (и значащую часть z_2), «подгоняется» к числу с большим порядком p_1 следующим образом.

1. Если $p_2 = p_1$, то порядки выровнены и преобразования закончены.
2. $p_2 = p_2 + 1$.
3. Сдвинуть значащую часть z_2 на один разряд вправо.
4. Перейти к шагу 1.

Для нашего примера разность порядков равна $10 - 0 = 10 = 2_{10}$, так что для выравнивания порядков значащую часть придётся сдвинуть дважды (порядок при этом увеличится на 2): $1{,}11 \cdot 2^0 = 0{,}0111 \cdot 2^{10}$. Подчеркнём, что ради проведения сложения нормализацию пришлось временно нарушить.

Теперь числа имеют одинаковый порядок и их **значащие части можно складывать:**

$$\begin{array}{rr} & 1{,}1101 \\ + & 0{,}0111 \\ \hline & 10{,}0100 \end{array}$$

Полный результат (с учётом порядка) равен $10{,}01 \cdot 2^{10}$ (убедитесь, что получившееся число равно 9_{10}). Но значащая часть результата больше 2, поэтому для записи числа в память его необходимо нормализовать: $10{,}01 \cdot 2^{10} = 1{,}001 \cdot 2^{11}$.

В этом примере мы нигде не учитывали ограниченность разрядной сетки, и для простоты специально взяли короткие числа. Как же обстоит дело в реальных вычислениях? При выравнивании порядков происходит сдвиг значащей части меньшего из чисел вправо, при этом её младшие (правые) разряды могут выйти за пределы разрядной сетки и будут отброшены. При сложении чисел с большой разностью порядков в результате таких сдвигов меньшее число может оказаться равным нулю. Например, представьте себе, что при 24-битной значащей части у слагаемых A и B разность порядков составляет, например, 26_{10}. В этом случае при выравнивании порядков произойдёт 26 сдвигов значащей части вправо, так что абсолютно все (!) её разряды исчезнут. В результате сложения окажется, что $A + B = A$, хотя $B \neq 0$ — это очередной (но далеко не единственный) пример погрешности компьютерных вычислений.

Умножение и деление

Числа, представленные в форме с плавающей запятой, «хорошо приспособлены» для выполнения умножения и деления. При **перемножении** достаточно (в полном соответствии с правилами математики) **перемножить** их **значащие части**, а **порядки сложить**. При **делении значащие части делятся**, а **порядки вычитаются**. Конечно, результат может оказаться ненормализованным, но это легко устраняется стандартной процедурой.

Рассмотрим, как выполняется умножение чисел $1{,}25_{10} = 1{,}01_2$ и $4{,}0_{10} = 100{,}0_2$. В нормализованном виде они запишутся как $1{,}01 \cdot 2^0$ и $100 \cdot 2^0 = 1{,}0 \cdot 2^{10}$. В этом примере значащие части можно перемножить устно: $1{,}01 \cdot 1{,}0 = 1{,}01$. Теперь сложим порядки: $0 + 10 = 10$. Таким образом, результат равен $1{,}01 \cdot 2^{10}$.

Он уже удовлетворяет требованиям нормализации, поэтому никаких дополнительных действий не требуется. Легко показать, что $1{,}01 \cdot 2^{10} = 101 \cdot 2^0 = 5_{10}$.

Знакомство с вещественной арифметикой убедительно показывает важную роль битовых операций, изученных в § 26. Для нормализации постоянно используются сдвиги; для выделения значащей части или порядка из общего кода числа обязательно потребуется логическая операция И, а для получения единого кода из порядка и значащей части можно использовать логическое ИЛИ. В обеих последних задачах также необходимы сдвиги.

Выводы

- Для сложения и вычитания вещественных чисел необходимо выравнивать их порядки.
- При вычислениях с вещественными числами накапливаются ошибки. Для повышения точности расчётов нужно стараться, если возможно, использовать только операции с целыми числами.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Почему перед сложением или вычитанием вещественных чисел требуется выравнивать порядки?
2. Какое число — большее или меньшее — подвергается сдвигу при выравнивании порядков? Почему?
3. Верно ли, что при выравнивании порядков значащая часть всегда сдвигается вправо?
4. Как вычислить количество сдвигов, которое потребуется произвести для выравнивания порядков?
5. Может ли оказаться так, что при выполнении операции сложения значащие части придётся не складывать, а вычитать?
6. Как изменится двоичный код вещественного числа, если это число умножить на 2? Сравните с тем, как меняется код целого числа при удвоении.
7. Почему в компьютерной арифметике возможны случаи, когда $A + B = A$ при $B \neq 0$? При каких условиях может так получиться?

*8. Может ли при вычитании быть переполнение? Антипереполнение?

9. Сформулируйте правила умножения и деления вещественных чисел.
*10. Верно ли, что когда для размещения результата умножения в значащей части не хватает разрядов — это переполнение? Когда возникает переполнение?
11. Может ли в результате арифметической операции нарушиться нормализация? Как в таких случаях нужно поступать?

ЭОР к главе 4 на сайте ФЦИОР (http://fcior.edu.ru)

- Дополнительный код числа. Алгоритм получения дополнительного кода отрицательного числа
- Число и его компьютерный код
- Числа с фиксированной и плавающей запятой

Практические работы к главе 4

Работа № 10 «Тренажёр "Лампанель"»
Работа № 11 «Операции с целыми числами»
Работа № 12 «Логические операции и сдвиги»

Глава 5
КАК УСТРОЕН КОМПЬЮТЕР

Компьютер — это универсальный 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 счёт этого во многих областях они заменили специализированные устройства.

Исторически существовали 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 объёмы данных.

В результате почти все современные компьютеры работают только с дискретной (цифровой) информацией. Поэтому в этой главе рассматривается только цифровая вычислительная техника.

---

1) Слово *hardware* означает металлические изделия, а применительно к компьютеру — его детали (платы, монитор и прочее «железо»). Термин *software* исключительно компьютерный, он возник из противопоставления слов *soft* (мягкий, гибкий, податливый) и *hard* (твёрдый, жёсткий, негнущийся). Это значит, что *software* гибко «подстраивает» *hardware* для решения разнообразных задач.

§ 29
Современные компьютерные системы

Ключевые слова:

- персональный компьютер
- моноблок
- промышленный компьютер
- ноутбук
- ультрабук
- планшетный компьютер
- смартфон
- встроенный компьютер
- параллельные вычисления
- суперкомпьютер
- кластер
- распределённая система
- грид-система
- облачные вычисления
- квантовый компьютер
- биокомпьютер

Сегодня вы ежедневно видите вокруг множество компьютеров, самых разнообразных по назначению, конструкции и внешнему виду. Все они относятся к четвёртому поколению **электронных вычислительных машин (ЭВМ)**, которые сейчас называют **компьютерами**.

Стационарные компьютеры

Настольные персональные компьютеры (рис. 5.1), состоящие из системного блока и подключённых к нему внешних устройств, появилис[illegible] в конце XX века.

Рис. 5.1

Многие современные компьютеры (в том числе и настольные!) изготавливаются как **моноблоки** (панельные компьютеры): в одном корпусе смонтированы все узлы компьютера, в том числе монитор (рис. 5.2, *а*). К моноблокам относятся также все планшетные компьютеры и смартфоны. Моноблокам с сенсорными

экранами (реагирующими на прикосновение) не нужны клавиатура и мышь, поэтому они часто используются в системах автоматизации (рис. 5.2, *б*).

а *б* *в*

Рис. 5.2

Промышленные компьютеры на предприятиях (рис. 5.2, *в*) применяют для управления сложной техникой, например в автоматизированных системах управления технологическими процессами (АСУ ТП). Они часто устанавливаются рядом с оборудованием и поэтому должны надёжно работать в неблагоприятных условиях: при вибрации, повышенной влажности, меняющейся температуре, пыли, перепадах давления.

Мобильные устройства

Отличительная черта XXI века — увеличение мобильности пользователей, которые хотят иметь возможность в любой момент посмотреть новости, прочитать сообщения электронной почты и ответить на них, обратиться к информационным системам и т. п.

Мобильным называется устройство, которое:

- находится постоянно с пользователем;
- готово к работе в любой момент;
- является персональным, обычно защищено паролем;
- может подключаться к компьютерным сетям.

В **портативных компьютерах** (рис. 5.3, *а*) — их называют **ноутбуками** (от английского слова *notebook* — тетрадь, блокнот) или **ультрабуками** — все устройства смонтированы в одном корпусе. По своим возможностям они сейчас практически не уступают настольным компьютерам.

Стремительно растёт популярность **планшетных компьютеров**, в которых для ввода данных нажимают пальцем на **сенсорный** экран (рис. 5.3, *б*).

а *б* *в*

Рис. 5.3

Приборы бытовой электроники всё больше и больше приближаются к компьютерам по своим возможностям. Например, **смартфоны** (рис. 5.3, *в*) выполняют функции и сотового телефона, и небольшого, но достаточно мощного компьютера. Уменьшение размеров компьютеров позволяет встраивать их в часы и даже в очки.

Встроенные компьютеры

Встроенные компьютеры (англ. *embedded computers*) — это компьютеры, которые входят в состав устройства, которым они управляют (рис. 5.4). В отличие от универсальных компьютеров они могут решать только заранее определённые задачи, программы для решения которых заранее введены в их память. Они должны иметь небольшие размеры и вес и потреблять как можно меньше электроэнергии.

Рис. 5.4

Сейчас встроенные компьютеры:

- обрабатывают звук и видео в реальном времени;
- управляют работой станков с числовым программным управлением (ЧПУ);
- управляют работой банкоматов и платёжных терминалов;
- управляют медицинскими приборами;
- управляют самолётами и судами и решают многие другие задачи.

В любом современном автомобиле установлено несколько цифровых управляющих устройств, и их программное обеспечение не уступает по сложности программному обеспечению настольных компьютеров.

Главный элемент встроенного компьютера — **микроконтроллер** — миниатюрный компьютер, в котором на одном кристалле кремния объединены процессор, оперативная и постоянная память, каналы ввода и вывода данных.

Встроенные компьютеры — основа систем управления роботами. На современных заводах используется огромное количество промышленных роботов — станков с ЧПУ. Такие станки обрабатывают детали по заложенной в них программе. Для того чтобы перестроить станок на изготовление другого типа деталей, достаточно просто заменить программу. Роботы используются на конвейерных линиях, изготавливающих микросхемы для компьютеров: процессоры, память и др. Нас окружают автоматизированные системы, которые тоже можно назвать роботами, например система управления движением поездов метро, система управления отоплением дома.

Параллельные вычисления

Важное направление в компьютерах четвёртого поколения — *параллельная* (одновременная) обработка данных.

Параллельные вычисления — это вычисления на многопроцессорных системах, при которых одновременно выполняются многие действия, необходимые для решения одной или нескольких задач.

Если решаемую задачу удаётся разбить на независимые друг от друга подзадачи, то их не обязательно делать друг за другом, а можно для экономии времени выполнять одновременно, поручив каждую задачу отдельному процессору. Более того, все современные процессоры — многоядерные, т. е. фактически в одном кристалле «упакованы» несколько процессоров. Это позволяет проводить параллельные вычисления даже на одном компьютере.

Различают два вида параллельных вычислений — по задачам и по данным. **Параллельное выполнение задач** означает, что каждый процессор выполняет свою собственную программу (задачу). Такой режим работы используется в многозадачных операционных системах и серверах. Разделить задачи между процессо-

рами (или ядрами одного процессора) легко, но при этом трудно обеспечить их равномерную загрузку.

При **параллельной работе с данными** несколько процессоров совместно решают одну задачу. Например, можно разделить обработку 1000 элементов массива между 10 процессорами, и тогда каждому придётся обработать всего по 100 чисел. В этом случае нагрузка на процессоры будет практически одинаковой. Но появляются другие проблемы, которые затрудняют параллельную реализацию алгоритмов:

- время выполнения программы увеличивается из-за передачи исходных данных и результатов вычислений между процессорами;
- часто одному процессору просто приходится простаивать, пока другой не закончит свою часть вычислений; например, нельзя получить окончательную сумму чисел, пока не готово хотя бы одно из слагаемых, вычисляемых другими процессорами;
- в параллельных алгоритмах появляются дополнительные действия, связанные с распределением и организацией вычислений;
- параллельный алгоритм, как правило, требует дополнительной памяти по сравнению с последовательным.

Существуют два подхода к организации параллельных вычислений. В первом из них все процессоры берут данные из общей памяти. В этом случае легко написать параллельную программу, но зато память, с которой одновременно работают все процессоры, сильно перегружается (на практике из-за этого невозможно использовать более 16–32 процессоров). Второй подход — объединение независимых процессоров, каждый из которых имеет свою память (такая память называется *распределённой*). Это напоминает компьютерную сеть из отдельных машин. Во время вычислений процессоры меньше мешают друг другу, зато необходимо специально организовывать передачу данных между ними.

Суперкомпьютеры

Мощные многопроцессорные компьютеры, в которых выполняется параллельная обработка данных, называют **суперкомпьютерами**. Все развитые страны ведут жёсткую 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;
- расчёт процессов горения и взрыва, а также моделирование других физических задач (обтекание летательных аппаратов, расчёт на прочность кузовов автомобилей);
- моделирование и прогнозирование ситуации в экономике;
- моделирование работы человеческого мозга;
- расчёты процессов нефте- и газодобычи, а также сейсморазведки недр;
- проектирование новых электронных устройств.

Приведём несколько примеров применения суперкомпьютеров. Исследователи фирмы *IBM* на протяжении десятилетий изучают деятельность мозга и пытаются моделировать её. В 2009 году появилось сообщение 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[1)] автомобилей удаётся сократить на треть.

Применение суперкомпьютеров для расчёта 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.

В 2009 году в МГУ им. М. И. Ломоносова введён в строй самый мощный российский суперкомпьютер «Ломоносов» (рис. 5.5[2)]) производительностью около 400 Тфлопс[3)]. В его состав входят 8892 многоядерных процессора (общее число ядер — 35 776). На момент запуска, «Ломоносов» занимал в мировом рейтинге суперкомпьютеров *Top500* двенадцатое место. В 2014 году запущен

---

1) Крэш-тесты — это тесты, исследующие поведение машин при сильном ударе о бетонное препятствие.

2) Фотография предоставлена компанией «Т-Платформы» (**www.t-platforms.ru**).

3) Флопс (англ. **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 количества информации; очевидно, что операции над вещественными числами намного сложнее и выполняются гораздо дольше, чем над целыми.

новый суперкомпьютер «Ломоносов-2», который работает в несколько раз быстрее своего предшественника.

Рис. 5.5

Распределённые вычисления

Вычислительные задачи, которые ставятся перед современными компьютерами, стали настолько сложны, что решить их за приемлемое время с помощью одного компьютера уже невозможно. Для получения нужных результатов учёным требуется обрабатывать петабайты данных, и даже их хранение представляет собой серьёзную проблему. Поэтому возникла идея объединить мощность многих компьютеров для решения особо сложных задач.

Первым шагом в этом направлении стало создание **кластеров** (англ. *cluster*). Так называют группу компьютеров, которые обычно расположены в одном или нескольких соседних зданиях и объединены в локальную сеть. Компьютеры в кластере работают совместно, выполняя общую задачу, и для пользователя кластер выглядит как одна мощная вычислительная система.

Как правило, все компьютеры в кластере строятся на одинаковой аппаратуре, используют одно и то же специальное программное обеспечение и решают одну задачу. Таким образом, кластер может в некоторых случаях заменить суперкомпьютер. Программное обеспечение для кластерных систем должно быть написано специально «под задачу», т. е. сразу решать произвольные задачи на кластере не получится.

Иногда кластеры собираются из обычных рабочих станций (англ. **COW**: *Cluster of Workstation*). Они относительно дёшевы и высокопроизводительны.

Развитие этой идеи привело к разработке распределённых вычислительных систем. Их также называют **грид-системами** (от англ. *grid* — сетка, решётка).

Распределённая система — это группа независимых компьютеров, которая для пользователей представляет собой единую вычислительную систему.

Компьютеры, объединённые в грид-систему, могут располагаться в любых точках земного шара. Для их связи используются не высокоскоростные локальные сети (как в кластерах), а сеть Интернет. Узлы распределённой системы обмениваются данными в виде сообщений, они могут строиться на любой аппаратуре и использовать различные операционные системы. В грид-систему могут входить как обычные персональные компьютеры, так и мощные кластеры. Любой узел в любой момент может отключиться или вновь подключиться к распределённой системе, при этом работа всей системы не нарушается. Фактически грид-система представляет собой виртуальный суперкомпьютер (его также называют мета-компьютером).

Для распределённых вычислений лучше всего подходят задачи переборного и поискового типа, где отдельные узлы практически не взаимодействуют друг с другом. Специальная программа на компьютере пользователя соединяется с управляющим узлом и получает от него очередную порцию работы. Выполнив задание, узел передаёт обратно отчёт о полученном результате (например, о том, что цель поиска достигнута). Таким образом, объединяя в грид-систему большое количество обычных персональных компьютеров, можно достичь вычислительной мощности мощнейших суперкомпьютеров.

Согласно исследованиям, 90% пользователей загружает свой процессор менее, чем на 40%. Поэтому каждый из нас может принять посильное участие в решении научных задач, добровольно подключив свой компьютер к системе распределённых вычислений в рамках одного из проектов, например с помощью системы *BOINC*[1] (англ. *Berkeley Open Infrastructure for Network Computing* — открытая инфраструктура для сетевых вычислений университета Беркли). Например, с помощью распределённых вычислений выполняется поиск новых больших простых чисел[2].

---

1) **https://boinc.berkeley.edu, http://www.boinc.ru**
2) **http://www.mersenne.org/primes/**

В мае 2016 года в распределённой сети BOINC было более 350 000 участников и около миллиона компьютеров общей производительностью 11,6 петафлопс.

Облачные вычисления

Сейчас мы используем всё более сложные программы, которые требуют для своей работы всё более мощных компьютеров. Нам нужно обрабатывать всё бóльшие объёмы данных, и эти данные нужно где-то хранить, поэтому мы приобретаем всё более дорогие компьютеры. Но, как правило, такие сложные задачи нужно решать довольно редко, и гигабайты данных нам тоже нужны не каждый день. Поэтому появилась идея временно арендовать нужные ресурсы, используя для решения сложных задач мощности серверов в Интернете.

Облачные вычисления (англ. *cloud computing*) — технология обработки данных, при которой компьютерные ресурсы предоставляются пользователю как интернет-сервис.

Термины «облако» и «облачные вычисления» впервые использовал в одном из своих докладов глава компании Google Эрик Шмидт в 2006 году. После его выступления эта модель стала активно развиваться.

Слово «облако» используется как образ сложной системы, за которым скрываются все технические детали. Пользователь имеет доступ к своим данным, но не должен заботиться об аппаратном и программном обеспечении системы, с которой он работает. Все данные постоянно хранятся на серверах в Интернете и для обработки временно копируются на компьютер пользователя (игровую приставку, ноутбук, смартфон и т. д.).

Сегодня практически все ведущие компьютерные компании (в том числе Google, Microsoft, Amazon, IBM) предлагают свои облачные технологии. Пользователь получает вычислительные мощности и программное обеспечение как услугу, которую должен обеспечить провайдер.

Существует три основные модели облачных вычислений:

- **программное обеспечение как услуга** (англ. **SaaS**: *Software as a Service*), пользователь арендует программное обеспечение провайдера, которое работает в облаке;
- **платформа как услуга** (англ. **PaaS**: *Platform as a Service*), пользователь получает возможность разрабатывать или развёртывать готовые программы с помощью инструментов и языков программирования, которые поддерживает провайдер;

- **инфраструктура как услуга** (англ. **IaaS**: *Infrastructure as a Service*), пользователь получает все средства для установки и выполнения любых нужных ему программ, включая операционные системы.

Использование облачных технологий даёт значительные **преимущества** как для компаний, так и для обычных пользователей:

- данные в облаке доступны из любой точки, где есть доступ в Интернет;
- данные надёжно хранятся в центрах обработки данных (ЦОД) крупных компаний;
- доступны большие вычислительные мощности для хранения и обработки данных;
- уменьшаются затраты — не нужно покупать дорогостоящие компьютеры и нанимать специалиста по обслуживанию локальных сетей; оплата начисляется только за фактически использованные ресурсы.

Однако у облачных технологий есть и **недостатки**. В первую очередь они связаны с тем, что ваши данные попадают в руки разработчика облачного программного обеспечения, который теоретически может воспользоваться ими, как захочет. Нередко пользователь не может полностью удалить с серверов свои данные, и они могут храниться годами на дисках провайдера.

Перспективы развития компьютеров

В конце XX века 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. произойдёт 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 четвёртого поколения удалось реализовать лишь отдельные детали проекта, причём 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 четвёртого поколения. Классификация поколений «замерла» в ожидании новых революционных идей.

Такие идеи особенно необходимы ещё и потому, что увеличивать быстродействие процессоров бесконечно не получится из-за ограничений, связанных с действием законов физики.

Для того чтобы обрабатывать данные, части процессора должны обмениваться сигналами. Скорость передачи электрических сигналов огромна, но не бесконечна — она не может быть больше, чем скорость света в вакууме. Поэтому единственный способ снизить время передачи данных — уменьшать размеры элементов процессора. Но при этом появляются другие проблемы:

- чем мельче детали, тем сложнее их изготовить с нужной точностью;
- при обмене сигналами электронные схемы сильно нагреваются и при перегреве могут перестать работать. Чем мельче элементы, тем труднее охладить процессор;
- если «провода», проводящие электрический ток, расположить слишком близко, между ними может произойти короткое замыкание и вся схема выйдет из строя.

Сейчас для увеличения быстродействия применяют многоядерные процессоры. Однако этот подход тоже не решает всех проблем.

- Далеко не все задачи можно разделить на части, которые могут выполняться одновременно (параллельно).
- Обмен данными между процессорами происходит медленнее, чем внутри одного процессора. Из-за этих потерь времени вся экономия от параллельных вычислений может быть «съедена», так что никакого увеличения скорости работы не получится.

Поэтому постоянно идёт поиск неэлектронных средств хранения и обработки данных. В первую очередь учёные попытались использовать в качестве носителя информации свет — так появились **оптические процессоры**. В них можно применять параллельную обработку данных, например одновременно выполнять какую-то операцию со всеми пикселями изображения. В 2003 году был выпущен оптический процессор *Enlight256*, 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 фотоприёмников. 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 определённому алгоритму. Конечное состояние системы кубитов — это и есть ответ в задаче. Особые свойства кубитов[1)] 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 ведётся 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 планируют построить транзистор размером с молекулу.

Выбор конфигурации компьютера

Конфигурация компьютера — это набор аппаратного и программного обеспечения, установленного на компьютере.

В первую очередь нужно определить, какой именно компьютер вам нужен — настольный, ноутбук, планшетный или, может быть, смартфон.

Дальнейший выбор конфигурации зависит от решаемых задач, которые определяют необходимое программное обеспечение. У каждой программы есть минимальные требования, которые необходимо выполнить для её нормальной работы. В список этих требований обычно входят:

- тактовая частота процессора (например, не ниже 2 ГГц);
- объём оперативной памяти (например, не менее 2 Гбайт);
- объём свободной внешней памяти (например, не менее 15 Гбайт на жёстком диске или флэш-накопителе);

[1)] В отличие от привычного нам бита кубит устроен так, что способен вместить в себя гораздо больше данных.

- наличие и характеристики дополнительных устройств: дисководов DVD, сетевой карты, звуковой карты, микрофона, наушников и др.;
- характеристики внешних устройств (например, разрешение и время отклика монитора);
- используемая операционная система (например, не ниже *Windows 8.1*).

Быстродействие компьютера зависит не только от процессора, но и во многом от объёма оперативной памяти. Если в ОЗУ не хватает места для данных, операционная система использует «файл подкачки» на дисках, и из-за этого работа программ существенно замедляется. Поэтому среди профессионалов популярна фраза «памяти много не бывает».

Подбирая самостоятельно части компьютера, нужно проверять их совместимость. Например, для любого процессора нужно специально выбирать материнскую плату с подходящим разъёмом для его крепления. На сайтах компьютерных фирм можно найти *конфигураторы* — программы для выбора конфигурации компьютера с учётом совместимости различных блоков.

Компьютер желательно выбирать с учётом дальнейшего усовершенствования («апгрейда», от англ. *upgrade*) — добавления или замены отдельных устройств для расширения его возможностей, установки нового программного обеспечения.

С точки зрения пользователя обычно выделяют несколько типов бытовых компьютеров.

Офисный компьютер предназначен, главным образом, для подготовки документов, работы с электронной почтой и Интернетом. Такие компьютеры закупаются массово, поэтому главные критерии выбора — надёжность и низкая цена. Как правило, им не нужны быстродействующие процессоры и жёсткие диски большого объёма. Для офисного компьютера нужен принтер и сканер, которые можно объединить в одном устройстве (МФУ), возможно, сетевом (доступном с нескольких компьютеров по локальной сети).

Домашний компьютер должен быть более универсальным, чем офисный, поскольку используется для решения самых разных задач: подготовки документов и учебных заданий (рефератов, электронных таблиц), проигрывания звука и видео, поиска информации в Интернете, игр и т. д. Для этого требуются бóльшие мощности: быстродействие процессора, объём оперативной памяти, размер жёсткого диска. Из внешних устройств такому компьютеру нужны принтер и сканер (или МФУ), для работы с музыкой — качественная звуковая карта.

Игровые компьютеры — одни из самых требовательных к быстродействию процессора и видеокарты и размеру оперативной памяти. Жёсткий диск должен быть достаточно объёмный, потому что современные игры занимают много места. Для создания «эффекта присутствия» нужна хорошая звуковая карта. Во многих сетевых играх участники действуют в командах и общаются между собой, поэтому необходимы наушники и микрофон. Руль и джойстик помогут управлять в играх, моделирующих автогонки, полёты на самолётах и т. п.

Компьютер для профессиональной работы с графикой и видео (рабочее место дизайнера) должен иметь мощный многоядерный процессор, большой объём оперативной памяти, вместительный жёсткий диск (или флэш-накопитель), DVD-дисковод. При обработке изображений большого размера и видео высокой чёткости выполняется огромный объём вычислений, для сохранения вспомогательных данных программа использует жёсткий диск. Для ввода рисунков нужен сканер с высоким разрешением и графический планшет, для просмотра изображений и видео — монитор большого размера (с диагональю 21 дюйм и более) и хорошей цветопередачей; для печати — цветной принтер.

В задачах 3D-моделирования много ресурсов компьютера требует *рендеринг* (построение двумерных картинок трёхмерных сцен с учётом источников освещения и свойств материалов) и ещё больше — трёхмерная анимация. Иногда для этих целей используют видеокарты с несколькими графическими процессорами.

В таблице 5.1 приведены примерные характеристики компьютеров разных типов по версии одной из компьютерных фирм (2016 год).

Таблица 5.1

Тип компьютера	Процессор	ОЗУ	Жёсткий диск	Видеокарта (память)
Офисный	3,1 ГГц; 2 ядра	2 Гбайт	500 Гбайт	Интегрированная
Домашний	3,4 ГГц; 2 ядра	8 Гбайт	1 Тбайт	Выделенная (2 Гбайт)
Игровой	3,7 ГГц; 4 ядра	16 Гбайт	3 Тбайт	Выделенная (4 Гбайт)
Мульти-медийный	3,4 ГГц; 4 ядра	16 Гбайт	2 Тбайт	Выделенная (2 Гбайт)

Выводы

- Моноблок — это компьютер, все узлы которого смонтированы в одном корпусе.
- Встроенный компьютер — это компьютер, который входит в состав устройства, которым он управляет.
- Микроконтроллер — это миниатюрный компьютер, в котором на одном кристалле кремния объединены процессор, оперативная и постоянная память, каналы ввода и вывода данных.
- Параллельные вычисления — это вычисления на многопроцессорных системах, при которых одновременно выполняются многие действия, необходимые для решения одной или нескольких задач.
- Кластер — это группа компьютеров, которые связаны скоростными линиями связи в локальной сети и решают одну задачу под управлением специального программного обеспечения.
- Распределённая система (грид-система) — это группа независимых компьютеров, которая для пользователей представляет собой единую вычислительную систему.
- Облачные вычисления — технология обработки данных, при которой компьютерные ресурсы предоставляются пользователю как интернет-сервис.
- Конфигурация компьютера — это набор аппаратного и программного обеспечения, установленного на компьютере. Выбор конфигурации определяется задачами, которые решаются на компьютере.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Почему универсальный компьютер с изменяемой программой удобнее, чем специализированная техника? Ответ обоснуйте.
2. Почему цифровая техника вытеснила аналоговую?
3. Назовите бытовые приборы, в которых применяются микропроцессоры.
4. Найдите в Интернете рейтинг суперкомпьютеров *Top500*. Какие страны занимают в нем лидирующее положение? Есть ли там российские компьютеры?

*5. Зачем в суперкомпьютерах так много процессоров? Подумайте, любая ли задача может быть решена быстрее, если её выполнять параллельно на множестве процессоров. (В качестве помощи можно воспользоваться аналогией с распределением частей одного большого задания между учениками класса.)

*6. Почему, по вашему мнению, уже довольно давно не происходило смены поколений компьютеров?

7. Найдите дополнительный материал о разрабатываемых в лабораториях принципиально новых типах компьютеров.

8. Что вы можете сказать по поводу роли программного обеспечения: уменьшается она или увеличивается по сравнению с предыдущими поколениями?

9. Предположим, что появился процессор с каким-то принципиально новым свойством. Как быстро этим свойством смогут воспользоваться потребители? Какова роль программного обеспечения в этом?

10. Быстродействие вычислительной техники постоянно растёт. Как же тогда объяснить, что пользователи жалуются на «медлительные» компьютеры и всё время стараются купить новые, ещё более производительные?

*11. Влияет ли развитие программных средств на развитие аппаратной части?

12. Насколько сейчас, по-вашему, актуально умение программировать? Попробуйте найти аргументы «за» и «против» (учитывайте разные цели работы на компьютере).

Подготовьте сообщение

а) «Физические ограничения быстродействия компьютеров»

б) «Квантовые компьютеры»

в) «Распределённые вычисления»

г) «Грид-системы»

Проекты

а) Перспективные компьютеры

б) Сравнение вариантов конфигурации компьютеров

в) Сравнение облачных сервисов

г) Параллельная обработка данных

§ 30
Принципы устройства компьютеров

Ключевые слова:

- фон-неймановская архитектура
- процессор
- память
- устройства ввода
- устройства вывода
- двоичное кодирование
- принцип адресности
- принцип хранимой программы
- принцип программного управления
- архитектура
- ARM
- однокристальная система

Дж. фон Нейман (1903–1957)

Классические принципы построения ЭВМ были предложены в работе *А. Беркса, Г. Голдстайна* и *Д. фон Неймана* «*Предварительное рассмотрение логической конструкции электронного вычислительного устройства*», написанной в 1946 году по результатам опыта создания первой ЭВМ «ЭНИАК». 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 её единоличным автором.

Обычно выделяют[1)] 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.

---

1) Эта техническая работа не содержит отдельного пронумерованного перечня принципов, поэтому в учебной литературе встречаются непринципиальные отличия в их формулировке и описании.

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;
- **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 (рис. 5.6).

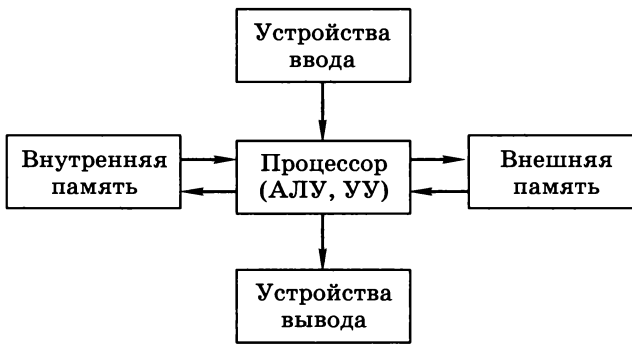


Рис. 5.6

Принцип двоичного кодирования. Устройства для хранения двоичной информации и методы её обработки наиболее просты и дёшевы. Поскольку в ЭВМ используется двоичная система счисления, 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, графику, звук, видео), но это не привело к отмене принципа двоичного кодирования. Даже цифровые сигнальные процессоры[1)], 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 на основе полупроводниковых технологий. Эти проблемы так и не были решены, тогда как наладить массовое производство аналогичных устройств для двоичных компьютеров оказалось значительно проще.

Принципы организации памяти

Принцип адресности памяти. Оперативная память машины состоит из отдельных битов. Для записи или считывания группы битов объединяются 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 объём 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.

[1)] 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 расчётов. Числа желательно было представлять как можно точнее, поэтому ячейки оперативной памяти в первых машинах были длинными. Кроме чисел машина должна была хранить в памяти ещё 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 серьёзному 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 5.7, *а* показана организация ячеек памяти первых ЭВМ, а на рисунке 5.7, *б* — современная (байтовая) структура памяти.

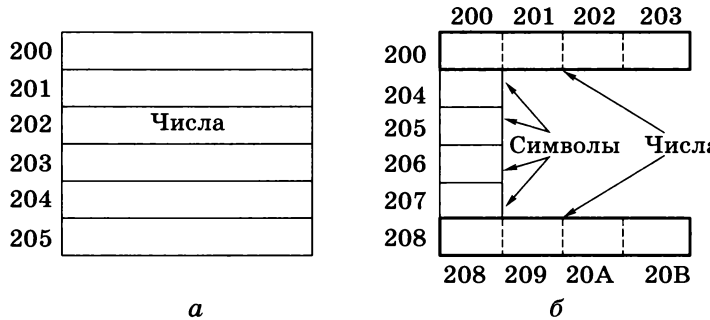


Рис. 5.7

На рисунке 5.7, *а* числа занимают по одной ячейке, причём номера этих ячеек отличаются на единицу. На рисунке 5.7, *б* показаны 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, 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 путём последовательного чтения.

Часто термин RAM отождествляют с русским термином **ОЗУ — оперативное запоминающее устройство**. Это не совсем точно. Дело в том, что кроме ОЗУ существует ещё 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 объёму, но это очень важная часть компьютера, поскольку в нём хранится доступное в любой момент программное обеспечение. Благодаря этому ПО компьютер может работать 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 предъявляются два противоречивых требования: её объём должен быть как можно больше, а скорость работы — как можно выше. Ни одно реальное устройство не может удовлетворить им одновременно. Любое существенное увеличение объёма памяти неизбежно приводит к уменьшению скорости её работы. Действительно, если память большая, то обязательно усложняется поиск в ней требуемых данных[1)], 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 объёмом памяти и её 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 техни-

[1)] Например, память большого объёма 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.

ки подтвердило необходимость построения иерархической памяти: в современном компьютере уровней иерархии гораздо больше.

Выполнение программы

Принцип хранимой программы. Первые ЭВМ программировались путем установки перемычек на специальных панелях, так что процесс подготовки к решению задачи мог растянуться на несколько дней. На рисунке 5.8 изображён фрагмент коммутационной панели устройства IBM-557; требуемая операция получается соединением (коммутацией) соответствующих отверстий. Такое положение дел никого не устраивало, и в фон-неймановской архитектуре было предложено представлять команды в виде двоичного кода. Код программы, записанный заранее[1] на перфокарты или магнитную ленту, можно было ввести в машину достаточно быстро.

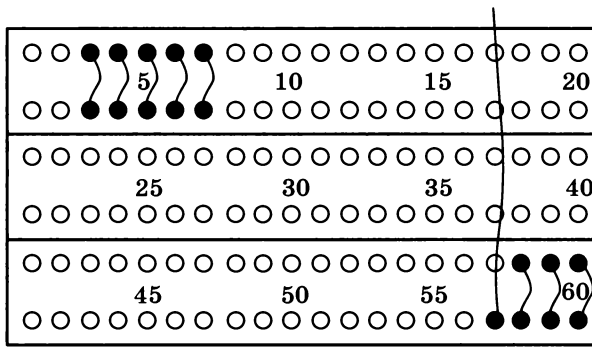


Рис. 5.8

Поскольку команды программы и данные по форме представления стали одинаковыми, их можно хранить в *единой памяти*[2] 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 программы. Именно так текст программы на языке высокого уровня переводится (транслируется) в машинные коды конкретной машины.

---

1) До появления персональных компьютеров для этого использовались специальные устройства *подготовки данных*. Такая схема ускоряла процесс ввода и исключала простои ЭВМ, связанные с длительным набором программ.

2) Известна также так называемая **гарвардская архитектура**, в которой программы и данные хранятся в разных областях памяти. Несмотря на повышение надёжности, она не получила широкого распространения.

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.

Важным элементом устройства управления в машине фон-неймановской архитектуры является специальный регистр — **счётчик адреса команд**[1]. В нём в любой момент хранится адрес команды программы, которая будет выполнена следующей.

Процессор выполняет команды по следующему алгоритму (его часто называют *основным алгоритмом работы процессора*):

1) из ячейки памяти, адрес которой записан в счётчике адреса команд, выбирается очередная команда программы; на время выполнения она сохраняется в специальном **регистре команд**;
2) значение счётчика 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. Чтобы использовать в программе ветвления и циклы, необходимо нарушить естественную последовательность выполнения команд. Для этого существуют специальные **команды перехода**, которые на этапе 3 заносят в счётчик адреса новое значение — адрес перехода. Чаще всего в программах используется **условный переход**, т. е. переход происходит только при выполнении определённого условия.

Легко понять, что для запуска основного алгоритма работы процессора в счётчик адреса команд должно быть предварительно занесено начальное значение — адрес первой выполняемой команды. В первых ЭВМ оператор вводил этот адрес вручную.

---

[1] В различных процессорах этот регистр может называться по-разному: например, в семействе *Intel* он обозначается **IP**: *Instruction Pointer* (указатель на инструкцию).

В современных компьютерах при включении питания в счётчик 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 передаётся дальнейшее управление, а стартовая программа из ПЗУ завершает свою работу. Начиная с этого момента, поведение компьютера уже определяется установленным на нём программным обеспечением.

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 передаёт 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 придётся обращаться к ОЗУ, так как она уже находится во внутренней памяти микропроцессора. Учитывая, что обращение к ОЗУ занимает значительно больше 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 удаётся выполнить, поскольку она использует результат предыдущей, или сразу нескольким командам потребуется одновременно обратиться к ОЗУ. Тем 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.

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.) формулиру-

ются 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;
- 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 жёсткого диска, ёмкость памяти, тактовую частоту). 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 неё 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!

Особенности мобильных компьютеров

Все современные мобильные устройства — это компьютеры. Рассмотрим для примера устройство смартфона.

Основные части мобильного телефона — это процессор, память, приёмо-передатчик сигналов, контроллеры для дисплея и клавиатуры, аккумулятор и SIM-карта (англ. *Subscriber Identification Module* — идентификационный модуль абонента). В SIM-карте находится собственный процессор: именно благодаря этому вы можете сохранять в карте записи телефонной книги.

Многие смартфоны для связи с другими устройствами имеют инфракрасный порт и адаптер *Bluetooth*. Сейчас технология *Bluetooth* активно используется для беспроводных гарнитур с наушниками и микрофоном.

В отличие от стационарных компьютеров и ноутбуков для мобильных телефонов нужно обеспечить:

- небольшие размеры и вес;
- поддержку специальных функций (например, приём и передачу речевых сигналов);
- экономию заряда аккумулятора.

Поэтому их аппаратное и программное обеспечение тоже особенное.

Большинство мобильных устройств строится на микросхемах с архитектурой ARM (англ. *Advanced RISC Machine* — усовершенствованная RISC-машина[1]). Компания ARM сама не производит процессоры, предпочитая продавать это право другим. Среди множества компаний, обладающих лицензиями, такие известные производители микросхем, как *Samsung*, *Apple*, *NVIDIA* и другие.

Современные процессоры ARM — 64-битные. Хотя по скорости вычислений они значительно уступают процессорам для стационарных компьютеров (особенно в операциях с вещественными числами), их энергопотребление значительно ниже. Если мобильное устройство не используется, включаются специальные экономичные режимы работы процессора. Например, для уменьшения расхода электроэнергии процессор существенно снижает свою тактовую частоту.

Микросхемы архитектуры ARM правильнее называть не процессорами, а однокристальными системами или «системами на чипе» (от англ. **SoC**: *System on a Chip* — система на кристалле). Кроме вычислительных ядер они содержат оперативную память, модули беспроводной связи и т. д. Процессоры ARM позволяют дополнительно подключить до 15 сопроцессоров (дополнительных процессоров), каждый из которых может выполнять свои задачи, например обработку изображений. Для эффективной обработки звука во многих ARM-процессорах есть специальные команды, которые обычно используются в специализированных цифровых сигнальных процессорах.

Выводы

- Основные устройства компьютера — процессор, память, устройства ввода и устройства вывода.
- В современных компьютерах используется двоичное кодирование данных.
- Память разделена на байтовые (восьмибитные) ячейки, адрес ячейки — это её номер. Нумерация ячеек памяти начинается с нуля.
- Невозможно создать память, которая имела бы большой объём и высокое быстродействие.
- Программы и данные хранятся в единой памяти.

---

[1] RISC-процессоры (англ. **RISC**: *Reduced Instruction Set Computer*) — это процессоры с сокращённым набором команд (см. § 32).

- Архитектура — это общие принципы построения конкретного семейства компьютеров.
- Большинство смартфонов строится на однокристальных системах с архитектурой ARM, которые отличаются низким потреблением электроэнергии.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Назовите основные компоненты вычислительного устройства. Согласны ли вы с тем, что полученный набор узлов логичен и обоснован?
*2. Используя дополнительные источники, выясните, какие научные достижения связаны с именем Джона фон Неймана.
*3. Найдите материалы о троичной ЭВМ «Сетунь». Сравните двоичные и троичные ЭВМ. В чём достоинства и недостатки каждого типа?
4. По какому алгоритму вводимые в компьютер десятичные числа можно перевести во внутреннее двоичное представление? Как перевести обратно результаты расчёта?
5. Почему появилась байтовая память?
6. Можно ли заменить в ячейке памяти содержимое одного бита, не затрагивая значений соседних? Почему?
7. С какой целью в компьютерных системах используют иерархическую организацию памяти?
8. Почему большая по объёму память обычно работает медленнее, чем маленькая?
9. Какие преимущества даёт принцип хранимой программы?
10. Можно ли выполнять арифметические операции с нечисловыми данными (символами, графическими и звуковыми данными)?
11. Как вы понимаете фразу «любая обработка данных в вычислительной машине происходит по программе»? Чем компьютер в этом отношении отличается от простого калькулятора?
12. Что такое счётчик адреса команд и какова его роль в основном алгоритме работы процессора?
13. Опишите, что происходит в момент включения компьютера с точки зрения принципа программного управления.
14. Можно ли нарушить последовательность выполнения команд в программе? Для чего это может потребоваться?
15. Всегда ли в новом компьютере есть какая-либо программа?
16. Что такое конвейер и как он работает при выполнении программы? Почему команды перехода нарушают работу конвейера?

17. Какие из принципов, предложенных фон Нейманом и его соавторами, продолжают применяться в современных компьютерах без всяких изменений, а какие принципы сохранились, но в несколько изменённом виде? Объясните, почему потребовались эти изменения.
18. Что такое архитектура? Какие детали устройства компьютера к ней не относятся?
19. В чём преимущества единой архитектуры семейств ЭВМ для пользователей и для производителей?

Подготовьте сообщение

а) «Архитектура ARM»
б) «Гарвардская архитектура»
в) «Конвейерный метод в обработке данных»

Проекты

а) Сравнение архитектур мобильных компьютеров
б) Платы Arduino
в) Использование одноплатных компьютеров Raspberry Pi
г) «Умный дом»

§ 31 Магистрально-модульная организация компьютера

Ключевые слова:

- шина
- контро́ллер
- принцип открытой архитектуры
- программно-управляемый обмен данными
- обмен по прерываниям
- прямой доступ к памяти

Что понимается под устройством 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-совместимых персональных компьютеров;
- различные конструкции компьютеров (настольные компьютеры, портативные компьютеры, смартфоны);
- *функциональное устройство* компьютера, т. е. его основные узлы и способы взаимодействия между ними.

Каждый из этих подходов полезен при решении определённых 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 подключённых к нему внешних устройств. Такая конструкция удобна для пользователя, поскольку все устройства можно разместить на столе так, как ему хочется.

В мобильных компьютерах (ноутбуках, планшетных компьютерах, смартфонах) весь минимально необходимый набор устройств собран в одном корпусе. Вместе с тем мощные серверы и суперкомпьютеры по-прежнему собираются в виде крупных «шкафов», напоминающих ЭВМ предыдущих поколений. Наконец, нельзя не упомянуть и о бытовой электронике, которая всё больше и больше приближается к традиционным компьютерам.

Разнообразие типов современных компьютеров говорит о том, что конструкция — это не самое главное. А вот что действительно практически не изменилось за многие годы, так это их *функциональное устройство*. Поэтому далее мы подробно рассмотрим основные узлы компьютера (процессор, память и устройства ввода и вывода) и взаимодействие между ними.

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 её по очереди. Такой информационный канал называется *шиной*.

Шина (магистраль) — это группа линий связи для обмена данными между несколькими устройствами компьютера.

Традиционно шина делится на три части (рис. 5.9) — это:

- шина данных, по которой передаются данные;
- шина адреса, определяющая, куда именно передаётся информация;
- шина управления, которая организует процесс обмена (несёт сигналы чтение/запись, обращение к внутренней/внешней памяти, данные готовы/не готовы и т. п.).

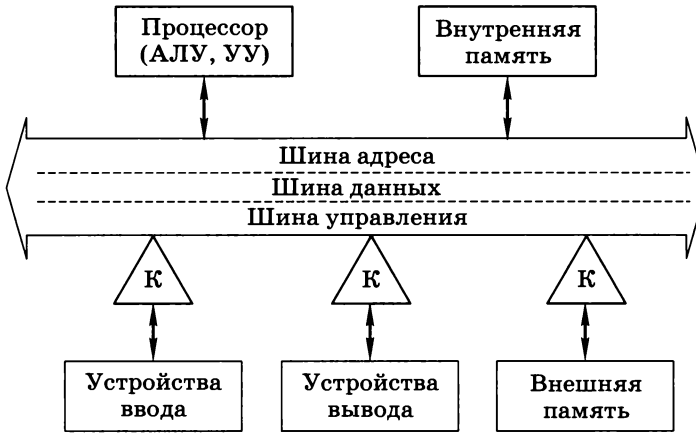


Рис. 5.9

Рассмотрим процесс записи данных из процессора в память. На шину данных процессор выставляет данные для записи, на шину адреса — нужный адрес памяти, а на шину управления — сигналы для записи данных в память. Далее он вынужден ожидать, пока данные будут «взяты» с шины. В это время все остальные устройства постоянно «слушают» шину (проверяют её состояние). В нашем примере по сигналам на шине память обнаруживает, что для неё имеются данные. Она сохраняет их по заданному адресу и должна по шине управления сообщить процессору, что операция завершена. На практике, учитывая высокую надёжность работы памяти, сигнал подтверждения часто не используется: процессор просто выжидает определённое время и продолжает выполнение программы. Из этого примера понятно, что для успешного обмена данными по шине необходимо ввести чёткие 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 поколении такие устройства назывались **каналами ввода/вывода**, а в четвёртом — контро́ллерами[1)] (на рис. 5.9 они обозначены буквой «К»).

Контроллер — это электронная схема для управления внешним устройством и простейшей предварительной обработки данных.

Современный контроллер — это специальный микропроцессор, предназначенный для обслуживания одного или нескольких однотипных устройства 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 жёсткого 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 удаётся!) и способен «на ходу» подменять их резервными, что создаёт видимость диска, который полностью свободен от дефектов.

Как видно из схемы на рис. 5.9, 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 счёт подключения к шине новых устройств, часто называют **магистрально-модульной**.

[1)] Это название происходит от английского слова *control* — управление; не следует путать с русским словом «контролёр».

Если спецификация на шину (детальное описание всех её логических и физических параметров) опубликована и общедоступна, то производители могут разрабатывать к такой шине любые дополнительные устройства. Такой подход называют **принципом открытой архитектуры**. При этом в компьютере предусмотрены стандартные разъёмы 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 управляет обменом данными между этим устройством и процессором.

В современных компьютерах для повышения эффективности работы используется несколько шин, например, одна — между процессором и памятью, другая связывает процессор с видеосистемой и т. д.

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 временны́е 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 раз, когда была нажата или отпущена клавиша; всё остальное время процессор выполняет программу, «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 идёт 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 своё решение. Затем он продолжает совещание, как ни в чём 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 учётом 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*, *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**: ***Direct Memory Access***).

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 передаёт 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 сведён в таблицу (табл. 5.2, здесь УВВ обозначает устройство ввода/вывода).

Таблица 5.2

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 идёт аппаратно)

Выводы

- Шина (магистраль) — это группа линий связи для обмена данными между несколькими устройствами компьютера. Шина делится на три части: шину данных, шину адреса и шину управления.
- Контро́ллер — это электронная схема для управления внешним устройством и простейшей предварительной обработки данных.
- Существуют три режима обмена данными между центральным процессором (ЦП) и внешними устройствами: 1) программно управляемый ввод/вывод; 2) обмен с устройствами по прерываниям; 3) прямой доступ к памяти (ПДП).

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы думаете, что более полезно для глубокого понимания работы компьютера: изучение функционального устройства компьютера или его конструкции? Обсудите этот вопрос в классе.
2. Как устройства компьютера обмениваются данными?
3. Почему обмен данными между устройствами компьютера с помощью шины оказался наилучшим решением?
4. Из каких частей состоит шина? Охарактеризуйте каждую из них.
5. Что такое магистрально-модульная архитектура и в чём её главное достоинство?
6. В чём заключается принцип открытой архитектуры?

7. Используя приведённое в тексте описание процесса записи данных в память, попробуйте объяснить, как происходит считывание данных из ячейки памяти с заданным адресом.
8. Объясните, как использование контроллеров позволяет повысить быстродействие компьютера в целом.
9. Сравните магистрально-модульную архитектуру компьютера с классической. Выделите наиболее перегруженный блок на каждой из схем.
10. Почему в современном компьютере используют несколько шин?
11. Что требуется для успешного подключения к компьютеру нового устройства?
12. Расскажите о разных режимах обмена данными с внешними устройствами.
13. Как расшифровывается сокращение ПДП и что это такое? Как выполняется обмен данными в режиме ПДП?
14. Какой режим лучше всего подходит для обмена данными с жёстким диском? С клавиатурой?
15. Где в программировании применяются принципы обработки прерываний?

Подготовьте сообщение

а) «Принцип открытой архитектуры»
б) «Что такое прерывания?»
в) «Зачем нужны контроллеры?»
г) «Прямой доступ к памяти»

§ 32
Процессор

Ключевые слова:

- процессор
- микропроцессор
- арифметико-логическое устройство
- устройство управления
- математический сопроцессор
- микрокоманда
- генератор тактовых импульсов
- такт
- регистр общего назначения
- тактовая частота
- разрядность
- CISC-процессор
- RISC-процессор
- операнд
- код операции

Центральным устройством, во многом определяющим возможности компьютера, является процессор (рис. 5.10 — вид микропроцессора со стороны выводов).

Рис. 5.10

Процессор — это устройство, предназначенное для автоматического считывания команд программы и их выполнения.

Название «процессор» происходит от английского глагола *to process* — обрабатывать. Иными словами, процессор — это блок компьютера, который автоматически обрабатывает данные по заданной программе.[1)]

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;
- **устройство управления (УУ)**, которое управляет выполнением программы и обеспечивает согласованную работу всех узлов компьютера.

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.

---

1) Не следует путать процессор как аппаратный блок с мощными программами обработки данных, которые также часто называют процессорами (например, текстовый процессор или табличный процессор).

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 используется 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 нулём) и отрицательность результата (определяется проверкой знакового разряда — см. главу 4). Результаты этого анализа заносятся в определённые 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.

Устройство управления

Главная задача устройства управления — обеспечить автоматическое выполнение последовательности команд программы в соответствии с основным алгоритмом работы процессора (см. § 30). Устройство управления выполняет следующие действия:

- извлечение из памяти очередной команды;
- анализ кода команды, определение необходимых действий и их распределение между узлами процессора;
- 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 команды запускается с помощью управляющего импульса (рис. 5.11). Опорную последовательность импульсов для этих целей УУ получает от **генератора тактовых импульсов**. Интервал между двумя соседними импульсами называется **тактом**.

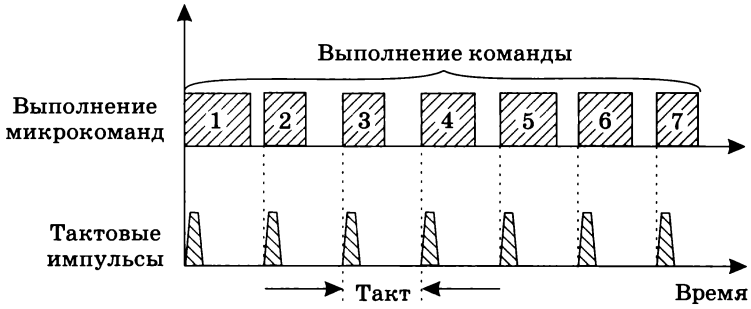


Рис. 5.11

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.

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)**, подчёркивая 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 расположенные рядом ячейки памяти, то к содержимому такого регистра нужно каждый раз прибавлять размер одной ячейки в байтах.

Количество регистров, а также их устройство в разных процессорах различается. 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 матрёшку (рис. 5.12).

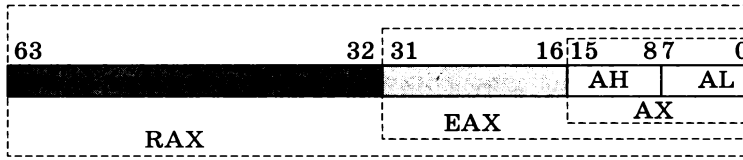


Рис. 5.12

На рисунке 5.12 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 AL и AH. Отчётливо 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* есть 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.

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 слу-

чае, 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.

Приближённо можно считать, что процессор выполняет за один такт одну простую команду (типа пересылки числа из регистра в регистр). Тогда при тактовой частоте 4 ГГц за одну секунду 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 объём памяти, который способен поддерживать процессор. Этот объём 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.

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 сдвига;
- 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 команды, входящие в систему команд процессора, 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 регистров или ячеек памяти, константы) называют **операндами**.

Рассмотрим для примера одну из наиболее простых команд процессора Intel, которая состоит из четырёх байт и имеет шестнадцатеричный код 81 C2 01 01. Она может быть разбита на три неодинаковые по длине части:

- код операции 81C обозначает сложение содержимого регистра с константой;
- первый операнд 2 — 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.

Выводы

- Процессор — это устройство, предназначенное для автоматического считывания команд программы и их выполнения.
- Две обязательные части процессора: арифметико-логическое устройство (АЛУ), выполняющее обработку данных, и

устройство управления (УУ), которое управляет выполнением программы и обеспечивает согласованную работу всех узлов компьютера.

- Выполнение каждой машинной команды состоит из элементарных действий, которые называются микрокомандами. Очередная микрокоманда начинает выполняться тогда, когда приходит тактовый импульс.
- Тактовая частота — это количество тактовых импульсов за одну секунду.
- Регистры общего назначения предназначены для использования программным обеспечением.
- Разрядность — это максимальное количество двоичных разрядов, которые процессор способен обработать за одну команду.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Какую роль в работе процессора выполняют АЛУ и УУ?
2. Почему удобно, что АЛУ автоматически сравнивает результат действия с нулём?
3. Подумайте, как с помощью логических операций с битами сумматора установить факт его равенства или неравенства нулю.
4. Для чего служит математический сопроцессор?
5. Зачем нужен генератор тактовых импульсов?
6. Что такое РОН? Для каких целей он может использоваться?
7. Найдите информацию о регистрах процессора *Intel*. Постарайтесь разобраться в назначении наиболее важных из них.
8. Объясните, как тактовая частота влияет на быстродействие компьютера.
9. Тактовые частоты двух процессоров, изготовленных фирмами *Intel* и *AMD*, равны. Означает ли это, что их быстродействие одинаково?
10. Объясните, как применение конвейера влияет на количество команд, выполняемых за один такт.
11. На что влияет разрядность процессора? Какие разновидности разрядности вы знаете? Что характеризует каждая из них?
12. Какие группы операций входят в систему команд любого процессора?
13. Сравните RISC- и CISC-процессоры. В чём состоят достоинства и недостатки каждого типа?
14. Какие части можно выделить в командах процессора?

Подготовьте сообщение

а) «CISC- и RISC-процессоры»
б) «Процессоры архитектуры ARM»
в) «Процессоры с малым энергопотреблением»
г) «Многоядерные процессоры»

Проекты

а) Сравнение систем команд процессоров
б) Сравнение процессоров Intel и AMD

§ 33
Память

Ключевые слова:

- память
- внутренняя память
- внешняя память
- машинный носитель информации
- флэш-память
- кэширование
- виртуальная память
- время доступа
- средняя скорость передачи данных

Как мы уже знаем, процессор способен выполнять программу, но её 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 сохра-

няется. 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 сохранить данные на длительный срок, пока они не потребуются, именно поэтому её ещё часто называют **долговременной**.

Внутренняя память

Внутренняя память — часть памяти компьютера, которая используется для хранения программ и данных во время решения задачи.

Часто её называют *основной памятью*. В состав внутренней памяти входят ОЗУ и ПЗУ.

Внутренняя память строится в соответствии с базовыми принципами, описанными ранее в § 30. 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 своё состояние даже после отключения питания. Однако при повторном включении (или перезагрузке) компьютера программное обеспечение не способно восстановить, где и какая информация находилась «в прошлый раз». Именно поэтому, если при наборе текста перезагрузить компьютер, работу придётся повторять заново.

Внутренняя 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 (рис. 5.13 — магнитное ОЗУ: слева на рисунке — биты памяти, справа — устройство выборки нужного адреса). Намагниченное состояние сердечника соответствовало единичному значению бита, ненамагниченное — нулевому. Заметим, что данные в магнитных ячейках памяти полностью сохранялись и после выключения питания. Наконец, развитие микроэлектроники позволило изготовить компактную полупроводниковую память (рис. 5.14 — мо-

дуль полупроводниковой памяти), которая сейчас и применяется в персональных компьютерах.

Рис. 5.13

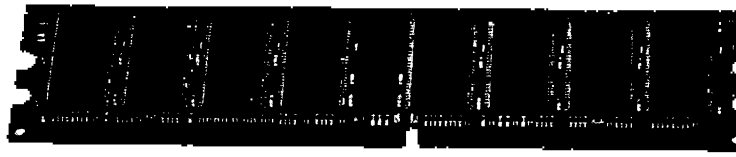


Рис. 5.14

Существуют **два типа оперативной памяти**, отличающиеся по технологии изготовления, — **статическая** и **динамическая** память. Первая строится *на триггерах* (об устройстве триггера см. 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 бóльшую ёмкость и меньшую стоимость, чем статическая. К сожалению, у неё 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 токопроводящие перемычки. Наличие перемычки означало единицу. Программатор мощными импульсами тока пережигал нужные перемычки, тем

самым устанавливая биты в нулевое состояние[1]. Очевидно, что процесс записи информации таким способом был необратимым.

Позднее появились **перепрограммируемые ПЗУ**, в которых очистка памяти выполнялась сначала ультрафиолетовыми лучами, а затем — электрическими импульсами. 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.

Рис. 5.15

Как правило, компьютер содержит микросхему ПЗУ (рис. 5.15), в которой записано встроенное программное обеспечение — набор программ, выполняющих проверку аппаратуры, начальную загрузку компьютера и обмен данными с некоторыми устройствами (клавиатурой, монитором, дисками). В компьютерах семейства IBM PC такое программное обеспечение называется **BIOS** (англ. *Basic Input/Output System* — базовая система ввода/вывода).

В IBM-совместимых компьютерах есть ещё 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**), причём

---

[1] Так сгорают плавкие предохранители в бытовой аппаратуре.

работать с ней пользователь может только до загрузки операционной системы (при включении компьютера).

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 съёмным (как в накопителях на оптических дисках), а может быть помещён внутрь неразборного устройства (жёсткий магнитный диск — «винчестер»).

В переносных устройствах внешней памяти, например во внешних жёстких дисках и флэш-накопителях, носитель и схема управления объединены в единый блок. Такие устройства подключаются к компьютеру снаружи через разъём.

Центральный процессор не может непосредственно обращаться к данным на носителе, он работает с ними через контроллер внешней памяти. На рисунке 5.16 схематично показано, как читаются данные с внешнего носителя информации в ОЗУ[1)].

Для связи с контроллером процессор использует **порты** — регистры контроллера, к которым процессор может обратиться по номеру. Процессор передаёт контроллеру «задание» на передачу данных, и контроллер берёт 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.

[1)] В действительности процесс обмена более сложен, в нём участвует ещё и контроллер ПДП.

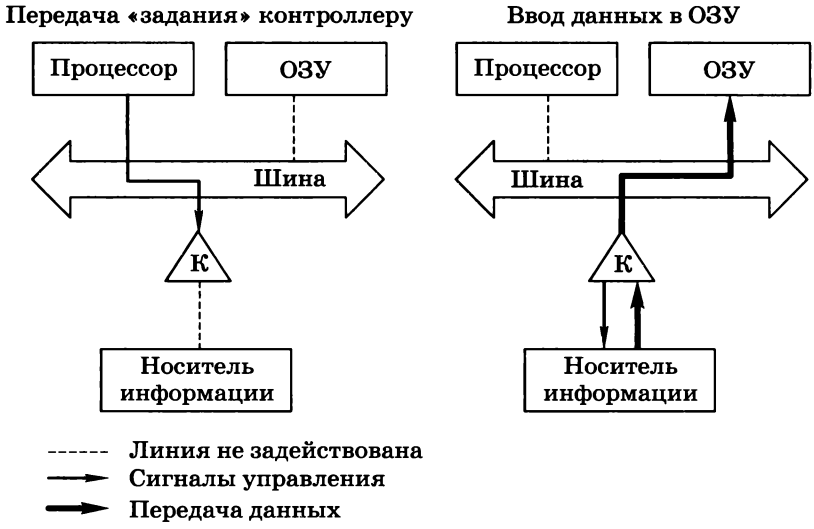


Рис. 5.16

Для внешней памяти характерны следующие черты:

- 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 невозможно;
- прежде чем процессор сможет непосредственно использовать программу или данные, хранящиеся во внешней памяти, их нужно предварительно загрузить в ОЗУ;
- обменом данными управляют контроллеры.

В качестве внешней памяти используются самые разные носители. Первоначально программы и данные сохранялись на бумажных **перфолентах** и **перфокартах** (рис. 5.17). Подписанные обычной ручкой или карандашом, они сортировались программистами вручную.

Перфолента

Перфокарта

Рис. 5.17

Затем произошёл переход к **магнитным носителям**: магнитным лентам, барабанам и дискам (рис. 5.18).

Рис. 5.18

На **магнитных дисках** биты данных хранятся в виде небольших намагниченных (или ненамагниченных) областей. **Секторы** размещаются на концентрических окружностях (имеющих общий центр), которые называются **дорожками**. Поскольку длина дорожки зависит от положения на диске, количество секторов на дорожках может быть разным. Доступ к секторам диска — произвольный, максимальная скорость достигается тогда, когда читаемые или записываемые секторы располагаются подряд.

Управление такой сложной системой очень трудоёмко, поэтому появление магнитных дисков привело к созданию специального ПО для работы с ними — операционных систем (ОС). ОС берёт 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 её от внутренней.

Следующей технологией хранения данных 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 информация хранится на отдельных замкнутых дорожках, данные на оптическом диске записываются вдоль непрерывной спирали, как на старых грампластинках[1)].

Сейчас широко используются оптические диски следующих поколений: **DVD** (англ. *Digital Versatile Disk* — цифровой многоцелевой диск, ёмкость до 17 Гбайт) и **Blu-ray-диски** (ёмкостью

[1)] В отличие от грампластинок спираль раскручивается от центра к краям.

до 500 Гбайт). Они имеют тот же диаметр, что и CD-диски, но для повышения плотности записи в дисководах DVD используют лазер с меньшей длиной волны.

Отметим, что на всех видах дисков есть разметка на секторы, 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 секторы диска как можно более равномерно.

Кроме широко распространённых флэш-накопителей («флэшек»), этот вид памяти используется в картах памяти (рис. 5.19) для фотоаппаратов, плееров и сотовых телефонов, а также в твёрдотельных накопителях (англ. **SSD:** *Solid State Drive*). Напомним, что ПЗУ также может изготовляться на базе флэш-памяти.

Рис. 5.19

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 § 30, невозможно создать память, которая имела бы как большой объём, так и высокое быстродействие. Поэтому используют многоуровневую (иерархическую) систему из нескольких типов памяти. Как правило, чем больший объём имеет память, тем медленнее она работает.

Самая быстрая (и очень небольшая) память — это регистры процессора. Гораздо больше по объёму, но заметно медленнее, внутренняя память (ОЗУ и ПЗУ). Далее следует огромная, но ещё более медленная внешняя память. Наконец, последний уровень — это данные, которые можно получить из компьютерных сетей (рис. 5.20).

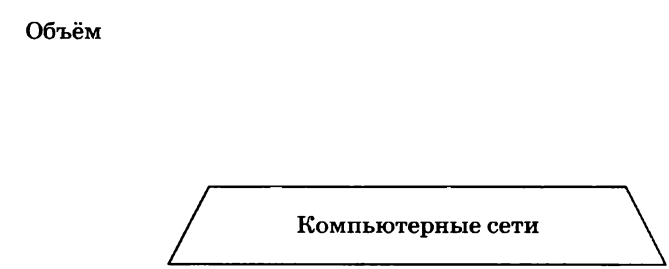


Рис. 5.20

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 ещё 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. Этот контроллер 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[1] (рис. 5.21). 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.

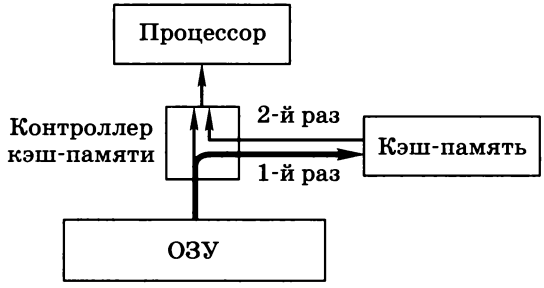


Рис. 5.21

Обычно в кэш-память заносится содержимое не только запрошенной ячейки, но и ближайших к ней (эта стрелка на рис. 5.21 показана более толстой линией). Таким образом, в кэше хранятся копии часто используемых ячеек ОЗУ, и передача этих данных в процессор происходит быстрее.

В работе кэш-памяти есть две основные трудности. Во-первых, объём кэша намного меньше объёма 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 реша-

[1] Это напоминает поиск в Интернете содержимого документа по его названию.

ет 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 (см. рис. 5.20) кэш только частично расположен внутри процессора). Кэш для программы и для данных изготавливается раздельно. Это удобно потому, что считываемую программу, в отличие от данных, не принято изменять, поэтому кэш команд можно делать проще.

Подчеркнём, что термин **«кэширование»** в вычислительной технике имеет довольно широкий смысл: речь идёт о сохранении информации в более быстродействующей памяти с целью повторного использования. Например, браузер кэширует файлы, полученные из Интернета, сохраняя их на жёстком диске в специальной папке. В накопителе на жёстком диске также используется кэширование, по тому же принципу, что и для оперативной памяти. 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 нём были предусмотрены все самые мелкие детали, которые им 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 всё больше увеличиваются в размере. Кроме того, объём обрабатываемых данных постоянно растёт. Поэтому компьютерам требуется всё больше и больше памяти, особенно в многозадачном режиме, когда одновременно запускается сразу несколько программ.

Как же согласовать эти требования с ограниченным объёмом ОЗУ? Современные операционные системы используют для этого идею **виртуальной памяти**. Предполагается, что компьютер обладает максимальным объёмом памяти, с которым может работать процессор, а реально установленное ОЗУ — лишь некоторая *часть* этого пространства. Оставшаяся часть размещается в специальном системном файле или отдельном разделе жёсткого диска. Если ёмкости ОЗУ не хватает для очередной задачи, система копирует «наименее нужную» (дольше всего не использовавшуюся) часть ОЗУ на диск, освобождая необходимый объём памяти. Когда, наоборот, потребуются данные с диска, они будут

возвращены в освобождённое 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 объёмом ОЗУ. В этом случае установка дополнительного ОЗУ может повысить быстродействие во много раз.

Использование виртуальной памяти ещё раз подтверждает, что деление памяти на внутреннюю и внешнюю — это искусственная мера. Она вызвана тем, что невозможно создать идеальную память, удовлетворяющую всем требованиям сразу.

Облачные хранилища данных. С развитием сети Интернет появились облачные хранилища данных (*Яндекс.Диск*, *Google Диск* и др.), в которых данные размещаются на серверах Интернета (в облаке). Нам достаточно знать, что «там» хранятся наши данные, и мы может обратиться к ним всегда, когда есть доступ в Интернет. При этом не надо заботиться о создании резервных копий данных, например на случай выхода из строя жёсткого диска — сохранность данных обеспечивает сервер. Скорость работы с данными в облаке обычно ниже, чем с данными на вашем компьютере. Кроме того, в облачных хранилищах трудно гарантировать безопасность данных (например, недоступность для посторонних).

Основные характеристики памяти

Для пользователя важны, прежде всего, **объём памяти, её быстродействие и стоимость.**

Информационная ёмкость — это максимально возможный объём данных, который может сохранить данное устройство памяти.

Ёмкость памяти измеряется в тех же самых единицах, что и объём информации, т. е. в битах, байтах и производных единицах (чаще всего — в мегабайтах или гигабайтах).

Для дисков часто говорят о *форматированной* и *неформатированной ёмкости*. Первая величина — это объём «полезной» памяти, а вторая включает ещё 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 случайные места памяти. На практике байты или секторы часто читаются по порядку, поэтому время ввода или вывода уменьшается.

Для ОЗУ время доступа измеряется в наносекундах (1 нс = 10^{-9} с), а для жёстких дисков — 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 учёта подготовительной стадии).

Эта характеристика обычно измеряется в мегабайтах в секунду (Мбайт/с).

Для оценки **стоимости памяти** используют отношение стоимости модуля памяти к его информационной ёмкости. 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.

Выводы

- Память — это устройство для хранения программ и данных.
- Внутренняя память используется для хранения программ и данных во время решения задачи.

- Внешняя память используется для долговременного хранения программ и данных.
- Кэш-память — это память, ускоряющая работу другого (более медленного) типа памяти за счёт сохранения прочитанных данных на случай повторного обращения к ним.
- Информационная ёмкость — это максимально возможный объём данных, который может сохранить данное устройство памяти.
- Время доступа — это интервал времени от момента посылки запроса информации до момента получения результата на шине данных.
- Средняя скорость передачи данных — это количество передаваемых за единицу времени данных после непосредственного начала операции чтения.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. С какой целью память делится на внутреннюю и внешнюю?
2. Верно ли, что внешняя память располагается вне корпуса компьютера? Приведите примеры.
3. К каким видам памяти применим принцип адресности фон Неймана?
4. Что означает термин «произвольный доступ к памяти»?
5. Зачем нужно ПЗУ в компьютере? Можно ли при необходимости изменить его содержимое на домашнем компьютере?
6. Что такое носитель информации? Какие носители вы можете назвать?
7. Какими носителями внешней памяти вы пользовались? Каков их объём и какую примерно его часть вы использовали?
8. Можно ли считать с диска отдельно взятый байт? Как его получить?
9. Какую роль играет контроллер при считывании данных с диска?
10. Почему любую программу перед выполнением требуется загрузить в оперативную память?
11. Сравните различные типы оперативной памяти. В чём состоят их достоинства и недостатки?
12. За счёт чего кэш-память повышает производительность компьютера?

13. Может ли программа обращаться к ячейкам кэш-памяти? Подумайте, относится ли кэш-память к архитектуре компьютера? Почему?
*14. Почему кэш называют ассоциативной памятью? Сравните с человеческой памятью, которую тоже часто называют ассоциативной.
*15. Чем ограничен объём виртуальной памяти?
16. Какие основные характеристики используются для памяти? В каких единицах они измеряются? Какая характеристика используется только для внешней памяти?

Подготовьте сообщение

а) «Статическая и динамическая память»
б) «Виртуальная память»
в) «Кэш-память»
г) «Что хранится в BIOS?»
д) «Как устроен флэш-накопитель?»
е) «Ассоциативная память»

Проекты

а) Сравнение флэш-накопителей
б) Сравнение облачных хранилищ данных

§ 34
Устройства ввода и вывода

Ключевые слова:

- клавиатура
- манипулятор
- мультитач
- сканер
- разрешающая способность
- 3D-сканер
- микрофон
- веб-камера
- датчик (сенсор)
- графопостроитель
- монитор
- принтер
- 3D-принтер
- плоттер

Устройства 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. Приём 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.

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, сенсорная панель, джойстик и др.), сканер, микрофон, видеокамера и другие источники мультимедийных данных, световое перо, графический планшет, датчики.

Одним из первых устройств ввода была **клавиатура**. Во многих типах клавиатур при нажатии клавиши соединяются два контакта и замыкается электрическая цепь (рис. 5.22).

Рис. 5.22

Роль контактов в наиболее распространённых моделях играет специальное *токопроводящее напыление*, наносимое на гибкую изолирующую полимерную плёнку.

Более качественные клавиатуры могут использовать, например, *герконы* (герметичные контакты), срабатывающие от приближающегося к ним магнита. Ещё один вариант — это *ёмкостные клавиатуры*, где при нажатии клавиши сближаются две небольшие пластины, образующие конденсатор. Ёмкостные клавиатуры более долговечны, так как в них нет механического контакта деталей.

Работой современной клавиатуры руководит встроенный в неё 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*[1]);
- при наличии данных посылает требование прерывания центральному процессору и затем (по его запросу) передаёт 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 номером. Распознавание кода набранного символа с учётом 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 определённые технические характеристики, такие как усилие нажатия клавиш (в ньютонах) и ход клавиш (в миллиметрах).

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.

Самый распространённый 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 своём движении с помощью электромагнитных волн (к компьютеру при этом подсоединяется специальное устройство для приёма и декодирования радиоволн — **адаптер**). Такие мыши более удобны, хотя стоят дороже и используют дополнительные элементы питания (батарейки или аккумуляторы).

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 (рис. 5.23).

---

[1] Скан-коды представляют собой номера клавиш и не имеют ничего общего с кодовыми таблицами символов, изученными в главе 2.

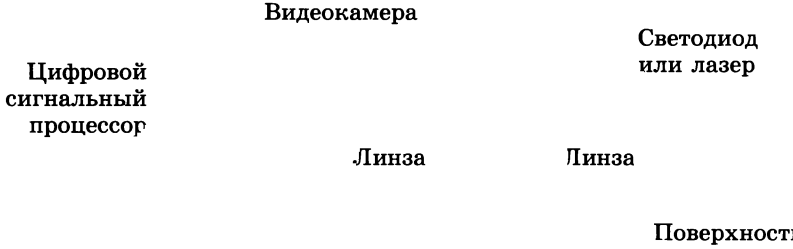


Рис. 5.23

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**: ***dots per inch***). 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 определяются датчиком, обычно они находятся в пределах от 16×16 до 30×30 пикселей. 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.

В ноутбуках в качестве встроенного «заменителя» мыши устанавливают ещё один тип манипулятора — **сенсорную панель** (англ. *touchpad*), воспринимающую движение по ней пальца. Некоторые панели способны анализировать касание в нескольких точках (режим *мультитач*, от англ. *multi-touch* — множественное касание).

Сканер — это устройство для ввода в компьютер графической информации.
Сканер передаёт в компьютер изображение документа в виде картинки. Чтобы отсканированный текст можно было редактировать, нужно превратить эту картинку в коды символов с помощью программы оптического распознавания символов (англ. **OCR**: *Optical Character Recognition*).

Принцип работы планшетного сканера, который часто используется в домашних условиях, показан на рис. 5.24. Луч света от яркого источника пробегает вдоль сканируемой поверхности, а светочувствительные датчики при этом воспринимают отражённые лучи и определяют их интенсивность и цвет. Можно сказать, что сканер — это очень сильно упрощённый цифровой фотоаппарат.

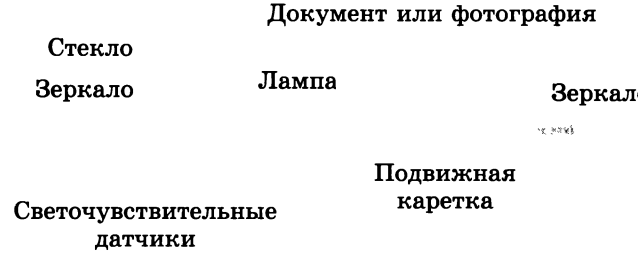


Рис. 5.24

Сканеры часто объединяют в одном корпусе с лазерным принтером, копировальным аппаратом и факсом — получается **многофункциональное устройство (МФУ)**.

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*). Рекомендуемое разрешение зависит от того, зачем сканируется материал (табл. 5.3).

Таблица 5 3

Для построения трёхмерных моделей объектов применяют **3D-сканеры**. Активные 3D-сканеры (рис. 5.25) направляют на объект луч света (или лазера), принимают отражённые от него лучи, обрабатывают полученные сигналы и строят 3D-модель. 3D-сканер в каждый момент «видит» только часть объекта, поэтому при сканировании необходимо перемещать объект или сам сканер. Для построения полной модели отдельные части приходится «сшивать».

Рис. 5.25

Существуют пассивные 3D-сканеры, которые используют видимые световые лучи окружающего освещения. Это напоминает съёмку видеокамерой с разных точек и восстановление формы предмета по множеству фотографий.

Модели, полученные с помощью 3D-сканеров, можно загружать в программы трёхмерного моделирования и затем использовать для печати на 3D-принтерах или для изготовления на станках с числовым программным управлением. Предполагается, что они найдут своё применение в археологии, медицине, моделировании одежды. киноиндустрии и других областях.

Для ввода звука используют **микрофоны** (рис. 5.26, *а*). Микрофон подключается к **звуковой карте**, которая выполняет *дискретизацию*, т. е. кодирует сигнал с микрофона в виде цепочки нулей и единиц (см. главу 2). Так можно записать любой звук, в том числе и голос человека. Более того, с помощью специальных программ можно распознать характерные звуки речи (сравнивая кусочки записанного звука с образцами) и сохранить речь в виде текста. Для этого можно использовать сайт **speechpad.ru**, который распознаёт несколько языков, в том числе и русский.

а *б* *в* *г*

Рис. 5.26

Графический планшет (рис. 5.26, *б*) — это устройство для ввода рисунков от руки. Он напоминает ручку и бумагу, только и «ручка» (перо), и «бумага» — электронные. Многие планшеты учитывают силу нажатия пера. С помощью планшета делают эскизы, наброски, переносят в компьютер рисунки, уже нарисованные на бумаге. Их можно использовать при обработке фотографий и работе с трёхмерными моделями.

Для ввода видеоизображений применяют **веб-камеры** (рис. 5.26, *в*) — небольшие цифровые видеокамеры, которые записывают изображение для передачи его по сети. Веб-камеры используют для проведения видеоконференций и в охранных системах. Сама веб-камера не может сохранить данные, для этого нужен компьютер с соответствующим программным обеспечением.

В системах автоматики для ввода данных применяют **датчики** (*сенсоры*, от англ. *sensor*, рис. 5.26, *г*) — устройства, регистрирующие какую-либо физическую величину и преобразующие её в сигналы (обычно электрические). Компьютер способен не просто хранить большое количество данных, полученных от многочисленных датчиков, но и проводить их математическую обработ-

ку. Таким образом, на основе компьютера может быть построена мощная цифровая лаборатория.

Различные датчики широко используются в робототехнике — с их помощью роботы могут ориентироваться на местности и оценивать свойства окружающей среды.

Датчики встраиваются в современные мобильные устройства: акселерометр (G-сенсор) измеряет ускорение при движении и повороте, гироскоп определяет положение устройства в пространстве, магнитометр реагирует на магнитное поле и его можно использовать как металлоискатель, датчик приближения отключает подсветку экрана, когда вы подносите телефон к уху. Во многие смартфоны и планшеты встроены барометры, датчики освещённости, приёмники спутниковой системы навигации (GPS), датчики отпечатков пальцев и др.

Многие датчики вырабатывают аналоговые данные. Поэтому для их подсоединения к компьютеру необходимо устройство, преобразующее аналоговые сигналы в цифровые, — **аналого-цифровой преобразователь (АЦП).**

Устройства вывода

Устройства вывода — это устройства, которые представляют компьютерные данные в форме, понятной человеку.

К устройствам вывода относятся мониторы, печатающие устройства (принтеры, плоттеры), наушники, звуковые колонки и др.

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.

Рг1

Рг2

См

Рис. 5.27

Схематический рисунок 5.27 изображает индикаторную панель на пульте ЭВМ первых поколений. Разноцветные колпачки патронов с лампочками помогали правильно считывать результат:

каждая группа из трёх бит — это одна восьмеричная цифра. Если считать, что горящие лампочки на рис. 5.27 обозначены тёмным цветом, 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 бумаге. 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.

Позже появились **графопостроители** (плоттеры), которые рисовали перьями на бумаге графики и картинки из линий (рис. 5.28).

Рис. 5.28

Революционным событием стало создание **мониторов**. Это позволило избавиться от ненужного расхода бумаги.

Компьютерный монитор состоит из **дисплея** (панели, на которую смотрит человек) и электронных схем, позволяющих выводить на этот дисплей текстовую и графическую информацию.

Экран любого монитора строится из отдельных точек. Как вы знаете из материала главы 2, любой цвет можно приближённо «разложить» на красную, зелёную и синюю составляющие, поэтому каждая точка монитора образована близко расположенными областями красного, зелёного и синего цветов (рис. 5.29 и цветной рисунок на форзаце). Расстояние между их центрами — доли миллиметра, поэтому глаз человека воспринимает все три составляющие как одну точку «суммарного» цвета.

Элемент экрана современного монитора — это жидкий кристалл, способный под воздействием электрического сигнала менять свои свойства. Сам жидкий кристалл не светится, он лишь регулирует пропускание света от расположенной за ним лампы.

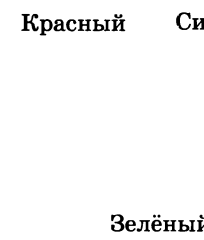


Рис. 5.29

Наиболее важные характеристики мониторов — это **размер диагонали** (в дюймах) и **максимальное разрешение** (количество точек экрана по ширине и высоте). Для жидкокристаллических мониторов максимальное разрешение — это количество элементов матрицы. Если установить другое (более низкое) разрешение, то качество изображения будет хуже, так как видеосистеме придётся «растягивать» картинку на реально существующие точки.

Процессор передаёт данные для вывода **видеокарте (видеоконтроллеру)**, которая управляет выводом изображения на монитор. Современная видеокарта содержит микропроцессор для обработки графической информации (графический ускоритель) и собственную память. Можно считать, что видеокарта — это специализированный компьютер, который ускоряет построение и вывод на монитор изображений.

В последние годы появились **3D-дисплеи (стереодисплеи)**, которые могут показывать объёмные изображения.

Печатающие устройства **(принтеры)** выводят текст и рисунки на бумагу или плёнку. Современные принтеры обрабатывают символы как графику, т. е. рисуют их. Существуют четыре основных типа принтеров: матричные, струйные, лазерные и сублимационные.

Матричные принтеры (рис. 5.30) — это принтеры с ударным принципом работы. Сейчас их используют там, где требуется печатать много и дёшево, а высокое качество печати не обязательно.

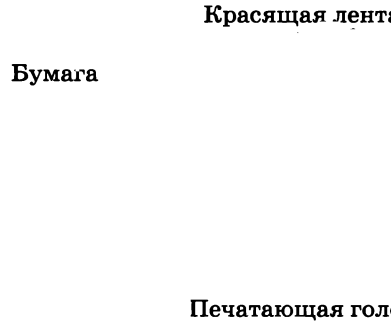


Рис. 5.30

Печатающая головка содержит вертикальный ряд иголок, которые при подаче управляющих сигналов ударяют по красящей ленте, оставляя на бумаге отпечатки в виде маленьких точек.

На таком принтере можно печатать не только тексты, но и чёрно-белые рисунки, однако вывод графики происходит очень медленно. Матричные принтеры и расходные материалы к ним (красящие ленты) очень дешёвые, они могут печатать практически на любой бумаге. Однако качество их печати невысокое, они 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 четыре картриджа: с голубой, пурпурной, жёлтой и чёрной красками (вспомните цветовую модель CMYK). Изображение строится только из точек этих цветов. В некоторых моделях для повышения качества используют шесть базовых цветов. Для печати необходима хорошая бумага, кроме того, напечатанное изображение расплывается при попадании воды.

Лазерные принтеры (рис. 5.31) печатают с очень высоким качеством. Компьютер строит в памяти полный образ страницы и передаёт его принтеру. Тот с помощью лазерного луча построчно переносит изображение на вращающийся барабан. Затем к барабану притягиваются мелкие частицы красящего порошка — **тонера**. На следующем этапе бумага прижимается к барабану, в результате на ней появляется отпечаток картинки. Чтобы краска не осыпалась, на выходе нагретые валики вплавляют частицы тонера в бумагу.

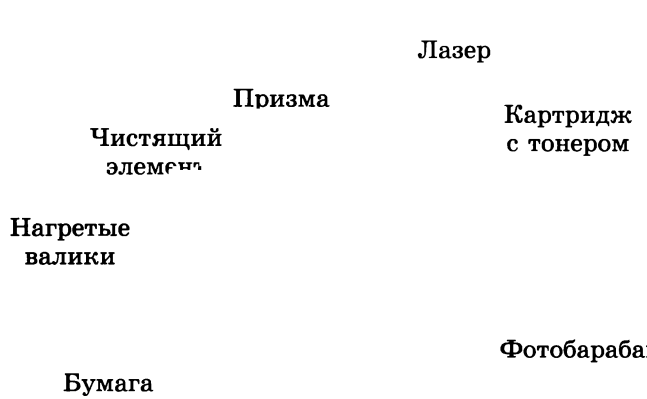


Рис. 5.31

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 плёнку с красителями, крохотные частицы краски с плёнки испаряются и «схватываются» специальной фотобумагой. Сверху наносится защитный слой, который предохраняет краску от разрушения солнечными лучами, и в итоге образуется очень стойкое изображение. Сублимационные принтеры используют для печати фотографий, а также для печати на пластиковых картах и компакт-дисках. Их недостатки — низкая скорость печати (более 1 минуты на одну фотографию) и высокая стоимость.

Важнейшая характеристика принтера — разрешающая способность.

Разрешающая способность (разрешение) принтера — это максимальное количество точек, которые он способен напечатать на единицу длины.

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*. При этом человеческий глаз не различает отдельных точек, из которых строится изображение.

Принтеры также часто сравнивают по скорости печати (в страницах 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.

Для печати изображений больших форматов, например, A1 (594 × 841 мм) и A0 (841 × 1189 мм), используют широкоформатные печатающие устройства, которые называют **плоттерами**, так же как графопостроители (рис. 5.32). Некоторые из них умеют не только печатать, но и вырезать различные фигуры.

Рис. 5.32

В конце XX века компьютеры научились обрабатывать звуковые данные. Для вывода звука чаще всего используются **наушники** и **звуковые колонки**, которые подключаются к выходу звуковой карты. Наушники и колонки — это аналоговые устройства, поэтому для вывода звука необходимо преобразовать дискретные компьютерные данные в аналоговую форму. Для этого используется специальная электронная схема в составе звуковой карты — **цифро-аналоговый преобразователь (ЦАП)**. Звуковые колонки бывают пассивные и активные; пассивные колонки работают за счёт мощности усилителя звуковой карты, а активные сами содержат усилитель и им нужно питание от сети.

Всё время разрабатываются устройства вывода новых типов. Например, созданы и получают все новые эффектные применения **3D-принтеры**, которые способны под управлением компьютера «печатать» объёмные тела из различных материалов, прежде всего из пластика (рис. 5.33).

Рис. 5.33

Устройства ввода/вывода

Некоторые компьютерные устройства нельзя однозначно отнести ни к устройствам ввода, ни к устройствам вывода. Фактически они объединяют два устройства в одном. Пример такого «гибрида» — сенсорный экран. С одной стороны, на него выводится информация, а с другой — пользователь вводит команды, нажимая на нужный участок изображения или проводя по нему пальцем. Сенсорные экраны применяют в мобильных компьютерах, платёжных и информационных терминалах, а также для представления презентаций.

В смартфонах и планшетных компьютерах сенсорный экран заменил клавиатуру и занимает всю переднюю панель. Большинство из этих устройств используют технологию **мультитач**. Например, сближая пальцы рук, можно уменьшить изображение на дисплее, а раздвигая — увеличить.

Выводы

- Устройства ввода предназначены для управления компьютером и первичного ввода данных.
- Разрешение устройства ввода — это количество точек, которые способно различить устройство на отрезке заданной длины.
- Датчик — это устройство, регистрирующее какую-либо физическую величину и преобразующее её в сигналы (обычно электрические).
- Устройства вывода — это устройства, которые представляют компьютерные данные в форме, понятной человеку.
- Разрешающая способность (разрешение) принтера — это максимальное количество точек, которые он способен напечатать на единицу длины.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Можно ли сетевую карту, через которую компьютер получает данные, назвать устройством ввода? Приведите разные точки зрения на этот вопрос.
2. 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 основной клавиши, используя клавиши сдвига?
3. Работая в электронной таблице, пользователь нажал клавиши «3», «2» и «1». Какие операции должен выполнить компьютер, чтобы соответствующее число было записано в память?
4. Зачем в клавиатуре установлен микроконтроллер?
5. Почему клавиатура не передаёт в компьютер готовые коды символов?
6. Как выполняется локализация (использование национальных символов) на клавиатуре? Найдите сведения по этому вопросу в Интернете.
7. Можно ли с помощью сканера получить фотографию реального объекта?
8. Как происходит распознавание отсканированного текста?

9. Какое устанавливать разрешение при сканировании? На что оно повлияет?
10. Сравните устройства ввода/вывода и внешней памяти. В чём их сходство и различие?
11. Что такое АЦП и ЦАП?
12. Сравните понятия «пиксель» и «точка экрана». Чем они различаются?
13. Компьютер выводит на экран монитора число, хранящееся в ячейке памяти. Какие действия он должен выполнить для этого?
14. Каким образом видеокарта позволяет разгрузить центральный процессор?
15. Что такое разрешающая способность принтера? В чём различие единиц dpi и ppi?

Подготовьте сообщение

а) «Устройство мониторов»
б) «Принципы работы 3D-сканеров»
в) «3D-принтеры»
г) «Сенсорные устройства ввода»

Проекты

а) Использование цифровой лаборатории
б) Печать на 3D-принтерах
в) Обработка данных с датчиков
г) Обработка данных с веб-камеры

ЭОР к главе 5 на сайте ФЦИОР (http://fcior.edu.ru)

- От абака до ноутбука. Поколения компьютерной техники
- Архитектура компьютера
- Архитектура машин пятого поколения
- Конфигурация компьютера. Выбор конфигурации в зависимости от решаемых задач
- Магистраль. Передача данных внутри компьютера
- Принцип открытой архитектуры
- Процессор
- Видеоплата. Звуковая плата

- Внутренняя память компьютера
- Внутренняя память компьютера. Внешняя память компьютера. Типы накопителей информации
- Устройства ввода информации
- Типы дисплеев
- Типы мониторов
- Устройства вывода информации

Практические работы к главе 5

Работа № 13 «Выбор конфигурации компьютера»
Работа № 14 «Исследование компьютера»
Работа № 15 «Моделирование работы процессора»
Работа № 16 «Использование облачных хранилищ данных»
Работа № 17 «Процессор и устройства вывода»

Глава 6
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ (§§ 35–38)

§ 35
Введение

Ключевые слова:

- системное ПО
- прикладное ПО
- кроссплатформенная программа
- инсталляция
- переносимая программа
- авторское право
- лицензия
- свободное ПО
- проприетарное ПО
- коммерческое ПО
- условно-бесплатное ПО
- бесплатное ПО

Что такое программное обеспечение?

Программное обеспечение (англ. *software* — «мягкое оборудование») — это программы, выполняющие ввод, обработку и вывод данных.
Выделяют три вида программного обеспечения: *прикладные программы*, *системные программы* и *системы программирования*.

Всех, кто работает с компьютерами, можно разделить на *пользователей*, *системных администраторов* и *программистов* (рис. 6.1). **Пользователи** решают свои задачи с помощью прикладных программ. Задача **системных администраторов** — настроить системное и прикладное ПО так, чтобы пользователи смогли нормально работать. Программисты создают новые программы с помощью **систем программирования** (инструментальных средств).

До недавнего времени программное обеспечение было «привязано» к определённой операционной системе (ОС). Например, некоторые программы работают только под управлением *Windows*, а другие — только под управлением *Linux*. В последние годы появились средства программирования, которые позволяют создавать

Рис. 6.1

так называемые **кроссплатформенные программы**, работающие в разных операционных системах. Чаще всего они разрабатываются на основе специальных библиотек, например *Qt* (**qt-project.org**), *GTK+* (**gtk.org**) и др. Эти библиотеки предоставляют программисту набор готовых функций для выполнения различных операций (например, для создания графического интерфейса). При этом все различия между операционными системами скрыты внутри библиотек, так что удаётся построить исполняемую программу для другой платформы без изменения её исходного кода.

Сейчас разработаны специальные программные средства, с помощью которых можно в одной операционной системе запускать программы, написанные для другой. Например, среда *Wine* (**www.winehq.org**) для *Linux* позволяет запускать программы, написанные для *Windows*. Среды типа *VirtualBox* (**www.virtualbox.org**) и *VMware* (**vmware.com**) дают возможность запустить на одном компьютере одновременно несколько операционных систем в разных «виртуальных машинах». Например, под управлением *Windows* в одном окне может работать виртуальная *Linux*-машина, а в другом — виртуальная машина с операционной системой *macOS*.

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.

Программное обеспечение для мобильных устройств

Мобильные устройства не могут работать без программного обеспечения. Простые современные мобильные телефоны — это компьютеры, всё программное обеспечение которых состоит из

микропрограммы, записанной в ПЗУ. Такое ПО называется **встроенным** (на английском языке — *firmware*, на жаргоне — «прошивкой»). Большинство устройств допускают замену встроенного ПО («перепрошивку»), например установку обновлённой версии.

Простые мобильные телефоны могут выполнять только программы, написанные на специальной версии языка *Java ME*. Технология *Jazelle*, применяемая в системах с архитектурой *ARM*, позволяет исполнять байт-код *Java* непосредственно в процессоре, так что простые приложения можно запускать практически на любом устройстве.

Смартфоны (более «умные» телефоны) работают под управлением операционной системы (*Android*, *iOS*, *Windows*[1] *Phone* и др.). Использование ОС позволяет легко запускать на устройстве любые программы-приложения. Их можно создавать на тех же языках программирования, которые используются при разработке программ для настольных компьютеров — *Java*, *C++*, *C#*.

Программное обеспечение для мобильных устройств должно учитывать их особенности:

- уменьшенный размер экрана;
- постоянное перемещение пользователя;
- ограниченность заряда аккумулятора;
- возможные сбои при подключении к сети.

Программы для мобильных устройств называют **мобильными приложениями**. Приложения для ОС *Android* — это программы в нестандартном байт-коде для виртуальной машины *ART* (*Andriod Runtime*). Обычно их разрабатывают на языке *Java* и и распространяют в виде файлов-архивов с расширением *apk*. Большие программы требуют кроме *apk*-файла установки дополнительного файла (он называется *кэш*), в котором хранятся данные — рисунки, звук и т. п.

Программы для операционных систем *macOS* и *iOS* пишут на языках *Objective-C* и *Swift* и распространяют как файлы с расширением *ipa*. Фактически это архив формата *ZIP*, в котором хранится программный код и все дополнительные файлы.

Меню мобильных приложений часто строится в виде длинных списков, которые удобно прокручивать с помощью сенсорного экрана (рис. 6.2).

Часто также используется горизонтальная прокрутка, позволяющая перемещаться между несколькими рабочими столами, на каждом из которых размещается свой набор пиктограмм.

---

[1] Сейчас фирма *Microsoft* пытается создать единую версию *Windows*, работающую как на стационарных, так и на мобильных компьютерах.

Рис. 6.2

Среди мобильных приложений существуют аналоги общеизвестных компьютерных программ, например офисных пакетов. Кроме того, есть и программы, учитывающие именно мобильность устройств и их технические характеристики: наличие акселерометра (датчика ускорения), барометра (датчика давления), гироскопа, геомагнитного датчика, датчика приближения, датчика освещённости, вибромоторов и т. д., которых нет у стационарных компьютеров и ноутбуков. Это, например:

- считыватели (*ридеры*, от англ. *read* — читать) QR-кодов и штрих-кодов;
- программы-переводчики, которые мгновенно переводят текст, захваченный камерой смартфона;
- разнообразные уровни, измерители углов и расстояний до объектов, которые работают через камеру смартфона и бывают весьма удобны при строительстве и ремонте;
- компасы и программы GPS-навигации (*2ГИС*, *Навигатор* и др.).

Программное обеспечение для мобильных устройств, как правило, распространяется через онлайн-магазины. Для устройств, работающих на операционной системе *Andriod*, такой магазин называется *Google Play*, для мобильных компьютеров компании *Apple* (*iPad*, *iPhone*) — *App Store*, для компьютеров, использующих операционную систему *Windows Phone* — *Windows Phone Store*.

В онлайн-магазинах есть платные и бесплатные программы (а также книги, музыка и фильмы), которые можно установить на мобильные компьютеры через Интернет. Все программы разби-

ты на категории. Пользователь просто выбирает нужную программу из меню (рис. 6.3), никаких сложных действий по установке и настройке программы не требуется.

Рис. 6.3

Большой популярностью пользуются программы для работы с различными информационными системами. С помощью спутниковой связи (*GPS*, *ГЛОНАСС*) или компьютерных сетей мобильный компьютер может определить и показать на карте место, где находится пользователь, проложить маршрут, сообщить прогноз погоды. Из любой точки, где есть доступ к Интернету, можно заказать билеты на самолёт и зарегистрироваться на рейс.

Часто компании (например, социальные сети или почтовые сервисы) специально для мобильных устройств разрабатывают собственные программы, облегчающие работу со своими веб-сайтами.

Разработчики могут размещать свои новые программы в онлайн-магазинах. За магазином следят специалисты компании, которая им управляет (*Google*, *Apple*, *Microsoft*). Каждая программа проходит тщательную проверку; если она не удовлетворяет требованиям, она отклоняется.

Как правило, за размещение платных программ в онлайн-магазине нужно заплатить небольшой взнос. Основная часть выручки от продаж программы поступает автору, часть расходуется на поддержание работы онлайн-магазина. В бесплатные программы может быть встроена реклама, могут предлагаться платные дополнительные услуги, например элементы оформления или дополнительные возможности.

Инсталляция и обновление программ

Большинство современных программ требуется устанавливать (**инсталлировать**, от англ. *install* — установить). Для этого есть несколько причин:

- 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 определённым 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 учётом 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, как правило, 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):

- просмотр лицензионного соглашения (договора о возможности использования программы);
- ввод ключа (серийного номера) программы;
- выбор компонентов программы, которые пользователь хочет установить;
- определение каталога, в котором нужно разместить файлы программы;

- распаковка и копирование файлов на жёсткий 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 — **менеджеры пакетов** (*apt-rpm* в дистрибутиве *AltLinux*, *apt-get* в *Ubuntu*, *yum* в *Fedora*, *Zypper* в *openSUSE*). Они позволяют проверить *зависимости* пакетов: устанавливается не только указанный пакет, но и другие пакеты, необходимые ему для работы.

Начинающим пользователям *Linux* проще всего использовать программы-оболочки для работы с пакетами, имеющие графический интерфейс, например *Aptitude* или *Synaptic* (рис. 6.4). В них можно мышью отметить пакеты, которые требуется установить, обновить или удалить.

Рис. 6.4

В ОС *Windows* для установки программ используется служба *Windows Installer*, которая работает с установочными пакетами — файлами в формате *msi*. Дистрибутив также может представлять собой программу (файл с расширением *exe*), которая содержит все необходимые данные и при запуске «ведёт» пользователя через все этапы установки.

Программы для операционной системы *macOS* 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 удалённую 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 жёсткий диск компьютера или запущены прямо с CD-, DVD-диска или флэш-накопителя. К этой группе относятся многие небольшие бесплатные программы для ОС *Windows* и *macOS*. Они очень полезны для тех, кто часто работает на разных компьютерах и хочет использовать привычный набор программ.

Часто делают специальные переносимые версии «обычных» программ (которые необходимо устанавливать); их можно найти, например, на сайте **portableapps.com**. Пользователь может запустить программу со своего флэш-накопителя, причём 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-накопителя. Они не меняют данные на жёстком диске, а все файлы, необходимые для работы, размещаются в оперативной памяти компьютера. Многие «живые диски» позволяют работать с файлами на жёстком диске, поэтому с их помощью можно спасти 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.

«Живые диски» можно использовать и для лечения вирусов: в этом случае компьютер загружает «чистую» операционную систему, поэтому вирус с жёсткого диска не попадает в память и не может блокировать работу антивируса.

Программное обеспечение для мобильных устройств пользователь, как правило, устанавливает через онлайн-магазины с помощью программ-установщиков, которые есть в стандартном наборе (например, *App Store* для мобильных устройств компании *Apple* или *Play Market* для ОС *Android*). При таком способе установки операционная система автоматически отслеживает обновления программы в онлайн-магазине и предлагает их установить.

Для полноценной работы со смартфоном *iPhone* или планшетным компьютером *iPad* нужно бесплатно зарегистрировать учётную запись *Apple ID* на сайте компании *Apple*. При установке любой программы нужно вводить логин и пароль к этой учётной записи, после этого загрузка и установка выполняются автоматически. Можно также загружать программы с помощью бесплатного плеера *iTunes*, который работает в операционных системах *macOS* и *Windows*. Программа сначала скачивается на стационарный компьютер или ноутбук, а затем по USB-кабелю копируется на мобильное устройство (выполняется *синхронизация*). Программы для *iPhone* и *iPad* распространяются в виде архивов с расширением *ipa*, однако установить такой архив самостоятельно штатными средствами нельзя.

В операционной системе *Android*, в отличие от *iOS*, можно устанавливать программы в обход онлайн-магазина *Google Play* (нужно только разрешить эту операцию в настройках смартфона или планшета). Для установки *apk*-архивов применяют специальные программы-установщики (например, бесплатное приложение *AppInstaller*) или браузер.

Многие современные программы способны определять по сети Интернет наличие обновлений (новых версий) и автоматически устанавливать их. Иногда это очень важно, например, когда обновление системного ПО устраняет ошибку, нарушающую безопасность компьютера. Обычно пользователь может отключить автоматическое обновление в настройках программы.

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) может ограничивать распространение и использование результатов своего труда.

В Конституции Российской Федерации записано, что «интеллектуальная собственность охраняется законом» (ст. 44 ч. 1). Интеллектуальная собственность — это права на результаты творчества человека. Эти права определены в Гражданском кодексе РФ

(часть IV, «Права на результаты интеллектуальной деятельности и средства индивидуализации»).

Согласно международному Соглашению по торговым аспектам прав интеллектуальной собственности (ТРИПС), входящему в пакет документов о создании Всемирной торговой организации, компьютерные программы с точки зрения авторского права рассматриваются как литературные произведения и имеют те же условия защиты.

Авторские права распространяются на:

- программы для компьютеров;
- базы данных.

Не охраняются авторским правом:

- алгоритмы и языки программирования;
- идеи и принципы, лежащие в основе программ, баз данных, интерфейса;
- официальные документы.

Важно понять, что *охраняется форма, а не содержание.* Это значит, что 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[1)];
- 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;
- передаётся 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 её названия;

и **имущественные права**: осуществлять или разрешать

- выпуск программы в свет;
- копирование в любой форме;
- распространение;
- изменение (в том числе перевод на другой язык).

[1)] Регистрацией программ и баз данных в России занимается Федеральный институт промышленной собственности (ФИПС).

Типы лицензий на программное обеспечение

Право на использование программы даёт документ (договор), который называют **лицензией** или **лицензионным соглашением**. Это соглашение между правообладателем и пользователем, где чётко определены права и обязанности сторон.

Обычно пользователь без дополнительного разрешения автора может

- установить программу на один компьютер (или так, как указано в договоре);
- вносить изменения, необходимые для работы программы на своём компьютере;
- сделать копию программы, чтобы можно было восстановить программу в случае сбоя.

Программы, которые получены и используются в соответствии с законом, называют **лицензионными**. Если же при копировании программы были нарушены авторские права, её называют **контрафактной** или **пиратской**.

Лицензии на ПО можно разделить на две большие группы — **проприетарные** («собственнические», от англ. *proprietary* — частное, в составе собственности, принадлежащий кому-то) и **свободные**.

Авторы **свободных программ** передают пользователю не только готовую программу, но и её исходный код, и предоставляют следующие права (свободы):

- использовать программу в любых целях;
- изучать исходный код и изменять его для своих целей;
- свободно распространять программу;
- улучшать программу и распространять изменённые версии на тех же условиях.

Проприетарным программным обеспечением называется ПО, которое ограничивает хотя бы одну из этих свобод.

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*, программа для трёхмерного моделирования *Blender* и многие другие программы.

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 **GNU GPL** (англ. *General Public Licence* — «универсальная общественная лицензия»). Её суть состоит в том,

что автор передаёт программное обеспечение в общественную собственность. Цель *GNU GPL* — предоставить пользователю права копировать, изменять и распространять (в том числе на коммерческой основе) программы, а также гарантировать, что и пользователи всех производных программ получат вышеперечисленные права. Запрещается создавать на основе свободной программы другую программу, не предоставляя пользователям её исходный код, нельзя включать программу в проприетарное ПО. Таким образом, эта лицензия не позволяет делать с программами «всё, что угодно».

Принцип «наследования» прав в свободных лицензиях называют **«копилефт»** (англ. *copyleft*). Если обычное авторское право (**«копирайт»**) ограничивает свободу копирования произведений, «копилефт» стремится использовать законы об авторском праве для расширения прав и свобод людей.

Как ни странно, свободное ПО может приносить прибыль. Например, некоторые фирмы оказывают платную техническую поддержку по развёртыванию и настройке системы *Linux*, которая относится к свободному ПО. Второй вариант заработка — продажа коммерческой лицензии в том случае, если открытый исходный код используется в коммерческих программах.

Особенность **проприетарных лицензий** в том, что автор (или правообладатель) сохраняет за собой основные существенные права (на копирование, распространение и изменение программы), предоставляя пользователю только право использовать одну или несколько копий программы.

Для сложных профессиональных программ, которые требуются многим пользователям, авторы часто требуют оплаты лицензии за каждую копию — такие программы относятся к **коммерческим**. Исходный код программы (т. е. текст, написанный программистами), как правило, не распространяется. Компании предоставляют 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. Пример коммерческого ПО — операционная система *Windows*.

Часто разработчики дают возможность бесплатно скачать пробную (англ. *trial*) версию программы из сети Интернет и проверить, 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;
- 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*). Их можно свободно скачать из Интернета и использовать бесплатно в некоммерческих целях (для частных лиц, медицинских и учебных заведений, некоммерческих организаций), однако за использование программы с целью извлечения прибыли нужно заплатить автору. Иногда бесплатно распространяются упрощённые версии коммерческих программ, в которых отключены некоторые возможности.

Ответственность за незаконное использование ПО

Обнаружив использование программы без покупки лицензии, её автор (или компания, которой принадлежат права) может через суд потребовать возмещение убытков и выплаты компенсации до 5 млн рублей (ст. 1301 Гражданского кодекса РФ).

При крупном ущербе (более 50 000 руб.) наступает уголовная ответственность (ст. 146 Уголовного кодекса РФ, «Нарушение авторских и смежных прав»). Присвоение авторства (*плагиат*) тоже рассматривается как уголовное преступление. Наказанием за плагиат может быть крупный штраф и даже лишение 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).

Выводы

- Программное обеспечение (ПО) — это программы, выполняющие ввод, обработку и вывод данных.
- Кроссплатформенная программа — это программа, версии которой существуют для различных операционных систем.
- ПО для мобильных устройств должно учитывать особенности таких устройств: уменьшенный размер экрана, постоянное перемещение пользователя, ограниченность заряда аккумулятора, возможные сбои при подключении к сети.
- Инсталляция — это установка и настройка программы на компьютере пользователя.
- ПО на мобильных устройствах обычно устанавливается и обновляется через онлайн-магазины.
- Переносимая программа — это программа, которая не требует установки.
- Программы и базы данных охраняются авторским правом.
- Лицензионное соглашение — это соглашение между правообладателем и пользователем программы, где определены права и обязанности сторон.
- Свободное программное обеспечение распространяется вместе с исходным кодом (текстами программ). Авторы свободных программ предоставляют пользователю права (свободы): использовать программу в любых целях; изучать исходный код и изменять его для своих целей; распространять программу; улучшать программу и распространять изменённые версии на тех же условиях.
- Программное обеспечение, лицензия на которое ограничивает хотя бы одну из свобод, называется проприетарным. Среди проприетарных программ выделяют коммерческие, условно-бесплатные и бесплатные.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Сравните задачи, которые решают с помощью компьютеров пользователи, системные администраторы и программисты.
2. Какие достоинства и недостатки имеет кроссплатформенное ПО?
3. Почему инсталляция необходима для многих современных программ?

4. Чем отличается дистрибутив от установленной программы?
5. Сравните методы установки программ в разных операционных системах.
6. Как вы думаете, почему ПО для мобильных устройств устанавливается, как правило, через онлайн-магазины? В чём достоинства и недостатки такого подхода?
7. Как вы думаете, зачем нужны законы об авторском праве? Что бы вы изменили в них?
8. Что значит положение «охраняется форма, а не содержание»?
9. Как действует авторское право после смерти автора?
10. Какие личные и имущественные права имеет автор?
11. Василий Федотов разработал программу SuperPuper в 2010 году. Как он должен правильно обозначить в тексте программы своё авторское право?
12. Что обычно может делать пользователь программы, не спрашивая дополнительного разрешения автора?
13. Что такое свободное ПО? Почему оно распространяется по лицензии?
14. Какие ограничения предусматривает лицензия GNU GPL?
15. Как, по вашему мнению, можно сделать разработку свободного ПО коммерчески выгодным?
16. Какие типы ПО можно законно загружать из Интернета?
17. Можно ли, не спрашивая автора (правообладателя):
    а) 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 сайте картинку с другого сайта?
18. Можно ли размещать в Интернете, не спрашивая авторов:
    а) произведения А. С. Пушкина;
    б) звукозаписи популярных исполнителей;
    в) документы, принятые Государственной думой;
    г) описание алгоритма решения квадратного уравнения;
    д) базу данных сотовых телефонов?

Подготовьте сообщение

а) «Особенности ПО для мобильных устройств»
б) «Размещения программ в онлайн-магазинах»
в) «Установка ПО для мобильных устройств»

г) «Обновление программ через Интернет»
д) «Зачем нужны инсталляторы?»
е) «Инсталляция программ в Windows, Linux и macOS»
ж) «Распространение программ в виде пакетов»
з) «"Живые диски" (Live-CD)»
и) «Лицензия GNU GPL»
к) «Свободное ПО: "за" и "против"»
л) «Авторское право в России и за рубежом»
м) «Как доказать авторское право?»
н) «Лицензии MIT и BSD»
о) «Лицензии Creative Commons»

Проекты

а) Сравнение ОС для мобильных устройств
б) Сравнение интернет-магазинов программного обеспечения
в) Сравнение свободных лицензий

Интересные сайты

portableapps.com — переносимые версии бесплатных программ
consultant.ru/document/cons_doc_LAW_64629/ — Гражданский кодекс РФ, часть четвёртая

§ 36
Программы для обработки текстов

Ключевые слова:

- виртуальная клавиатура
- редактирование текста
- форматирование текста
- текстовый редактор
- текстовый процессор
- тезаурус
- шаблон
- рассылка
- набор формул

Технические средства ввода текста

Для ввода небольших текстов в компьютер чаще всего используют клавиатуру. На планшетных компьютерах и смартфонах стандартной клавиатуры нет, в этом случае используют **виртуальные клавиатуры**, нарисованные на экране.

Большие объёмы текста вводят в компьютер (оцифровывают) с помощью сканера. Как вы знаете, сканер вводит изображение как точечный рисунок (набор пикселей). Для того чтобы значительно уменьшить объём файлов и сделать возможным поиск по тексту и его редактирование, нужно в комбинациях пикселей разных оттенков распознать символы и сохранить документ в текстовом формате. Такая задача называется задачей **оптического распознавания символов (OCR**, от англ. *Optical Character Recognition*), она относится к области искусственного интеллекта. Существуют программы, которые в некоторых случаях могут распознать даже рукописный текст, но они работают не очень надёжно.

Самая известная программа для распознавания текста — *FineReader* компании *ABBYY*. Это коммерческая программа, но часто одна из её версий поставляется на диске при покупке сканера.

Бесплатная программа распознавания *CuneiForm* относится к свободному программному обеспечению. Она работает под управлением *Windows*, *Linux*, *macOS* и других операционных систем.

Для ввода текста используют и специальные программы, распознающие речь человека. Они анализируют оцифрованный звук, сравнивая фрагменты записи с образцами звуков речи заданного языка, распознают произнесённые слова и записывают их в виде текста. Ввести текст таким образом можно в браузере *Google Chrome* на сайте **speechpad.ru**. Качество распознавания зависит от качества записи и разборчивости речи, и может достигать 80–90%.

Текстовые редакторы и текстовые процессоры

Обычно различают **редактирование текста** (изменение содержания; замена, вставка и удаление символов и слов) и **форматирование** (изменение внешнего вида текста — выбор шрифта, изменение размера, цвета, разбивка на абзацы и т. п.).

Текстовые редакторы (рис. 6.5) 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* или *UTF-8*.

Рис. 6.5

Примеры текстовых редакторов:

- *Блокнот* и *Notepad++* **(notepad-plus-plus.org)** в операционной системе *Windows*;
- *nano, gedit, KWrite* и *Kate* в операционной системе *Linux;*
- кроссплатформенные редакторы *Vim* **(www.vim.org)**, *Emacs* **(www.gnu.org/software/emacs)**, *Sublime Text* **(sublimetext.com).**

Основные **возможности** современных **текстовых редакторов:**

- ввод и редактирование текста;
- создание, открытие, 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 (*C*, *Паскаль* и др.) и разметки (*XML*, *HTML*, $LaTeX$);
- проверка орфографии.

Текстовые редакторы часто используются системными администраторами для редактирования файлов с настройками программ (файлов конфигурации). Тексты программ тоже хранятся в формате «только текст», поэтому программисты набирают и редактируют их в текстовых редакторах.

Текстовые процессоры — это следующий шаг в развитии редакторов текста. На рис. 6.6 показано окно текстового процессора *OpenOffice Writer*.

Рис. 6.6

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* (англ. *Portable Document Format* — переносимый формат документов).

Поиск и замена

Для того чтобы найти, где встречается какое-то слово в документе, можно читать текст с начала до тех пор, пока не встретится нужное слово. Однако в любых современных редакторах и текстовых процессорах есть функция поиска. Как правило, она вызывается с помощью комбинации клавиш *Ctrl+F* (от англ. *find* — найти). Можно также заменить одно слово (или словосочетание) на другое (рис. 6.7).

Рис. 6.7

Если установить флажок *Учитывать регистр*, то программа будет различать прописные и строчные буквы. Например, слово «Паровоз», записанное с прописной буквы, она уже не найдёт. Установленный флажок *Только слово целиком* говорит о том, что нужно только полное слово (а не часть слова), например сочетание букв «пар» в составе слова «паровоз» программа не находит.

Проверка правописания и грамматики

В текстовых процессорах есть встроенные средства для проверки **правописания** (орфографии). Если какое-то слово написано неверно, программа подчёркивает его красной волнистой линией (рис. 6.8).

карова

Рис. 6.8

Программа ищет набранное слово в словаре, который записан в памяти компьютера в виде файла специального формата. Если слово подчёркнуто, это значит, что его нет в словаре, хотя может быть, что оно написано верно.

Если нажать правую кнопку мыши на слове, которое, по мнению текстового процессора, написано ошибочно, появляется контекстное меню со списком возможных исправлений.

Проверка по словарю — это самый простой, но не самый лучший способ. Дело в том, что в словарь приходится добавлять все формы слова — все падежи существительных и прилагательных, все формы глаголов. Более совершенные системы умеют выполнять анализ **грамматики** — «узнают» части речи, сами образуют другие формы слов и проверяют, правильно ли построено предложение.

В некоторых случаях слово или целое предложение подчёркивается не красной, а зелёной линией. Так программа сообщает о грамматической ошибке: фраза не соответствует правилам построения предложений в выбранном языке.

По умолчанию проверка орфографии и грамматики в текстовых процессорах выполняется автоматически, но можно запустить её и вручную, нажав клавишу *F7* или соответствующую кнопку на панели инструментов.

Компьютерные словари и переводчики

Компьютеры могут быстро искать нужную информацию в больших массивах данных, поэтому сейчас компьютерные (электронные) словари практически вытеснили бумажные. В текстовых процессорах есть также и словарь специального типа — *тезаурус*.

Тезаурус (в текстовом процессоре) — это словарь, который содержит синонимы, антонимы и родственные слова.

Для вызова тезауруса в *Word* установите курсор на нужное слово и щёлкните на кнопке *Тезаурус* на вкладке *Рецензирование*. На рисунке 6.9 показан результат вызова тезауруса для слова «справедливость».

В программе *OpenOffice Writer* тезаурус для выделенного слова вызывается через контекстное меню (пункт *Синонимы*) или главное меню: *Сервис* → *Язык* → *Тезаурус*.

Для тех, кому приходится переводить тексты на другие языки, очень важны двуязычные словари. Они могут устанавливаться как отдельные программы на компьютеры или смартфоны, но существуют и онлайн-версии, т. е. сайты в Интернете. Кроме всех возможных переводов слова такие словари показывают произношение (транскрипцию) и дают возможность прослушать, как произносится это слово носителями языка (для которых этот язык родной).

Рис. 6.9

Переводить текст значительно сложнее, чем переводить отдельные слова. Прежде всего потому, что многие слова имеют несколько значений. Например, немецкое слово *Zug* имеет 21 значение. Но даже человеку, знающему значения всех слов, иногда бывает нелегко понять смысл всего предложения. Для компьютеров это тем более очень сложная задача, которая относится к области искусственного интеллекта. Однако и её постепенно решают компьютерные программы-переводчики. В них заложены правила, которые позволяют выбрать нужный вариант перевода каждого слова. Для перевода текстов по специальной тематике (например, по математике или финансам) можно подключить дополнительные словари.

К сожалению, пока автоматический перевод можно использовать только для того, чтобы получить первое представление о тексте. Его нельзя применять в ответственных случаях, особенно при переводе художественной литературы.

Шаблоны

Шаблон — это документ-заготовка, который служит основой для создания однотипных документов (писем, отчётов, квитанций, резюме и т. д.).

Использование шаблонов избавляет нас от многократного набора и копирования стандартных текстов.

Шаблон — это почти готовый документ, в котором оставлены поля для заполнения. В них нужно вписать конкретные данные — фамилии, имена, даты, адреса.

В текстовых процессорах шаблоны обычно имеют собственное расширение имени файлов: для *OpenOffice Writer* — расширение *ott*, для *Microsoft Word* — расширение *dot* или *dotx*. Обычно они хранятся в отдельной папке (*репозитарии*, или хранилище шаблонов), но могут располагаться в любом месте на устройстве внешней памяти.

При создании нового документа текстовый процессор предлагает выбрать шаблон из списка, в который входят шаблоны из хранилища на вашем компьютере. Можно также загрузить дополнительные шаблоны из Интернета.

Если открыть шаблон в текстовом процессоре (например, двойным щелчком мышью в файловом менеджере), на его основе автоматически создаётся новый документ. Чтобы изменить шаблон, нужно открыть его с помощью команды главного меню *Файл* → *Открыть* (в *Microsoft Word*) или главного меню *Файл* → *Шаблоны* (в *OpenOffice Writer*).

Каждый может создать свой шаблон и поместить его в хранилище или в любую другую папку. Чтобы сохранить документ как шаблон, в *Microsoft Word* нужно использовать команды меню *Файл* → *Сохранить как...* и выбрать формат сохранения *Шаблон* (расширения *dot* или *dotx*). В программе *OpenOffice Writer* для этого используется команда главного меню *Файл* → *Шаблоны*.

В шаблоне важно как-то выделить те части документа, которые нужно заполнить. Например, можно использовать фигурные скобки и выделение маркером (фоном).

В *Microsoft Word* удобно использовать текстовые *поля*, которые обычно затеняют серым фоном или маркером (рис. 6.10). При желании можно запретить редактирование всего документа, кроме полей.

Кому

Куда

Рис. 6.10

Для вставки текстового поля в шаблон используется кнопка Aa на вкладке *Разработчик*.

Рассылки

Рассылка — это письма с информацией, которые отправляются по электронной почте группе пользователей.

С помощью рассылок авторы сайтов в Интернете общаются с читателями, а компании — с клиентами.

Для рассылки используется список адресатов, например, в виде электронной таблицы (рис. 6.11).

Фамилия	Имя	Отчество	Пол	Адрес
Иванов	Иван	Петрович	м	*iip@mail.ru*
Петров	Сидор	Иванович	м	*petrov@gmail.com*
Сидорова	Дарья	Петровна	ж	*sdp@yandex.ru*
Семёнов	Сергей	Данилович	м	*serge1951@yahoo.com*

Рис. 6.11

В текстовом процессоре нужно заранее подготовить шаблон письма (рис. 6.12).

Уважаем

Приглашаем Вас принять участие в праздновании 20-летия нашей компании «Белка и Стрелка»!

С уважением,
Борис Стрелков,
генеральный директор,
компания «Белка и Стрелка».

Рис. 6.12

По этому шаблону строится отдельное письмо для каждого адресата, причём для мужчин нужно использовать обращение «Уважаемый», а для женщин — «Уважаемая» (в зависимости от значения в столбце «Пол»). В программе *Microsoft Word* эта задача решается с помощью вкладки *Рассылки*, а в программе *OpenOffice Writer* — через команду главного меню *Сервис* → *Рассылка писем*.

Письма создаются с помощью мастера — программы, которая задаёт пользователю вопросы и строит документы на основании его ответов. Мастер предлагает:

- выбрать список адресатов;
- выбрать шаблон письма;
- добавить в шаблон поля (места для вставки фамилии, имени и т. д.); в зависимости от значений полей (например, от пола адресата) можно добавлять в письмо разный текст;
- просмотреть готовые письма.

При желании эти письма можно сразу разослать по электронным адресам из таблицы с помощью почтовой программы.

Вставка математических формул

В электронных документах, которые готовят математики и физики, часто встречаются формулы, которые желательно оформить красиво и в одинаковом стиле.

Простые формулы можно набирать прямо в тексте, используя стили оформления, принятые для формул. Обычно в формулах используется шрифт с засечками (например, Times New Roman), имена переменных выделяются курсивом.

Для того чтобы сделать верхний (или нижний) индекс в формуле, нужно уменьшить размер символов и сдвинуть их вверх (или вниз) относительно уровня основного текста. В *Microsoft Word* для этого служат кнопки x_2 и x^2 на панели *Главная*, а в *OpenOffice Writer* — команда меню *Формат* → *Символ*.

Греческие буквы (α, β, γ и др.) и математические знаки (≥, ≤, ≈, ≠, ∈) входят в состав шрифта *Symbol*, их можно вставить в документ с помощью команды меню *Вставка* → *Символ*.

Сложные (например, «многоэтажные») формулы ввести таким простым способом не удаётся, поэтому в современных текстовых процессорах есть специальные средства для набора формул.

Наибольшими возможностями среди текстовых процессоров обладает программа *Microsoft Word*. Формула в *Word* редактируется в графическом режиме, так что сразу видно, как она будет выглядеть при печати документа. Такой режим называется *WYSIWYG* — «что видишь, то и получишь» (от англ. *What You See Is What You Get*).

На вкладке *Вставка* есть кнопка *Уравнение*, после щелчка на ней в позиции курсора вставляется поле для объекта-формулы (рис. 6.13).

Рис. 6.13

Когда выделено это поле, появляется дополнительная вкладка *Конструктор* для ввода элементов формул (рис. 6.14).

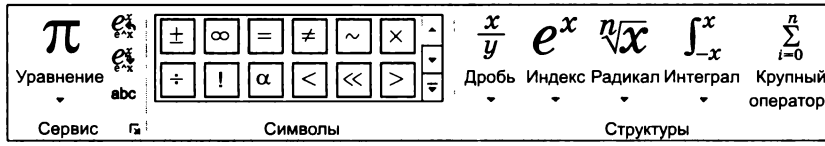


Рис. 6.14

Буквы и цифры в формулах набираются на клавиатуре. С помощью панели *Символы* можно вставить математические символы, которых нет на клавиатуре, например знаки ± и ≠. Меню *Дробь* содержит шаблоны для дробей разного стиля (числитель и знаменатель разделены чертой): $\frac{a}{b}$, a/b или a/b. Меню *Индекс* служит для добавления верхних и нижних индексов, например: x^2, x_2 или x_1^2. С помощью меню *Скобка* можно добавить «растягивающиеся» скобки, которые автоматически увеличиваются так, чтобы вместить всё содержимое.

Для примера покажем, как набрать формулу для вычисления корней квадратного уравнения $ax^2 + bx + c = 0$:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Добавим формулу и сразу вставим поле с нижним индексом $\square_{\square}$ с помощью меню *Индекс*. В первую выделенную точками область заносим «x», а во вторую — индекс «1,2» (рис. 6.15).

Рис. 6.15

Теперь нужно выйти из области нижнего индекса, нажав клавишу → на клавиатуре. Набираем на клавиатуре знак «=», затем вставляем дробь $\frac{\square}{\square}$ (рис. 6.16).

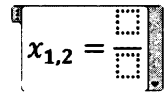


Рис. 6.16

Выделяем мышью «окно» числителя ⬚, сначала набираем «$-b$», затем вставляем знак «$\pm$» из панели *Символы*, затем знак квадратного корня $\sqrt{\square}$ из панели *Радикал* (рис. 6.17).

$$x_{1,2} = \frac{-b \pm \sqrt{\square}}{\square}$$

Рис. 6.17

Дальше заполняем оставшиеся две пустые области, используя те же приёмы.

В программе *OpenOffice Writer* можно вставить формулу с помощью редактора *Math*. После выбора команды меню *Вставка* → *Объект* → *Формула* в нижней части окна редактора появляется область, в которой вводится текстовое описание формулы, и всплывающая панель *Элементы* для выбора элементов — дробей, степеней, корней и др. В основной части документа мы видим формулу так. как она будет напечатана (рис. 6.18).

Рис. 6.18

Наберём ту же самую формулу, которую мы вводили в *Word*. Вставим объект-формулу, на панели *Элементы* в верхней части выберем группу *Формат*, а затем в нижней части панели из этой группы выберем элемент (нижний индекс). После этого в окне редактирования формулы мы увидим шаблон

```
<?>_{<?>}
```

Вместо первого сочетания <?> нужно вписать «*x*», а вместо второго (в фигурных скобках) — нижний индекс «1,2»:

```
x_{1,2}
```

Вы сразу увидите, что изображение формулы в окне редактора изменилось. Теперь вводим знак «=» и выбираем на панели *Элементы* дробь из группы *Операторы*:

```
x_{1,2} = {<?>} over {<?>}
```

Первая комбинация <?> — это место для числителя, после слова *over* (англ. «над») в фигурных скобках нужно записать знаменатель. Начинаем вводить числитель: сначала «$-b$», затем знак «±» из группы *Операторы*. Остальная часть числителя стоит под знаком квадратного корня, вставляем его из группы *Функции* и видим, что квадратный корень обозначается словом *sqrt*, а подкоренное выражение записывают в фигурных скобках:

```
x_{1,2} = {-b +-sqrt{<?>} } over {<?>}
```

Вводим выражение до конца, заполняя все пустые поля. Для ввода «b^2» используем элемент из группы *Формат*:

```
x_{1,2} = {-b +-sqrt{b^2 - 4 a c} } over {2 a}
```

Обратите внимание, что между отдельными элементами выражения (например, между «4», «a» и «c») нужно ставить пробелы.

Итак, мы познакомились ещё с одним подходом к набору формул: формула вводится как простой текст с помощью условных обозначений (например: *over*, *sqrt*). Эта запись представляет собой «программу», по которой строится изображение формулы при выводе на экран. Такой же метод используется в популярной системе набора математических текстов, которая называется TEX.

Систему TEX придумал и создал американский математик Дональд Кнут[1] для оформления своих книг, статей и презентаций. Одно из главных достоинств TEX — детально разработанная система набора сложных математических формул. Поэтому TEX очень популярен среди математиков и физиков, а также в издательствах, которые выпускают математическую литературу.

Дональд Кнут (род. в 1938 г.)

Система TEX — кроссплатформенное свободное программное обеспечение. Она выдаёт одинаковый результат на всех компьютерах под управлением всех операционных систем. Большинство систем компьютерной вёрстки при наборе документа сразу показывают результат, который получится при печати (используют режим *WYSIWYG*). В TEX используется другой принцип: автор задаёт только текст и его структуру (заголовки, параграфы, списки, таблицы), а затем программа самостоятельно форматирует документ.

Исходный документ TEX — это обычный текстовый файл с расширением *tex*. Программа TEX обрабатывает этот файл и в результате готовит документ с расширением *dvi* (англ. *device independent* — независимый от устройства), который можно вывести на экран или преобразовать в формат PDF.

Чаще всего используют TEX с пакетами расширений LaTEX или AMS-TEX, которые значительно упрощают набор документа. Мы кратко познакомимся с пакетом LaTEX.

Документ LaTEX состоит из заголовка (вступительной части, *преамбулы*) и основного текста. Заголовок начинается с объявления класса документа:

```
\documentclass{article}
```

Все команды TEX начинаются с символа «\».

Команда `\documentclass` задаёт класс документа, в фигурных скобках указан аргумент команды: слово `article` обозначает статью. Есть и другие классы документов, например: `report` — отчёт, `book` — книга, `slides` — слайды.

В заголовке нужно подключить правила переноса для того языка, на котором написан документ (`russian` — русский):

```
\usepackage[russian]{babel}
```

и кодировку, в которой закодирован текст (здесь — UTF-8):

```
\usepackage[utf8]{inputenc}
```

---

[1] Д. Кнут — автор многотомной книги «Искусство программирования», которая стала классической.

Основное содержание документа записывается между командами

```
\begin{document}
```

и

```
\end{document}
```

Главное достоинство TEX — это удобный набор формул. Формулы в тексте выделяются символами «$», например `$a^2+b^2=c^2$`. Здесь (как и в *OpenOffice Math*) знак «^» обозначает верхний индекс. Формула, занимающая отдельную строку, ограничивается с двух сторон двумя знаками «$»:

```
$$ a^2 + b^2 = c^2$$
```

Некоторые примеры формул и обозначений приведены в табл. 6.1.

Таблица 6.1

Элемент формулы	Как набрать?	Результат
Верхний индекс (степень)	`$x^2, x^{y+1}$`	x^2, x^{y+1}
Нижний индекс	`$x_2, x_{y+1}$`	x_2, x_{y+1}
Верхний и нижний индексы	`$x_1^2$`	x_1^2
Квадратный корень	`$\sqrt{a+b}$`	$\sqrt{a+b}$
Дробь	`$\frac{a+b}{2}$`	$\frac{a+b}{2}$
Высокие скобки	`$\left(\frac{1}{x}\right)^n$`	$\left(\frac{1}{x}\right)^n$
Знак умножения	`$x_1 \cdot x_2$`	$x_1 \cdot x_2$
Горизонтальный интервал	`$x \quad y$`	$x \quad y$

Информацию о других возможностях LaTEX вы можете найти в литературе или в Интернете. Вот пример готового документа:

```
\documentclass{article}
\usepackage[russian]{babel}
\usepackage[utf8]{inputenc}
\begin{document}
```

```
\textbf{Теорема Пифагора.} Пусть $a$ и $b$ --
катеты прямоугольного треугольника, а $c$ -- его ги-
потенуза.
Тогда выполняется равенство:
$$
a^2 + b^2 = c^2.
$$
\end{document}
```

Здесь использована новая команда `\textbf`, которая выделяет жирным шрифтом текст, записанный в фигурных скобках. Для того чтобы выделить текст курсивом, применяют команду `\textit`.

Если этот документ обработать программой LaTeX (например, в онлайн-редакторе на сайте **www.overleaf.com**), получим следующий результат (рис. 6.19).

Теорема Пифагора. Пусть a и b — катеты прямоугольного треугольника, а c — его гипотенуза. Тогда выполняется равенство:

$$a^2 + b^2 = c^2.$$

Рис. 6.19

Выводы

- Для ввода текста используют клавиатуру (реальную или виртуальную), сканер или системы голосового ввода.
- Текстовые процессоры могут проверять орфографию и грамматику текста. Для этого используется встроенный словарь и правила построения предложений.
- Тезаурус (в текстовом процессоре) — это словарь, который содержит синонимы, антонимы и родственные слова.
- Шаблон — это документ-заготовка, который служит основой для создания других однотипных документов (писем, отчётов, квитанций, резюме и т. д.).
- Рассылка — это письма с информацией, которые отправляются по электронной почте группе пользователей.
- Существует два способа ввода формул в текстовый документ: 1) формула редактируется в том виде, в каком она выглядит на печати (режим WYSIWYG); 2) формула редактируется в виде текста, содержащего специальные команды.

- Для набора больших математических текстов чаще всего применяют систему TEX. Исходный файл TEX — это простой текст без оформления, формулы редактируются в текстовом виде.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Обсудите в классе такую идею: «Не нужно знать грамматику, программа покажет все ошибки». Что вы думаете по этому поводу?
2. Чем отличается проверка грамматики от проверки орфографии? Какая из них сложнее?
3. Почему сложно распознавать рукописный текст? Как вы думаете, будет ли когда-нибудь решена эта задача?

Подготовьте сообщение

а) «Как проверяют орфографию?»
б) «Как проверяют грамматику?»
в) «Компьютерный перевод текстов — "за" и "против"»
г) «Распознавание символов»
д) «Система компьютерной вёрстки TEX»

Проекты

а) Подготовка рассылок
б) Сравнение программ распознавания символов
в) Библиотека шаблонов школьных документов
г) Исследование скорости набора формул в разных программах
д) Сравнение онлайн-переводчиков

Интересные сайты

abbyy.ru — сайт компании ABBYY

cognitiveforms.com — бесплатная программа для распознавания текста *CuneiForm*

newocr.com — распознавание текста онлайн

free-ocr.com — распознавание текста онлайн

ocronline.com — распознавание текста онлайн

speechpad.ru — голосовой ввод текста в браузере *Google Chrome*

lingvo-online.ru/ru — онлайн-словарь *Lingvo*

translate.ru — онлайн-переводчик *Prompt*

translate.yandex.ru — онлайн-переводчик *Яндекс*

translate.google.com — онлайн-переводчик *Google*

www.overleaf.com — онлайн-редактор LaTeX с возможностью совместной работы

§ 37
Многостраничные документы

Ключевые слова:

- формат страницы
- ориентация листа
- поля страницы
- колонтитулы
- оглавление

В этом параграфе мы научимся работать с большими документами (размером более 10–15 страниц): определять формат страницы (размер бумаги, поля); добавлять нумерацию страниц; сохранять единый стиль оформления всего документа; строить оглавление; добавлять сноски и ссылки.

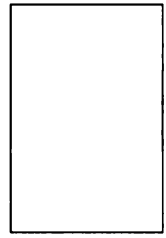
Форматирование страниц

Прежде всего, нужно настроить формат страницы: выбрать размер бумаги, ориентацию листа, установить поля. В программе *Word* эти свойства настраиваются на вкладке *Разметка страницы*, а в *OpenOffice Writer* — с помощью меню *Формат* → *Страница* (вкладка *Страница*).

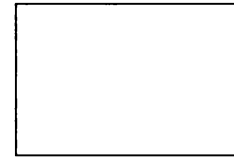
Размер бумаги можно выбрать из стандартных форматов или задать свой собственный — определить длину и ширину страницы в сантиметрах.

Ориентация листа выбирается из двух вариантов — **книжной** и **альбомной** (рис. 6.20).

Поля — это свободные области по краям страницы. У многих принтеров область печати меньше, чем полный размер листа, поэтому в любом случае поля слева или справа будут оставлены, даже если вы установите для них нулевые значения. Кроме того, читать документ без полей очень неудобно. Часто рекомендуют оставлять поле слева не менее 3 см, справа — не менее 1 см, сверху и снизу — не менее 2 см.



Книжная ориентация



Альбомная ориентация

Рис. 6.20

Для документов, которые печатаются с двух сторон листа бумаги и потом сшиваются, обычно устанавливают *зеркальные поля*. Это значит, что для страниц с чётными номерами, которые будут напечатаны на обратной стороне листа, левое и правое поля меняются местами.

Колонтитулы

Колонтитулы — это информация, которая помещается над и под текстом каждой страницы.

Вы можете добавить в документ верхние и нижние колонтитулы (рис. 6.21). В колонтитулах помещают названия глав и параграфов книг, фамилии авторов, номера страниц.

Верхний колонтитул

Нижний колонтитул

Рис. 6.21

Колонтитулы могут быть разные на чётных и нечётных страницах. Например, в колонтитуле чётных страниц может выводиться название главы, а в колонтитуле нечётных страниц — название параграфа книги. Колонтитул первой страницы также может отличаться от всех остальных (часто он вообще отсутствует).

Для того чтобы добавить информацию в колонтитулы, в программе *Word* нужно сделать двойной щелчок на верхнем или нижнем поле страницы (или использовать кнопки *Верхний колонтитул* и *Нижний колонтитул* на вкладке *Вставка*). В *OpenOffice Writer* колонтитулы включаются с помощью меню *Формат → Страница*.

Чаще всего в колонтитулах помещают номер страницы. В программе *Word* для этого используется кнопка *Номер страницы* на вкладке *Вставка*, а в *OpenOffice Writer* — меню *Вставка → Поля → Номер страницы* (курсор нужно поставить в колонтитул).

На первой (титульной) странице больших документов, например, рефератов или отчётов, номер обычно не ставят.

Оглавление

Для того чтобы облегчить чтение больших документов, в них выделяют разделы и подразделы. Они показывают **структуру документа** — его составные части. Чтобы нужный раздел было легче искать, строят **оглавление** — список всех заголовков разделов и подразделов с указанием страниц, где эти разделы начинаются (рис. 6.22). Оглавление обычно расположено в начале или в конце документа.

Для того чтобы программа обнаружила заголовки разделов и подразделов, их нужно выделить **стилями**. Для заголовков разделов используется стандартный стиль *Заголовок 1*, для заголовков подразделов — стиль *Заголовок 2* и т. д.

Когда все заголовки оформлены нужными стилями, можно строить оглавление. Установим курсор в то место, куда надо вставить оглавление. Затем в программе *Word* используем кнопку *Оглавление* на вкладке *Ссылки*, а в *OpenOffice Writer* — меню *Вставка → Оглавление и указатели*.

При редактировании документа могут измениться заголовки и номера страниц, с которых начинаются разделы. В таких случаях нужно изменять и оглавление, которое в текстовых процессорах оформляется как **поле**, т. е. обновляемый элемент

документа. Поле обновляется с помощью команды *Обновить поле* из контекстного меню[1].

Рис. 6.22

Режим структуры документа

Для быстрого перехода между разделами можно использовать **режим структуры** документа, в котором выводится только список заголовков (оформленных стилями *Заголовок 1*, *Заголовок 2* и т. д.), причём каждый заголовок становится гиперссылкой: щелчок на его названии приводит к переходу на выбранный раздел. В *OpenOffice Writer* для этого используется окно *Навигатор* (меню *Вид* → *Навигатор* или клавиша *F5*) — рис. 6.23.

В *Microsoft Word* режим структуры включается с помощью кнопки *Структура* на вкладке *Вид*. В этом режиме можно создавать структуру нового документа и изменять существующий: переводить заголовки с уровня на уровень, быстро изменять расположение разделов.

---

[1] В программе *Word* можно также использовать клавишу *F9*.

Рис. 6.23

Нумерация рисунков (таблиц, формул)

В документах с большим количество рисунков каждый рисунок должен иметь номер, причём желательно, чтобы программа поддерживала такую нумерацию автоматически. Номер рисунка в *Microsoft Word* вставляют с помощью кнопки *Вставить название* на вкладке *Ссылки*. В диалоговом окне (рис. 6.24) выбирается тип подписи (рисунок, таблица, формула), в поле *Название* после номера можно добавить какие-то символы, например точку и пробел.

Кнопка *Нумерация* позволяет использовать нумерацию с номером главы и разные стили (арабские числа, римские числа или буквы). Для обновления номера (после вставки или удаления рисунка) нужно выделить его и нажать клавишу *F9*. С помощью кнопки *Список иллюстраций* на вкладке *Ссылки* (рис. 6.25) можно добавить в документ список всех рисунков (таблиц, формул) с указанием номеров страниц.

Рис. 6.24

Рис. 6.25

В программе *OpenOffice Writer* для вставки номера и названия рисунка нужно выбрать пункт *Название* из контекстного меню этого рисунка. Соответствующее диалоговое окно показано на рис. 6.26.

Рис. 6.26

Сноски и ссылки

Сноски в документах делают для пояснений, которые автор не хочет включать в основной текст. Сноски размещают в нижней части страницы или в конце текста. Например:

Считается, что слово «информатика[1]» в современном значении образовано в результате объединения двух слов: «информация» и «автоматика».

[1] Впервые этот термин использовал немецкий учёный К. Штейнбух в 1957 году (в немецком языке – *Informatik*).

Рис. 6.27

В программе *Word* сноски добавляют с помощью кнопки *Вставить сноску* на вкладке *Ссылки* (рис. 6.25), а в *OpenOffice Writer* — с помощью меню *Вставка* → *Сноска.*

На любой рисунок (таблицу, формулу, пункт списка) с номером можно установить ссылку, для этого в *Microsoft Word* используют кнопку *Перекрёстная ссылка* на вкладке *Ссылки.* Таким же способом можно сослаться на любой элемент любого нумерованного списка в документе (рис. 6.28), например на книгу или статью в списке литературы.

Рис. 6.28

Ссылка в документе *Microsoft Word* — это поле, которое не обновляется автоматически. Чтобы обновить ссылки, нужно выделить весь текст, содержащий поля, и нажать клавишу *F9*.

В *OpenOffice Writer* ссылки также устанавливаются с помощью полей (меню *Вставка* → *Поле*). Объект, на который мы хотим сослаться, выбирается в диалоговом окне (рис. 6.29).

Рис. 6.29

Гипертекстовые документы

Гипертекст — это текст, содержащий активные ссылки (гиперссылки) на другие документы или закладки в документах.

Ссылки, с которыми вы только что познакомились, — это гиперссылки на закладки в том же документе — нумерованные рисунки, таблицы, формулы.

Кроме того, текстовые процессоры позволяют вставлять в документы **ссылки на другие файлы, а также веб-страницы и файлы в Интернете.** В *Microsoft Word* для этого используют кнопку на вкладке *Вставка* (рис. 6.30), а в программе *OpenOffice Writer* — меню *Вставка* → *Гиперссылка*.

Рис. 6.30

Текстовые процессоры позволяют сохранять документы **в формате веб-страниц** (через меню *Файл* → *Сохранить как...*). При этом получается один файл с расширением *html* (он содержит всю текстовую информацию) и отдельные файлы с рисунками. Такую страницу можно просматривать в браузере и размещать на веб-сайте, но её внешний вид будет заметно отличаться от оригинального, сохранённого в стандартном формате текстового процессора. Кроме того, и *Word*, и *Writer* при сохранении вставляют в веб-страницу много своих дополнительных данных, поэтому такой способ создания веб-страниц практически не используется.

Правила оформления рефератов

Реферат — это письменный доклад (сообщение) по определённой теме, в котором представлена информация из одного или нескольких источников.

Автор реферата должен исследовать выбранную тему, переработать найденную информацию и сделать выводы, а не просто списать текст из книжек или сайтов в Интернете. Реферат должен читаться как единый связный текст, написанный автором.

Реферат обычно содержит:

- титульный лист;
- содержание (оглавление);
- аннотацию;
- введение (1–2 страницы);

- основную часть (10–15 страниц);
- выводы или заключение (1–2 страницы);
- список использованных источников.

Каждая часть начинается с новой страницы. Страницы нумеруются, на втором листе реферата строится содержание (оглавление) с указанием номеров страниц.

Титульный лист — это первый лист реферата. На нём должны быть:

- название министерства (Министерство просвещения Российской Федерации);
- название организации (школы, лицея);
- слово «реферат», название предмета;
- название реферата;
- фамилия и имя автора;
- фамилия, имя и отчество руководителя;
- в последних двух строках — город и год.

Пример правильного оформления титульного листа показан на рис. 6.31.

Министерство просвещения Российской Федерации

Государственное бюджетное образовательное учреждение
средняя общеобразовательная школа № 1

РЕФЕРАТ
по информатике

СЕНСОРНЫЕ ЭКРАНЫ

Выполнил ученик 8^ класса
Никаноров Авенир

Руководитель
учитель информатики
Семёнова Мария Ивановна

Южноуральск
2015

Рис. 6.31

Аннотация — это краткая характеристика реферата (обычно объёмом до 500 знаков). Аннотация позволяет читателю понять, интересны ли ему тема и содержание работы, и решить, стоит ли читать её полностью. В аннотации нужно чётко определить проблему и цели работы, описать полученные результаты. В аннотации к научной статье обязательно указывают, что нового содержится в этой работе по сравнению с другими, какое значение могут иметь её результаты, где их можно применить на практике.

Во **введении** вы должны написать, почему выбрана такая тема, чем она важна для вас, чем актуальна для других, какую культурную или научную ценность она представляет. Здесь нужно поставить вопросы, на которые вы хотите ответить в конце реферата (в выводах). Часто бывает удобно писать введение уже после того, как реферат готов.

Основная часть реферата состоит из нескольких разделов, в которых постепенно раскрывается тема. Основные мысли подкрепляются доказательствами, взятыми из литературы. В конце каждого раздела основной части формулируется вывод.

Текст реферата разбивается на абзацы, каждый абзац выражает самостоятельную законченную мысль. По правилам, принятым в России, каждый абзац начинается с красной строки.

Обычно в рефератах используют два уровня **заголовков** — разделы (используется стиль *Заголовок 1*) и подразделы (стиль *Заголовок 2*). Из этих заголовков потом автоматически строится оглавление.

Рефераты принято оформлять на листах стандартного формата А4 (размером 297 × 210 мм), с книжной ориентацией страницы. Поле слева устанавливают равным 3 см, поля сверху и снизу — по 2 см, поле справа — 1,5 см.

Для основного текста обычно используют шрифт с засечками (Times New Roman, Cambria, Garamond, Bodoni), для заголовков — рубленый шрифт (Arial, Calibri, Helvetica). Основной текст набирают шрифтом размера 14 пунктов, с полуторным интервалом между строками (для того чтобы было легче читать) и выравниванием по ширине.

Заголовки выделяются жирным шрифтом. Для заголовков первого уровня часто используют шрифт размером 16 пунктов, для заголовков второго уровня — 14 пунктов, для заголовков третьего уровня — 14 пунктов, курсив. Точку в конце заголовка не ставят.

Выводы — это ваши мысли о том, что вы узнали при изучении темы. В выводах вы должны ответить на вопросы, поставленные во введении, и высказать своё мнение по теме реферата.

В **списке использованных источников** перечисляются все материалы, использованные при составлении реферата: книги, статьи, интернет-сайты, электронные ресурсы и др. Обычно они сортируются в алфавитном порядке по фамилии первого автора, а работы одного автора — по возрастанию года издания. Ссылки на интернет-ресурсы записывают в конце списка.

Приведём примеры правильного оформления элементов списка использованных источников согласно государственному стандарту (ГОСТ Р 7.0.5-2008):

Книга:

1. *Маслов Л.А.* Нашествия инопланетян на Европу [Текст] / Л.А. Маслов. — СПб.: НЛОиздат, 2001. — 344 с.

Статья в журнале:

2. *Васильев А.Н.* О глобальных физических проблемах современной астрономии [Текст] / А. Н. Васильев, А. Л. Петров, М. Д. Сидоренко // Вестн. Моск. ун-та. Сер. 3, Физика. Астрономия. — 2011. — № 6. — С. 43–45.

Электронный документ в Интернете:

3. *Артемьев К. С.* Развитие самосознания в эпоху раннего неолита [Электронный ресурс] // Вестн. НИУГУ. 2012. № 3. URL: http://www.niugu.ru/download/1238.pdf (дата обращения: 20.11.2015).

Статья на сайте в Интернете:

4. Археологи узнали о влиянии ячменя на судьбу Тибета // LENTA.RU: ежедн. интернет-изд. 2014. 21 ноября. URL: http://lenta.ru/news/2014/11/21/tibet/.

Сайт целиком:

5. Официальный сайт Государственного Эрмитажа // Санкт-Петербург, 2014. URL: http://www.hermitagemuseum.org (дата обращения: 20.11.2015)

Ссылки на использованные источники в тексте реферата заключают в квадратные скобки, например: «Как отмечал К. А. Мясоедов [5], проблема значительно шире».

Можно точно указать страницу, на которую вы ссылаетесь: «Как известно, лошади едят сено [8; с. 234].».

Ссылки необходимо делать всегда, когда вы приводите слова других людей, а также данные, результаты, диаграммы, графики, рисунки, сделанные не вами. Это обеспечивает соблюдение закона по охране авторского права и повышает доверие к вашему тексту.

Список использованных источников в электронном документе оформляется в виде нумерованного списка. Ссылки в тексте на использованные источники вставляются как поля.

Выводы

- Большие документы обычно разбивают на разделы и подразделы.
- Настройка формата страницы включает выбор размера бумаги, ориентации листа, размеров полей.
- Заголовки разделов и подразделов оформляются стилями *Заголовок 1*, *Заголовок 2* и т. д.
- Колонтитулы — это информация, которая помещается над и под текстом каждой страницы. В колонтитулах могут выводиться названия книг, глав и параграфов, фамилии авторов, номера страниц.
- Оглавление документа собирается автоматически из заголовков различного уровня, оформленных стилями *Заголовок 1*, *Заголовок 2* и т. д.
- Для рисунков, таблиц и формул нужно использовать автоматическую нумерацию.
- В документе можно устанавливать гиперссылки на другие документы; на закладки в том же документе; на веб-страницы или файлы в сети Интернет.
- Реферат — это письменный доклад (сообщение) по определённой теме, в котором представлена информация из одного или нескольких источников.
- Аннотация — это краткая характеристика документа, обычно объёмом до 500 знаков.
- Оформление списка использованных источников определяется государственным стандартом ГОСТ Р 7.0.5-2008.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Как вы рассуждаете, когда выбираете ориентацию листа (книжную или альбомную)?
2. Зачем нужны поля страницы?
3. Как вы думаете, зачем нужны колонтитулы в книгах?
4. Какую информацию, на ваш взгляд, можно помещать в колонтитулы (кроме той, которая приведена в тексте параграфа)?
5. Зачем делают разные колонтитулы и поля у страниц с чётными и нечётными номерами?
6. С какими проблемами вы можете столкнуться, если будете строить оглавление вручную?
7. Как вы думаете, где лучше строить оглавление — в начале или в конце документа?

Подготовьте сообщение

а) «Набор текста в несколько колонок»
б) «Закладки в документе»
в) «Сноски и ссылки в документе»

Проект

Оформление реферата на выбранную тему.

Интересные сайты

wordexpert.ru — профессиональная работа в *Word*
www.openoffice.org — офисный пакет *OpenOffice*
www.libreoffice.org — офисный пакет *LibreOffice*
myooo.ru — сайт о работе в программах пакета *OpenOffice*

§ 38
Коллективная работа над документами

Ключевые слова:

- рецензирование
- примечание
- исправления

Очень часто над документом работают несколько человек. Один пишет начальный вариант, другие комментируют его, высказывают замечания и предложения по улучшению. Это называется коллективной работой над документом. При этом важно, чтобы была возможность восстановить предыдущую версию в том случае, если изменения были внесены ошибочно или в результате изменений результат (текст) стал хуже. В этом параграфе вы узнаете, какие средства можно использовать для решения этой задачи.

Рецензирование

Начнём с простой задачи: у вас есть готовый текст и нужно высказать замечания по нему, предложить исправления. Такую процедуру называют **рецензированием**, а человека, выполняющего эту работу, — **рецензентом**.

Текст на бумаге очень легко исправить — вычеркнуть ненужный фрагмент текста и вписать сверху нужный, записать замечания и вопросы к автору на полях страницы. Важно, что при такой правке виден и исходный текст, и все исправления. Так раньше работали редакторы и корректоры в издательствах[1]. Но сейчас можно и нужно делать такие исправления в электронном виде.

Конечно, можно выделить (цветом или маркером) фрагменты текста, которые вызывают вопросы. Но это не очень удобно, потому что не понятно, что имел в виду тот, кто выделил текст. Комментарии можно вставлять прямо в текст, но тогда придётся выделять их цветом, чтобы они были заметны другим авторам. Лучше использовать для этой цели специальные средства текстовых процессоров.

На ленте программы *Word* есть вкладка *Рецензирование* (рис. 6.32), с помощью которой можно вносить исправления и писать примечания (комментарии).

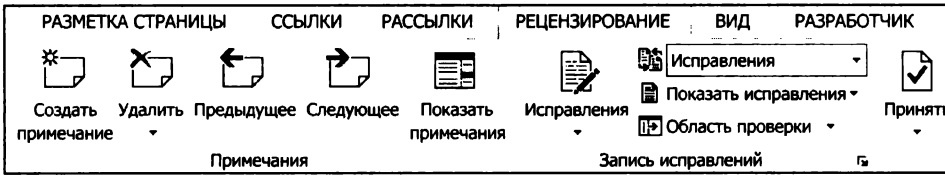


Рис. 6.32

Если выделить часть текста и щёлкнуть на кнопке *Создать примечание*, то будет создано примечание на полях страницы (рис. 6.33).

При этом важно, чтобы была возможность восстановить предыдущую версию в том случае, если изменения были внесены неверно. В этом параграфе вы узнаете, какие средства можно использовать для решения этой задачи.

kp
Я не уверен, но думаю что лучше употребить слово «ошибочно»

Рис. 6.33

Над текстом примечания показаны инициалы автора (из настроек программы).

Кнопка *Исправления* включает (а при повторном нажатии — отключает) режим исправлений. В режиме исправлений

---

[1] Редактор — это человек, который помогает автору улучшить содержание документа (например, научный редактор). Корректор исправляет ошибки в тексте (грамматические, стилистические и др.).

все ваши действия будут записываться программой и на экране будет показываться как исходный, так и исправленный вариант текста (рис. 6.34).

При этом важно, чтобы была возможность восстановить предыдущую версию в том случае,если изменения были внесены ~~неверно~~ ошибочно. В этом параграфе вы узнаете какие средства можно использовать для решения этой задачи.

Рис. 6.34

С помощью кнопок панели *Изменения* (см.рис. 6.32) можно принять или отклонить изменения. Для этой цели также можно использовать команды контекстного меню.

Рецензирование в программе *OpenOffice Writer* выполняется точно так же, как и в *Word*. Примечания вставляются с помощью меню *Вставка* → *Примечание*, а для работы в режиме исправлений нужно использовать меню *Правка* → *Изменения*.

Онлайн-офис

Бурное развитие Интернета привело к появлению **онлайн-офисов** — специальных сайтов (интернет-сервисов), которые предоставляют основные возможности офисных пакетов: текстового редактора, электронных таблиц, средств для создания презентаций. Для использования такой службы необходим компьютер с доступом в Интернет, причём не имеет значения, какая операционная система на нём установлена. Документы пользователей хранятся на сервере, для доступа к ним нужно зайти на сайт под своей учётной записью, которая защищена паролем.

Онлайн-офисы используют технологию, известную под названием **облачные вычисления** (англ. *cloud computing*). Её суть в том, что пользователь размещает свои данные на серверах Интернета и не должен заботиться о способе их хранения, операционной системе и программном обеспечении.

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. Так вы можете:

- познакомить группу людей с документом;
- организовать его обсуждение через Интернет;
- провести анкетирование;
- вместе редактировать документ.

Любой документ может быть **экспортирован** (сохранён) в файл на диске компьютера.

Самый известный онлайн-офис — **Google Docs (docs.google.com)**. Для работы с ним нужно бесплатно зарегистрироваться (получить учётную запись) на сайте **accounts.google.com**. Все ваши документы будут храниться в Интернете, в удалённом хранилище, которое называется *Google-диск*. Здесь можно создавать и редактировать текстовые документы, электронные таблицы, презентации, рисунки, формы для проведения анкетирования. Вы будете работать с документами в привычных редакторах, напоминающих программы *Microsoft Office* и *OpenOffice*, но сами документы (и программы для работы с ними!) будут находиться не на вашем компьютере, а на серверах компании *Google* в Интернете.

По умолчанию к любому документу имеет доступ только его автор. Если вы хотите пригласить к совместной работе других пользователей, нужно изменить уровень доступа к документу. Вы можете открыть доступ:

- для всех пользователей Интернета;
- только для тех, у кого есть точная ссылка;
- только для тех, кому отправлено приглашение по электронной почте.

Для всех, кому вы высылали персональное приглашение, можно установить три уровня доступа:

- только чтение;
- чтение и комментирование;
- редактирование.

В любой момент вы видите, сколько пользователей работает с документом. С ними можно общаться с помощью чата, и даже можно увидеть, как другие редактируют текст.

Все изменения автоматически сохраняются на сервере. Вы можете просмотреть, какие изменения были внесены в текст, с помощью меню *Файл → Просмотреть историю изменений*. В правой части экрана открывается список изменений. Щёлкнув на какой-нибудь строке этого списка, вы увидите версию документа после этих изменений и при желании сможете вернуться к ней (рис. 6.35).

Созданный документ можно сохранить на диске своего компьютера с помощью меню *Файл → Скачать как*. При этом программа предлагает выбрать формат, в котором сохраняется документ.

Рис. 6.35

С помощью кнопки *Создать* можно загрузить с вашего компьютера новый документ для совместной работы. Если вы загружаете файл в формате *DOCX*, он появляется с иконкой программы *Word*, и редактировать его в таком виде нельзя. При попытке открыть документ система предлагает два варианта: работать с ним в режиме чтения или перевести в формат *Google-документа* (при этом будет создан новый файл). Если необходимо редактировать текст, нужно выбрать второй вариант. Точно так же можно загружать и редактировать файлы с электронными таблицами и презентациями.

В разделе *Доступные* будут храниться те документы, которыми поделились с вами (рис. 6.36).

Рис. 6.36

Если автор документа предоставил вам соответствующие права, можно оставлять комментарии (примечания) к документу и редактировать его. Все изменения записываются в журнал и хранятся на сервере.

Правила коллективной работы

Когда вы работали над каким-то документом самостоятельно, вы могли делать с ним, что хотели, и полностью отвечали за результат. При совместной работе могут возникать проблемы, связанные с тем, что кто-то удалил или изменил фразу, написанную другим. Поэтому нужно установить правила, позволяющие избежать конфликтов.

Самое важное правило — это уважение к чужому тексту. Ни в коем случае нельзя удалять или изменять фразу, написанную другим, не согласовав это с автором. По всем спорным вопросам решение нужно принимать сообща. Для этого можно задавать вопросы своим соавторам, чтобы выяснить их позицию, использовать комментарии и чат (при работе в режиме онлайн).

Каждый имеет право на ошибку, к людям нужно относиться терпимо. Нельзя использовать в комментариях грубые выражения и резкую критику.

Все, кто совместно создают документ, являются его авторами, никто из них не может присваивать себе единоличное авторство и представлять документ как результат только своей работы.

Коллективная работа требует активного сотрудничества всех участников. Никто не должен делать всё за других, и никто не должен оставаться в стороне, «отмалчиваться». Поэтому желательно, чтобы при совместной работе появился лидер, который будет руководить работой остальных участников.

Выводы

- Рецензирование — это комментирование документа. В текстовых процессорах есть специальные возможности для рецензирования: добавление примечаний и запись исправлений.
- Онлайн-офис — это специальный сайт, предоставляющий основные возможности офисных пакетов: текстового редактора, электронных таблиц, средств для создания презентаций.
- Сайт *Google Docs* позволяет организовать совместную работу над документами. Автор документа приглашает к обсуждению других людей, которые могут читать, комментировать или редактировать документ (в зависимости от предоставленных им

прав). Все изменения записываются, в любой момент можно восстановить любую из предыдущих версий.

- При коллективной работе над документом нужно соблюдать правила, позволяющие избежать конфликтов. Главное из них — уважение к чужому тексту.

Нарисуйте в тетради интеллект-карту этого параграфа.

Вопросы и задания

1. Во всех научных журналах предусмотрено рецензирование поступающих статей. Как вы думаете, зачем это нужно?
2. Какие правила нужно соблюдать при коллективной работе с документами?
3. Какие преимущества есть у онлайн-офисов по сравнению с пересылкой документов по электронной почте?
4. В чём вы видите недостатки онлайн-офисов?
5. Чем различаются режимы доступа «для всех, у кого есть ссылка» и «для всех, кому выслано приглашение»?

Подготовьте сообщение

а) «Онлайн-офисы — "за" и "против"»
б) «Облачные хранилища данных»
в) «Облачные сервисы»

Проект

Коллективная подготовка презентации на выбранную тему

Интересные сайты

docs.google.com — документы *Google*
cloud.mail.ru — облачное хранилище данных *Облако@Mail.Ru*

ОГЛАВЛЕНИЕ

Учебное издание

Поляков Константин Юрьевич
Еремин Евгений Александрович

ИНФОРМАТИКА

(базовый и углублённый уровни)

(в 2 частях)
10 класс
Часть 1

Учебник

Ведущий редактор *О. Полежаева*
Ведущий методист *И. Хлобыстова*
Художники *Н. Новак, Я. Соловцова*
Технический редактор *Е. Денюкова*
Корректор *Е. Клитина*
Компьютерная верстка: *В. Носенко*

Подписано в печать 25.03.19. Формат 70x100/16. Усл. печ. л. 28,6.
Тираж 15 000 экз. Заказ № м7814.

ООО «БИНОМ. Лаборатория знаний»
127473, Москва, ул. Краснопролетарская, д. 16, стр. 3,
тел. (495) 181-5344, e-mail: binom@Lbz.ru
http://www.Lbz.ru, http://metodist.Lbz.ru

Отпечатано в филиале «Смоленский полиграфический комбинат»
ОАО «Издательство «Высшая школа».
Российская Федерация, 214020, Смоленск, ул. Смольянинова, 1
Тел.: +7 (4812) 31-11-96. Факс: +7 (4812) 31-31-70
E-mail: spk@smolpk.ru http://www.smolpk.ru