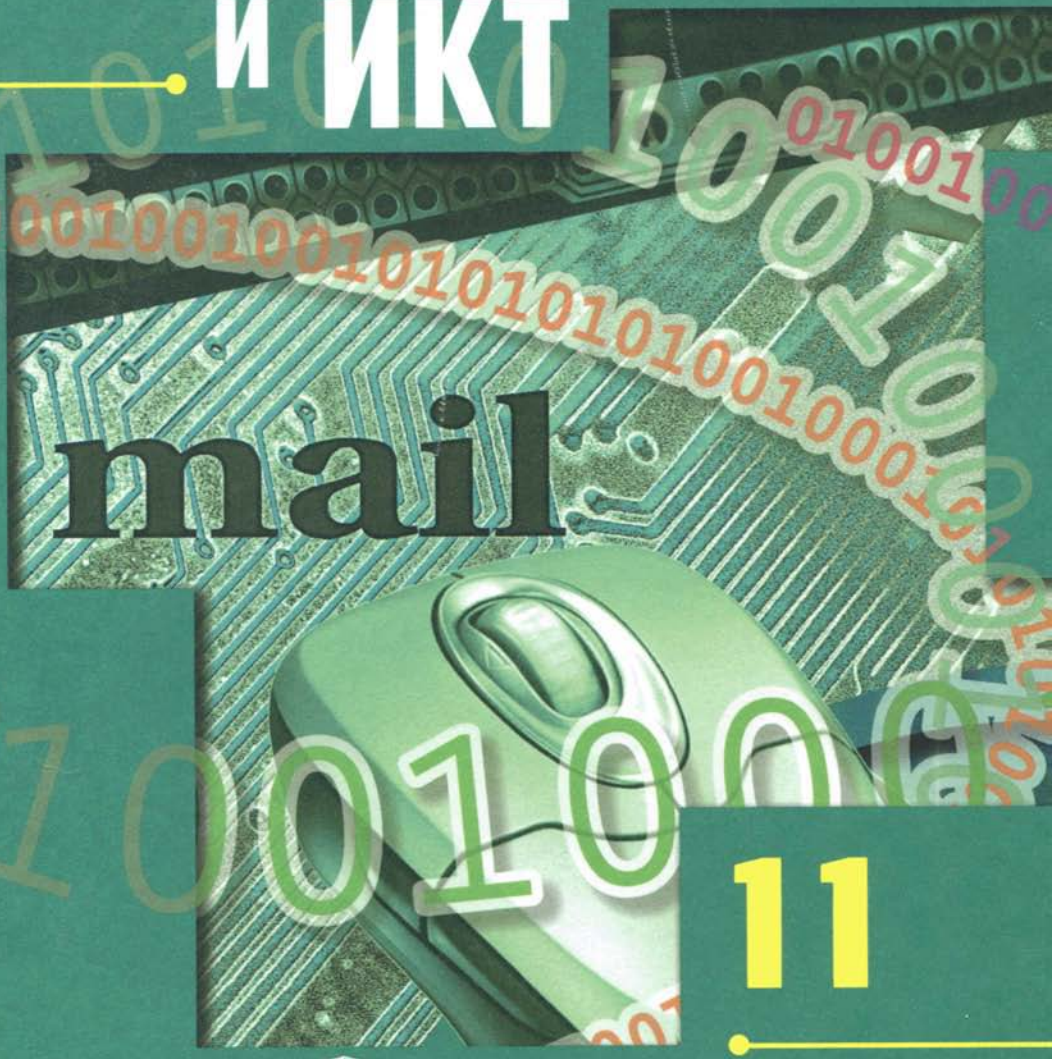
А.Г. ГЕЙН А.И. СЕНОКОСОВ

ИНФОРМАТИКА И ИКТ

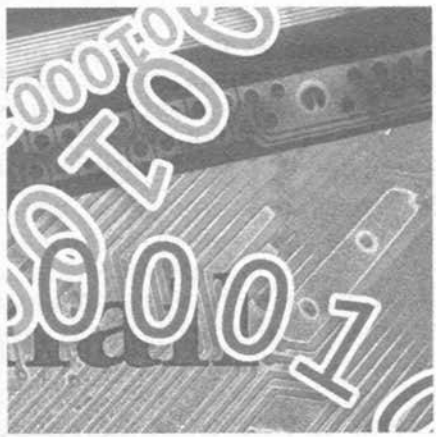


11

ПРОСВЕЩЕНИЕ
ИЗДАТЕЛЬСТВО

А. Г. ГЕЙН А. И. СЕНОКОСОВ

ИНФОРМАТИКА И ИКТ



11 класс

Учебник
для общеобразовательных
учреждений

Базовый и профильный
уровни

Рекомендовано
Министерством образования и науки
Российской Федерации

2-е издание

Москва
«Просвещение»
2012

Предисловие

Уважаемые старшеклассники!

Книга, которую вы держите в руках, — продолжение учебника «Информатика и ИКТ. 10 класс», по которому, как мы надеемся, вы занимались в 10 классе. Начнем мы с повторения: ведь нельзя двигаться дальше, не вспомнив того, что уже пройдено. Повторять материал вы будете, изучая главу 1. Но и в ней будет немало нового. Если раньше основное внимание уделялось формальной обработке информации, то в этой главе большое место уделено тому, как грамотно работать с информацией.

Работая с этим учебником, вы продолжите знакомство с тем, как кодируется информация, более глубоко, чем в основной школе, освоите создание информационных объектов разных видов — текстовых, графических, звуковых, видео и т. д. Значительное место в учебнике уделено миру Интернета. Мы также считаем важным познакомить вас с различными социальными эффектами и проблемами, возникающими в современном обществе в связи с его бурной информатизацией.

В параграфах, относящихся к профильной части курса, вы узнаете о методах, которые позволяют повысить надежность и экономичность кодирования, расширите свои знания в области структур данных, освоив понятия графа и стека, познакомитесь с методами исследования алгоритмов. Вам предстоит узнать о том, как моделируются задачи управления в терминах игры, что такое стратегия и эвристика. Именно в этих параграфах (их названия напечатаны зеленым цветом) информатика предстанет прежде всего как строгая научная дисциплина, в логике своей родственная математике (хотя самой математики в ней почти нет).

Учебник имеет модульную структуру. Это означает, что его главы могут изучаться независимо друг от друга. Особенно это важно в том случае, если информатика не будет для вас профильным курсом, какие-то темы окажутся для вас интересными и могут составить основу элективного курса.

Практические навыки в обработке информации и решении задач с помощью компьютера вы будете получать, выполняя лабораторные работы в компьютерном классе. Описание этих работ также приведено в учебнике в виде отдельного раздела. Возможно, что

на ваших компьютерах установлено программное обеспечение, отличное от описанного в учебнике. Это вовсе не означает, что вы не сможете выполнить данную работу. Ведь функции у большинства прикладных программ похожи, а как конкретно ими воспользоваться, вам подскажет учитель или вы прочитаете в инструкции пользователю. Компьютерный практикум, сопровождающий профильные параграфы учебника, играет другую роль. Его основу составляет ваша самостоятельная исследовательская работа. Компьютер в ваших руках должен стать инструментом исследования свойств информации, алгоритмов ее обработки, эффективности применяемых моделей. Поэтому в тексте, сопровождающем описание заданий для той или иной лабораторной работы, мы неоднократно призываем вас поразмышлять над полученными результатами, систематизировать их тем или иным способом, сделать необходимые выводы. Иногда такие размышления требуют времени, которое совсем необязательно проводить, сидя за компьютером. Мы надеемся, что вы ощутите себя не рабом компьютера, общение с которым сводится к вводу информации (даже если это написанная вами программа) и ожиданию от него результата, а прежде всего мыслящим человеком, для которого компьютер лишь средство усиления интеллектуальных способностей.

В нашей книге, как и в любом другом учебнике, вам встретятся новые понятия и термины. **Термины** напечатаны жирным шрифтом. Определения, свойства и правила заключены в рамку. Заучивать свойства и правила совсем необязательно, но очень важно понимать их смысл и уметь применять на практике.

Обучение невозможно без самоконтроля. Чтобы вы могли осуществлять его, учебник содержит разнообразные вопросы и задания. Вопросы помогут понять, хорошо ли вы усвоили теоретический материал. Выполняя задания, вы проверите, насколько хорошо умеете применять полученные знания. Некоторые задания будут для вас простыми, другие окажутся сложнее. Самые трудные (на наш взгляд) отмечены знаком *. В конце учебника приведены задания, по форме и содержанию близкие к тем, которые предлагаются выпускникам общеобразовательной школы на ЕГЭ по информатике.

А теперь в путь, за новыми знаниями в необъятном мире информатики!

1 ГЛАВА Информационная культура общества и личности

Сегодня жизнь каждого человека обусловлена окружающим его быстро меняющимся информационным пространством. Теперь для полноценной жизни каждому приходится принимать решения, учитывая информацию, относящуюся не только к ближайшему окружению. И наши решения зачастую уже имеют последствия не только для того узкого круга, в котором мы вращаемся, но и для многих людей, удаленных от нас географически или социально. К примеру, в течение нескольких лет сотни людей с разных концов нашей страны имеют возможность вступить в прямой диалог с президентом России, и такое общение непосредственно сказывается на ситуации в стране. Конечно, не каждый из вас станет президентом или губернатором, но круг информационной деятельности, которую вы ведете сейчас и тем более которую будете вести в будущем, только расширяется. Сегодня взаимодействие человека и информации определяет вектор развития каждой личности и общества в целом. В этих условиях формируется новый аспект культуры — **информационная культура общества**. Ее важнейшей составляющей, в свою очередь, является **информационная культура личности**. Об этих понятиях и пойдет речь в данной главе.

§ 1 Понятие информационной культуры

Человечество за века своего существования создало неисчислимые духовные и материальные ценности. Они представлены достижениями науки, отражены в произведениях искусства, выражены мировоззренческими теориями, иными словами, образуют кладезь духовной и материальной культуры. Использовать это богатство современное общество может, лишь органично сочетая применение традиционных и новых информационных технологий, посредством которых культурные достижения могут быть предоставлены каждому члену общества. Но сколько раз в истории человечества оказывалось так, что достижения культуры пропадали втуне лишь потому, что был недостаточен уровень личной культуры отдельных людей. Поэтому и сегодня важнейшей частью культуры информационного общества является информационная куль-

тура каждого человека. Это понятие подразумевает готовность человека к жизни и деятельности в высокоразвитой информационной среде, умение эффективно использовать ее возможности и защищаться от ее негативных воздействий. А для этого человеку необходимы:

- понимание закономерностей информационных процессов;
- умение организовать поиск и отбор информации, необходимой для решения стоящей задачи;
- умение оценивать достоверность, полноту, объективность поступающей информации;
- умение представлять информацию в различных видах, обрабатывать ее посредством подходящих информационных (в том числе компьютерных) технологий;
- умение применять полученную информацию для принятия решений;
- этичное поведение при использовании информации.

Обладание такой культурой предусматривает наличие у человека информационного мировоззрения, а также системы знаний и умений, обеспечивающих целенаправленную самостоятельную деятельность по оптимальному удовлетворению индивидуальных информационных потребностей. Поэтому информационная культура личности выступает важнейшим фактором успешной профессиональной и непрофессиональной деятельности, а также социальной защищенности человека в информационном обществе.

Информационное мировоззрение — это система взглядов человека на мир информации, на свое место в этом мире. Оно включает убеждения, идеалы, принципы познания и информационной деятельности.

Надо, однако, понимать, что культура — это не только обеспечение собственного комфортного существования человека в обществе. Личная культура подразумевает такую социализацию человека, при которой он не просто умело пользуется культурными достижениями общества, но и приумножает их. Информационная культура личности в контексте общечеловеческой культуры предоставляет осознанную свободу выбора, ограниченную культурными ценностями человеческой цивилизации.

Информационные возможности отдельного человека и общества в целом увеличиваются благодаря повышению доступности информации, увеличению круга ее реальных источников. Это и означает постоянное изменение информационной среды, насыщение ее все новыми компонентами. Поэтому неотъемлемой частью информационной культуры являются знание информационных технологий и умение их применять как для автоматизации рутинных операций, так и в неординарных ситуациях, требующих творческого подхода.

Появление компьютеров и глобальных телекоммуникационных сетей изменило информационный лик нашего общества. Хорошо известно, что работа с компьютером оказывает существенное влияние на формирование мышления: дисциплинирует его, способствует большей четкости, точности, строгости. Характерными чертами такого мышления являются умения планировать структуру своих действий, организовывать поиск необходимой информации, строить информационные модели объектов и процессов. Но никакой компьютер не может научить творческому применению знаний, умению думать. Научиться думать, творчески мыслить, овладевать знаниями человек должен сам, путем собственной деятельности. Говоря об информационной культуре человека, мы воспринимаем его как личность создающую. Информационная культура в этой ситуации выступает как условие освоения и адаптации человека к внешней среде и как способ гармонизации внутреннего мира человека в ходе освоения всего объема социально-значимой информации.

Известная народная мудрость гласит: «Сколько ни говори слово «халва», во рту слаще не станет». Так же и с личной культурой — она приобретается не столько теоретическим знанием ее составляющих, сколько овладением соответствующими умениями и воспитанием в себе чувств, ограждающих от неэтичного использования информации. Конечно, этические нормы в использовании информации существовали всегда. Но в современных условиях, когда информации стало больше, а доступ к ней благодаря глобальным компьютерным сетям стал намного легче, информационная этика обретает новые черты. Подробнее мы обсудим этот вопрос в главе 4.

Вопросы и задания

1. Перечислите составляющие информационной культуры человека.
2. Ниже приведено несколько определений информационной культуры личности, которые сформулированы специалистами в этой области.

   Н. И. Гендина: «Информационная культура — одна из составляющих общей культуры человека; совокупность информационного мировоззрения и системы знаний и умений, обеспечивающих целенаправленную самостоятельную деятельность по оптимальному удовлетворению индивидуальных информационных потребностей с использованием как традиционных, так и новых информационных технологий».

   Ю. С. Зубов: «Информационная культура — это систематизированная совокупность знаний, умений и навыков, обеспечивающая оптимальное осуществление индивидуальной информационной деятельности, направленной на удовлетворение как профессиональных, так и непрофессиональных потребностей».

В. Н. Михайловский: «Информационная культура — это новый тип общения, дающий возможность свободного выхода личности в информационное бытие; свобода выхода и доступа к информации как на локальном, так и на глобальном уровнях; новый тип мышления, формирующийся в результате освобождения человека от рутинной информационно-интеллектуальной работы».

Сравните эти определения: что в них общего и чем они различаются? Какое из определений, на ваш взгляд, наиболее близко к трактовке понятия «информационная культура», приведенной в объяснительном тексте параграфа?

3. Что принято включать в понятие «информационное мировоззрение»?

4. Совпадают ли, по вашему мнению, понятия «компьютерные технологии» и «информационные технологии»? Если да, то постарайтесь аргументировать свою точку зрения; если нет, то укажите, в чем состоит различие.

§ 2 Информационная грамотность — базовый элемент информационной культуры

Грамотность современного человека — базовый элемент его культуры. Неграмотный человек просто неспособен ни приобщиться ко всему тому богатству культурного наследия, которое накопило человечество за тысячелетия своего развития, ни тем более использовать его. Точно так же базовым элементом информационной культуры выступает информационная грамотность. Под обычной грамотностью мы понимаем умения человека читать, писать, воспринимать графические образы и наличие навыков использования этих умений в бытовой и профессиональной деятельности. А что такое информационная грамотность?

Кратко можно сказать так: **информационная грамотность** — это умение формулировать информационную потребность, запрашивать, искать, отбирать, оценивать и интерпретировать информацию, в каком бы виде она ни была представлена. Такое определение информационной грамотности дала Международная ассоциация школьных библиотек (IASL). И в нем присутствуют два очень важных момента.

Первый — умение формулировать информационную потребность. Это означает, что вы способны четко осознать, а затем и выразить словами, где проходит граница между вашим знанием и незнанием, которая не позволяет вам пока добиться нужного результата. Только после того, как осознана и сформулирована информационная потребность, вы сможете построить запрос на поиск

необходимой вам информации. Этому, в частности, вы учились для того, чтобы эффективно работать с компьютерными базами данных. Важно понять, что данное умение — это составной элемент общей информационной грамотности, а не только работы с конкретной информационной технологией.

Второй момент — умение интерпретировать полученную информацию. Это означает, что вы не просто законспектировали или даже усвоили добытую вами информацию, а сумели соотнести ее со своими знаниями, оценить ее соответствие вашим собственным взглядам, сделать из нее выводы и, быть может, измениться сами.

Обсудив два указанных момента, мы можем теперь более развернуто ответить на вопрос, что такое информационная грамотность. Это умения человека:

- осознать и сформулировать потребность в информации для решения той или иной проблемы;
- выработать стратегию поиска информации;
- найти соответствующую информацию;
- оценить качество информации: полноту, достоверность, актуальность, объективность;
- сформировать собственное отношение к этой информации;
- представить (аудитории или самому себе) свою точку зрения, новые знания и понимание или решение проблемы;
- оценить эффективность проделанной работы по следующим параметрам: полученные знания, приобретенные навыки и успешность в решении поставленной задачи;
- осознать, что знания и навыки, полученные в процессе решения данной проблемы (или учебной задачи), можно распространить на другие задачи и даже другие сферы деятельности человека;
- осознать влияние тех знаний, которые были получены в ходе решения задачи, на ваши личные позиции и поведение.

Уже второй раз в этой главе вы встречаетесь с такими характеристиками информации, как полнота, достоверность, актуальность, объективность. Как понимать эти термины?

Информация достоверна, если принимается, что она отражает реальное положение дел, в частности не вступает в противоречие с уже имеющейся информацией, также признаваемой в качестве достоверной. Классическая традиция подготовки энциклопедий требует, чтобы любые фактические данные подтверждались по крайней мере тремя независимыми источниками. Вовсе не исключается, что с поступлением новой информации данная информация уже перестанет быть достоверной.

Информация объективна, если она не зависит от свойств источника информации. Надо понимать, что абсолютно не зависеть от свойств источника информация не может, однако при тех

или иных условиях можно считать, что такое влияние пренебрежимо мало.

Информация актуальна (иными словами, своевременна), если она оказывает влияние на формирование целенаправленной деятельности именно в данный момент времени.

Информация полна, если ее достаточно для достижения цели. Полная информация может быть избыточной, если для достижения цели достаточно только части данной информации. К примеру, в словах русского языка гласные буквы, как правило, несут избыточную информацию, поскольку нередко могут быть однозначно восстановлены по оставшимся согласным буквам (например, слово «победа» однозначно восстанавливается по «пбд»). Это, однако, не означает, что надо стремиться избавиться от избыточности, — она часто используется для защиты информации от возможных искажений в процессе ее передачи. Ясно, что оценить полноту информации можно, только указав цель, для достижения которой она будет использоваться. В приведенном примере с русским языком целью является восстановление смысла текста. Если же целью была проверка грамотности пишущего (например, во время диктанта), то пропуск буквы «о» делает имеющуюся информацию неполной.

Важно понимать, что перечисленными свойствами информация обладает в рамках конкретно протекающего информационного процесса. Следовательно, все они носят временный, можно даже сказать, сиюминутный характер. Разумеется, какие-то свойства могут сохраняться и длительный промежуток времени, но надо всегда принимать во внимание относительный по времени характер указанных свойств. А вот влияние информационных процессов на человека и общество нередко носит глобальный характер. Об этом влиянии мы и поговорим в следующем параграфе.

Вопросы и задания

1. Почему информационную грамотность следует считать базовым элементом информационной культуры личности?
2. Раскройте содержание понятия «информационная грамотность».
3. Что такое интерпретация информации?
4. Какие свойства информации необходимо принимать во внимание при ее использовании?
5. а) Всякая реклама несет информацию для покупателя. Найдите в газете какую-нибудь рекламу и укажите, какими из перечисленных в объяснительном тексте свойств обладает информация в этой рекламе.

б) Какими свойствами, на ваш взгляд, должна обладать информация в любой рекламе?

6 Довольно широко распространено мнение, что информация, полученная человеком непосредственно наблюдением, обязательно достоверна. Приведите примеры, когда информация, получаемая посредством наблюдения, оказывается недостоверной.

7 В приведенных ниже примерах определите, полна ли информация для принятия требуемого решения. Если, на ваш взгляд, она не полна, то какую еще информацию вы хотели бы иметь?

а) Вашему классу предлагают в ближайшее воскресенье поехать на экскурсию в соседний город. Вам сообщили стоимость и продолжительность экскурсии. Требуется принять решение, заказывать ли эту экскурсию.

б) Ваша семья собирается приобрести автомобиль. Вы выяснили, автомобили каких марок продаются в доступных вам магазинах, а также для каждой марки вместимость автомобиля, расход горючего на 100 км пути, мощность двигателя, вид потребляемого топлива, цену автомобиля. Требуется принять решение, какой автомобиль купить.

8 Французский ученый Т. Адорно считает, что под влиянием телевидения сложился новый тип личности: «Знает много, понимает мало, неспособен к критической оценке, не имеет собственной точки зрения». Как вы думаете, существует ли защита от такого влияния?

§ 3 Социальные эффекты информатизации

Немногим более 10 лет назад появился первый электронный магазин. Сегодня его создатель и владелец сайта amazon.com Джеф Безос — официальный и не преследуемый государством миллиардер.

Студент, организовавший продажу пикселей своего сайта (вы когда-нибудь держали в руках пиксель?) по доллару за штуку, заработал полмиллиона за полгода — похоже, люди оценили оригинальность его идеи.

Два студента Стэнфордского университета выполнили совместный дипломный проект по организации поиска в Интернете и решили его внедрить. Сегодня их системой пользуется весь мир, а в Оксфордский словарь вошел глагол, который в русском варианте звучит как «гуглить».

Горожане покидают офисы в небоскребах и спокойно работают через Интернет, находясь при этом где-нибудь в пригородах на свежем воздухе. Городские власти в тревоге — здания надо обслуживать, иначе они придут в негодность и, может быть, даже разрушатся, а где взять на это денег, если никто не платит за аренду? Это не сказка.

И уж тем более не сказка — больницы для лечения от интернет-зависимости, которой страдают сегодня уже тысячи людей, причем главным образом очень молодых.

Конечно, эти примеры взяты из американской жизни, где телекоммуникационные технологии стали повседневностью миллионов людей раньше, чем у нас. Но жизнь развивается по одним и тем же законам, так что совсем нелишне присмотреться к тому, что уже с кем-то произошло.

Информационное общество, которое приходит на смену обществу индустриальному, характеризуется тем, что производство информационных товаров и оказание информационных услуг становятся ведущей отраслью хозяйства, именно в этой сфере занята основная часть трудоспособного населения. При этом одним из важнейших принципов общества становится его информационная открытость.

Принцип информационной открытости означает право каждого человека на получение любой информации, кроме той, распространение которой нарушает права личности или приводит к утрате безопасности существования общества. Информационное общество каждому из своих членов предоставляет равные возможности в получении информации, позволяющей полноценно реализовать себя в профессиональной и общественной деятельности, обеспечивающей реализацию гражданских прав и удовлетворение потребностей в области культуры, повышающей комфортность бытовой сферы. Для этого в информационном обществе должна быть высокоразвитая информационная инфраструктура, составной частью которой являются телекоммуникационные сети с открытым персональным доступом к ним.

Информационное общество не появляется вдруг, оно является результатом информатизации. **Информатизация общества** — это социально-экономический и научно-технический процесс активного формирования информационных ресурсов и средств их использования, а также создание оптимальных условий для удовлетворения информационных потребностей и реализации прав граждан, органов государственной власти, органов местного самоуправления, общественных объединений. Информатизация затрагивает все стороны жизни общества: материальное производство, управление, образование и даже бытовую сферу.

Основной целью *информатизации сферы материального производства* является информационное обеспечение отраслей общественного производства путем внедрения высоконадежных, эффективных автоматизированных рабочих мест, комплексной автоматизации технологических и производственных процессов, создания гибких перестраиваемых модулей, участков и производств. Информатизация призвана охватить все стадии жизненного цикла создаваемой продукции: исследование — проектирование — производство — сбыт и эксплуатация.

Материальное производство по-прежнему осуществляется в цехах заводов и фабрик, но чтобы управлять ими, совсем не обязательно находиться в непосредственной близости от этих цехов. Управленческое подразделение может располагаться там, где это наиболее эффективно для целей бизнеса. *Информатизация сферы управления* играет особую роль, поскольку она повышает эффективность управления на всех его уровнях и позволяет увеличить отдачу целенаправленной деятельности человека в других сферах. Важно и то, что применение интернет-технологий позволяет каждому человеку своевременно получать информацию, благодаря которой он может выстраивать собственное оптимальное поведение и участвовать в выработке коллективных управляющих воздействий. Ярким примером может служить ежегодное интернет-общение граждан России с президентом.

Информатизация научно-исследовательской сферы позволяет своевременно получать информацию о научных достижениях, ускоряет внедрение научных разработок в практику, ликвидирует дублирование научных работ, позволяет координировать усилия различных научных коллективов, работающих над разрешением близких проблем.

Информатизация образования улучшает его качество за счет применения в обучении информационных технологий, предоставляет возможность дистантного обучения, реализуя равные права на образование для всех граждан страны.

Информатизация бытовой сферы расширяет возможности выбора товаров и услуг для удовлетворения конкретных запросов каждого человека, повышая тем самым качество жизни.

В то же время информатизация общества приводит к появлению жизненных реалий и связанных с ними проблем, которые не могли возникнуть в доинформационном обществе. Вот один пример. Если завод выпускает автомобиль, то этот автомобиль он продаст потребителю только один раз. Соответственно и прибыль он получит от него единожды. На производство следующего автомобиля закупить сырье нужно снова. Если же фирма производит информационный продукт, например некоторую базу данных или какое-либо программное обеспечение (скажем, графический редактор), то она может продавать произвольное число его копий, не затрачивая средств ни на его изготовление, ни на приобретение сырья (в данном случае информации). Более того, на основе одной и той же информации (т. е. при разовой закупке сырья) могут быть изготовлены различные информационные объекты.

Как видите, рынок материальных товаров и услуг принципиально отличается от информационного рынка. Законы этого рынка совсем иные, а значит, и товарно-денежные отношения на информационном рынке должны регламентироваться иными механизмами, нежели отношения на рынке материальных товаров. А такой экономический механизм пока не разработан, и имен-

но применение механизмов обычного рынка в информационной сфере позволило столь стремительно разбогатеть некоторым его участникам.

Немаловажным фактором возникновения проблем в экономике является виртуализация экономических отношений. Все, к примеру, знают, что если государство для расчетов с населением просто печатает дополнительно денежные знаки, то происходит обесценивание этих денег. Заработная плата после таких действий даже у рядового работника может выражаться в миллионах, но реально на эти миллионы ничего не купишь. Но это только один, можно сказать, самый первый шаг на пути информатизации экономических отношений, ведь реальные продукты заменяются их информационной моделью в виде денежного эквивалента. Шаг этот был сделан давно. Виртуализация современной экономики состоит в том, что сделки совершаются преимущественно не в виде обмена материальными товарами и услугами, а даже без обеспечения реальными денежными средствами. Безналичные расчеты, разнообразные формы кредитования, так называемые фьючерные сделки (т. е. договоры о покупке товаров в будущем по сегодня установленным ценам) и многое другое породили такое состояние современной мировой экономики, при котором до 90% всех финансовых средств вращаются в сфере ценных бумаг и лишь 10% поддерживают материальный сектор.

Правовые проблемы возникают в связи с превращением информации в основной ресурс развития общества. Использование этого ресурса должно регламентироваться законами правового регулирования в информационной сфере. К этому надо добавить проблемы правонарушений в информационной сфере.

В социальной сфере возникающие проблемы связаны в первую очередь со значительным расширением возможностей средств коммуникации. На принятие человеком тех или иных решений все больше влияют не реальные объекты или субъекты, с которыми ему предстоит иметь дело, а их информационный образ, который доносят до людей средства массовых коммуникаций. От простенькой рекламы бытовых товаров или продуктов питания до создания нужного образа того или иного политического деятеля — все это реализуется средствами современных информационных и коммуникационных технологий.

Не случайно одной из ключевых социальных проблем современности считается проблема предотвращения массовых манипуляций человеческим сознанием, которые основываются на так называемых Public Relation Technology. Эффективное противостояние таким технологиям может оказать лишь тот человек, который обладает достаточным уровнем информационной культуры и владеет методами работы с информацией. О некоторых таких методах речь пойдет в следующем параграфе.

Вопросы и задания

1. Что понимают под информатизацией общества?
2. Каковы основные направления информатизации материально-производственной сферы?
3. Какие особенности информационных товаров и услуг отличают их от материальных?
4. Почему можно сказать, что информатизация управленческой сферы играет особую роль? (С о в е т. Готовя ответ на этот вопрос, полезно перечитать главу 5 из учебника для 10 класса.)
5. Как реализация принципов информационного общества может способствовать повышению комфортности бытовой сферы?
6. В чем, на ваш взгляд, проявляется влияние информатизации общества на формирование рынка труда?
7. Каковы возможные негативные последствия виртуализации экономики? Попытайтесь привести примеры (скажем, опубликованные в средствах массовой информации) таких негативных явлений.
8. Приведите примеры социально полезной и социально вредной информации.
9. Какие социальные проблемы может порождать информатизация общества? Попытайтесь привести конкретные примеры проявления таких проблем.
10. Составьте словарь известных вам терминов, определяющих зависимость или отчуждение отдельных людей и некоторых социальных групп от современных информационно-коммуникационных технологий, от растущих и усложняющихся информационных потоков (например, компьютерофобия, хакер и др.).
11. В чем заключается принцип информационной открытости общества?

§ 4 Методы работы с информацией

Как почувствовать себя уверенным в информационном пространстве современного общества? Выход один — овладеть методами работы с информацией.

Работу человека с информацией можно условно разделить на три этапа.

На первом из них — назовем его стартовым — важно задать самому себе вопрос: «Для чего мне нужна информация?» — и сфор-

мулировать четкий ответ. Возможно, вы готовитесь к сдаче экзаменов; возможно, предполагаете участвовать в конкурсе исследовательских проектов; возможно, хотите определиться с будущей профессией, а может быть, вам требуется разрешить какие-то личные проблемы. На этом этапе вы активизируете все те знания по интересующей проблеме, которые уже имеются у вас. Именно в этот момент у вас появляется возможность сформулировать информационную потребность, и чем четче вы ее сформулируете, тем легче вам будет найти необходимую информацию.

Второй этап — осмысление полученной информации. На этом этапе прежде всего происходит восприятие информации, т. е. своеобразная ее расшифровка, когда из отдельных частей складывается общее ее содержание. Затем осуществляется интерпретация полученной информации, при которой происходят упорядочение и классификация, объяснение найденных фактов и их суммирование, различение, сравнение и сопоставление, группировка, анализ и обобщение, соотнесение с собственным опытом, размышление над контекстом и формулировка выводов. Можно сказать, что на этом шаге происходит извлечение смысла. Наконец, размышление над полученной информацией приводит к выдвижению гипотез и высказыванию предположений, формулированию суждений, моделированию и обобщению. На этом шаге человек создает новый смысл индивидуальным, только ему одному присущим способом. В ответ на полученную информацию у него возникают собственные чувства, мысли, образы. В этой индивидуальности одно из объяснений сосуществования различных точек зрения по многим вопросам.

Тем самым осмысление информации также осуществляется по шагам:

- восприятие;
- извлечение смысла;
- создание собственного смысла.

Те, кто на этом этапе останавливается на первом шаге освоения информации, осваивают ее **репродуктивно**, они способны лишь механически воспроизвести содержание. Когда-то этого было достаточно для получения образования (стоит лишь вспомнить классическую зубрежку). Сегодня, когда необходимо постоянное обновление знаний и образование продолжается в течение всей жизни, на первый план выдвигается творческое освоение информации, которое требует ее интерпретации, оценки и создания собственных смыслов.

Третий этап — рефлексивный. На этом этапе новый опыт, новое знание встраивается в систему личностных смыслов, становясь для человека своим.

На каждом этапе работы с информацией человек использует различные операции, приемы, механизмы работы, объединенные

определенной последовательностью и правилами. Вместе они составляют то, что можно назвать стратегией обработки информации. У каждого человека такая стратегия своя, но можно выделить некоторые общие приемы, которые полезно использовать при построении собственной стратегии.

О методах поиска информации, т. е. о том, что относится к первому этапу работы с информацией, мы расскажем позже — в главе 4. А сейчас рассмотрим методы, которые помогают выполнить второй этап — осмысление информации. Осмысление информации всегда сопровождается процессом рассуждений. А что значит рассуждать? Если вы попытаетесь ответить на этот вопрос, то у вас получится примерно следующее. Рассуждать — это как бы беседовать с самим собой, задавая себе вопросы и обдумывая ответы на них, делая выводы и формулируя новые вопросы. Не случайно говорят, что по тому, как человек умеет задавать вопросы, можно определить, как он умеет думать. Большинство людей при работе с информацией ограничиваются лишь простыми вопросами, ответами на которые служат какие-то факты, имена, даты и другие сведения. Американский педагог и психолог Бенджамен Блум показал, что, помимо простых вопросов, есть еще пять групп вопросов. В таблице 1.1 приведены названия этих групп, общая характеристика входящих в них вопросов и примеры.

Вопросы и задания

1. Каковы этапы работы с информацией? Раскройте содержание каждого из этапов.

2. Ниже процитированы тексты разных авторов и для каждого из них приведены суждения читателей о содержании этого текста. По этим высказываниям попытайтесь определить, на каком шаге осмысления информации они были сделаны. Обоснуйте свой ответ.

   а) «Скажи-ка, дядя, ведь недаром
   Москва, спаленная пожаром,
   Французу отдана?» (М. Ю. Лермонтов).

   Суждения:

   1) Здесь говорится о том, какой ценой была взята Москва войсками Наполеона.

   2) Выражена просьба рассказать о событиях войны 1812 года.

   3) Здесь говорится о самоотверженности всего русского народа — как тех, кто сражался на Бородинском поле, так и мирных жителей, которые, пожертвовав всем, фактически не отдали Москву.

   б) «Не только вы собрали книги, но и книги собрали вас» (В. Шкловский).

Таблица 1.1

Название группы	Характеристика вопросов	Возможные формулировки
Простые	Относятся к фактографической информации	Кто совершил ...? Когда произошло ...? Где случилось ...? Кем наблюдалось ...? Что это было ...?
Уточняющие	Служат для установления обратной связи с источником информации	Верно ли я понял, что ...?
Интерпретационные	Выясняющие причины	Почему случилось ...? Как происходит ...? В чем заключается связь между ...?
Творческие	Содержат элементы условности, фантазии, прогноза	Как могло бы быть, если бы ...? Можно ли изменить так, чтобы ...? Следует ли ожидать ...?
Оценочные	Служат для определения значимости информации	Что привлекательного в ...? Почему есть сомнения в правильности ...?
Практические	Направлены на установление возможности использовать информацию	Где может пригодиться ...? Как использовать ...?

Суждения:

1) Книги обладают свойством объединять людей.
2) С помощью книг можно найти единомышленников.
3) По книжным собраниям можно судить о внутреннем мире людей.
4) Людей всегда объединяют общие интересы, книги — лучший фундамент для такого объединения.

❸ В чем, на ваш взгляд, заключается полезность знания групп вопросов, предложенных Б. Блумом?

❹ Ниже приведен отрывок из книги Шона Кови «7 навыков высокоэффективных тинейджеров: как стать крутым и продвинутым» (М.: Добрая книга, 2006). Прочитав его, попытайтесь сформулировать хотя бы по одному вопросу из каждой группы вопросов, фигурирующей в таблице 1.1.

«Я написал эту книгу, потому что жизнь подростков перестала быть легкой и беззаботной. Сегодня это джунгли. Если я сделал свою работу как следует, эта книга может стать компасом, который поможет тебе выбраться из них.

От реальной жизни не спрячешься, как ни пытайся. Я и не пытаюсь: вместо этого я предлагаю тебе набор инструментов, которые прекрасно помогают справляться с реальной жизнью. Что это за инструменты? Это семь навыков эффективных тинейджеров, или, другими словами, семь качеств, которыми отличаются все успешные тинейджеры в мире. Сейчас ты, наверное, гадаешь, что это за навыки... Пожалуй, я перестану держать тебя в напряжении и назову их.

Навык 1. *Будь проактивен*[1].
Возьми на себя ответственность за свою жизнь.

Навык 2. *Начинай, представляя конечную цель.*
Определи свою миссию и цели в жизни.

Навык 3. *Начинай с самого важного.*
Расставляй приоритеты и в первую очередь делай самое важное.

Навык 4. *Действуй по принципу «выиграть — выиграть».*
Поддерживай установку «выиграть может каждый».

Навык 5. *Сначала стремись понять, потом — быть понятым.*
Искренне слушай других людей.

Навык 6. *Синергия*[2].
Сотрудничай с другими людьми, чтобы достичь большего.

Навык 7. *«Затачивай пилу».*
Регулярно обновляй себя».

❺ Перечитайте еще раз § 3 и попытайтесь сформулировать хотя бы по одному вопросу из каждой группы вопросов, фигурирующей в таблице 1.1.

---

[1] Проактивное поведение — сознательный выбор, основанный на внутренних ценностях человека.

[2] Синергия — эффект от взаимодействия двух и более элементов, результат которого представляет собой больше чем сумму результатов деятельности отдельных элементов.

§ 5 Методы свертывания информации

Основная доля накопленной человечеством информации зафиксирована в текстовой форме. В древности это были глиняные таблички с клинописью и папирусы с иероглифами, позже — рукописные книги, затем появились книги печатные. Сегодня это обширные электронные текстовые ресурсы. Именно поэтому мы не представляем себе грамотного человека без умения читать и писать. Однако чтение чтению рознь. Можно за вечер «проглотить» толстую книгу и лишь поверхностно воспринять сюжетную линию. А можно задуматься над тем, что написал автор, и увидеть многообразие жизненных моделей, лишь одна из которых представлена в данном тексте. Вам может показаться, что речь сейчас идет лишь о художественной литературе. Вовсе нет — нередко там, где все казалось очевидным и простым, внимательному человеку открывается неразгаданная тайна мироздания. Таким простым и понятным кажется большинству постулат Евклида о параллельных прямых, изучаемый в школьной геометрии. Но Н. Лобачевский отказался от него и открыл новую геометрию. Такой простой и понятной казалась планетарная модель атома, предложенная Э. Резерфордом. Но М. Планк увидел в ней противоречие с законом сохранения энергии, и родилась квантовая теория, перевернувшая представления человека о законах микромира. Постоянно обдумывать получаемую информацию — нелегкий труд, но необходимый.

Напомним, что изобретение письменности, приведшее в конечном итоге к существующему ныне изобилию текстов, нередко называют первой информационной революцией. Именно это изобретение позволило человечеству накапливать и передавать информацию на носителях, более стойких к воздействию времени, нежели устное творчество и человеческая память. Тем не менее человек намного легче и лучше воспринимает видеоинформацию — в исследованиях по восприятию информации было установлено, что в среднем около 80% информации человек получает посредством зрения. Поэтому представление информации в графическом виде (например, в виде схем, диаграмм, графиков и т. п.) весьма существенное подспорье в работе с информацией. Так что преобразование текстовой информации в подходящие графические формы — важный инструмент, владеть которым должен каждый человек, считающий себя информационно грамотным. Впрочем, обратное умение — преобразование визуализированной информации в текстовую — не менее важно, поскольку умозаключения человек умеет осуществлять только в виде рассуждений, т. е. в текстовой форме.

Для выполнения таких преобразований нужно уметь выделять в информации главное, концентрируя внимание на опорных фразах и словах. Процесс перевода информации в текст-экстракт или

графическую форму с сохранением основного смысла исходной информации называется **смысловым свертыванием.** В основе смыслового свертывания лежит свойство избыточности информации, которое мы обсуждали в § 2. Только теперь, говоря об избыточности, мы имеем в виду не пропуск букв, а то, что в тексте сообщения используется гораздо больше слов, чем требуется для понимания. По исследованиям лингвистов, избыточность в русском языке достигает 60—70%. Эти, казалось бы, «лишние» слова формируют контекст, который обеспечивает условия для лучшего понимания. А вот когда смысл информации вами понят, то для дальнейшего ее использования вам и будет полезна свернутая форма представления данной информации.

Слова и фразы, которые несут основную смысловую и эмоциональную нагрузку содержания информации, называются **ключевыми.** Выбор ключевых слов и фраз — первый этап смыслового свертывания. Как правило, большинство ключевых слов находится в тексте, но далеко не всегда в заголовке или подзаголовке.

Далее из каждого смыслового фрагмента, который можно выделить в рассматриваемой информации, подбираются слова или фразы, передающие суть фрагмента, связанную со смыслом ключевых слов. Соединенные, эти фразы фактически представляют собой резюме изучаемой информации. Само слово «резюме» (от французского résumé — краткое изложение сути, выводы) означает изложение результатов исследования, итогов проделанной работы. В нем уточняются основные положения и делаются выводы, расставляются окончательные акценты.

Этот способ свертывания информации часто называют **стратегией магнита.** В этом названии отражена образная аналогия роли ключевых слов, которые, словно магнит, притягивают к себе другие смысловые фрагменты информации.

Резюме далеко не единственный вариант текстового представления свернутой информации. Аннотация, тезисы, конспект, реферат и даже обыкновенная шпаргалка (умно составленная) — все это примеры свернутой информации.

Теперь поговорим о графических формах свертывания информации. К ним относятся схемы, графики, диаграммы, таблицы, карты, планы и т. п. Графические формы позволяют наглядно представить сходство и различие, связи, иерархию (расположение от высшего к низшему), динамику процессов и явлений. Такое представление помогает анализировать явления и факты, классифицировать их, выделять в них главное и существенное.

Одним из способов графического свертывания информации является схематизация. Схема (от греческого σχεμα— наружный вид, форма) — это рисунок, в котором условными графическими обозначениями показаны составные части чего-либо и связи между ними. Схематизация позволяет лучше усвоить информацию, а затем ее воспроизвести.

Составление схемы, как и любое свертывание информации, начинается с выделения ключевых слов и фраз. Затем, как и в стратегии магнита, подбираются слова, отражающие смысл того или иного фрагмента информации, связанный с ключевыми словами. Их помещают в геометрические фигуры (прямоугольники, круги, овалы и т. п.), соединяя линиями или стрелками. Иногда рисуют символические изображения объектов.

Схема позволяет быстро охватить взором всю картину. Связи наглядно показывают логические отношения между причиной (или условием) и следствием, проблемой и ее решением, главным и второстепенным, а также соподчиненность целого и части. Составление схемы заставляет выделять элементы и соединять их в целостную картину, помогая тем самым осмыслить информацию.

Разновидностью графической схемы является **кластер** (от англ. cluster — гроздь, пучок). Кластером называют схему, применяющуюся для структурирования сведений по одному вопросу и имеющую обычно вид грозди. В кластере отражаются различные связи и направления, каждое из которых может развиваться дальше по принципу подчинения. К примеру, информацию, с которой вы ознакомились в § 3, можно в свернутом виде представить кластером, изображенным на рисунке 1.1.

Нередко свертывание информации в кластер можно осуществить, используя прием, представленный на рисунке 1.2. В качестве ключевого слова или ключевой фразы выступает некоторый

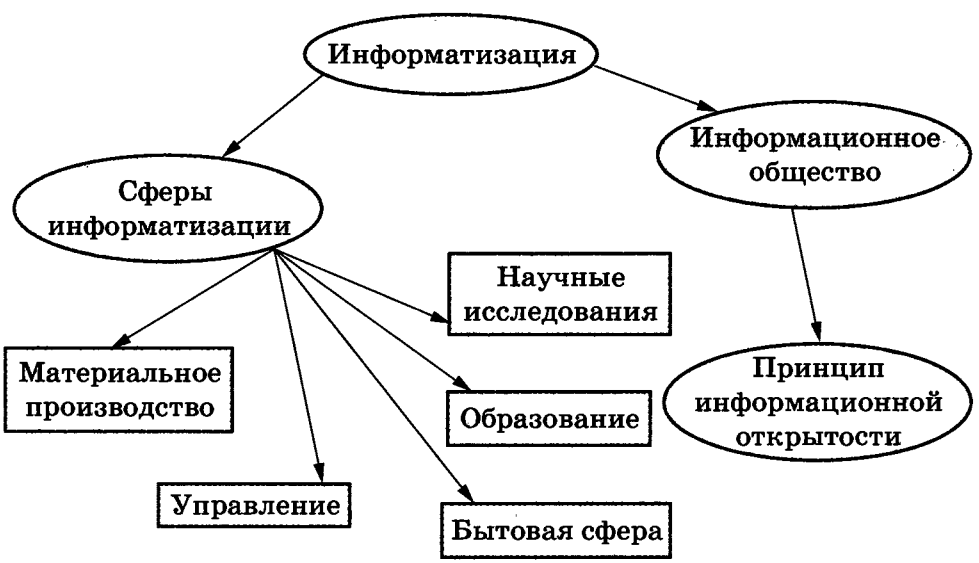


Рис. 1.1

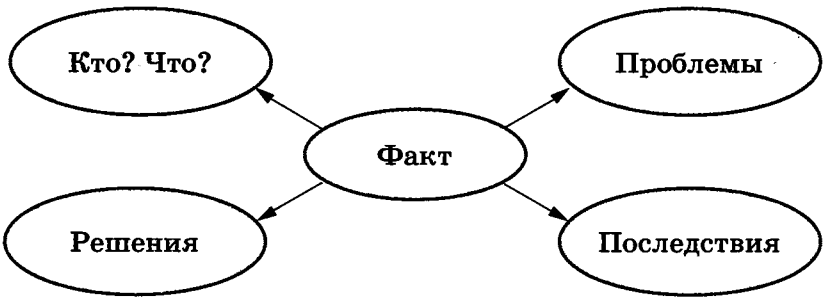


Рис. 1.2

факт; эта фраза записывается в центральный овал. А в окружающие его овалы записываются краткие ответы, характеризующие указанные в них темы.

Для примера применим этот прием к анализу следующей информации:

«Функция прессы в обществе всегда одна и та же. При чтении происходит осмысление людьми текущей реальности. Если какие-то проблемы замалчиваются, как, например, дедовщина в армии, насилие в тюрьмах или чудовищное положение с русским языком, то этих сторон реальности для людей как бы не существует. И тогда в сознании людей возникает диссонанс, вызванный расхождением между тем, что они видят в реальности, и тем представлением об этой реальности, которое навязывается им извне».

Нетрудно понять, что ключевые слова здесь — «негативные явления», хотя в самом тексте этих слов нет. Их-то и надо вписать в центральный овал. Проблема, разумеется, состоит в замалчивании этих явлений. В овал со словами «Кто? Что?» с очевидностью вписывается слово «пресса». Последствия — это диссонанс в сознании людей. В кластере, представленном на рисунке 1.3, остался один незаполненный овал — решения. Но нам кажется, что заполнить его теперь не составит труда никому из читателей. Решение

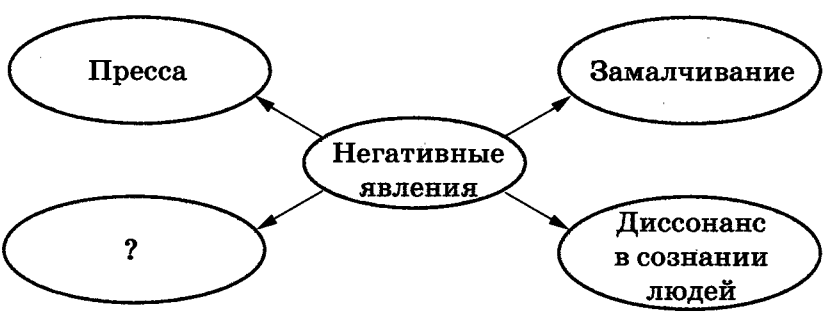


Рис. 1.3

можно сформулировать так: освещение острых вопросов. Заметьте, в самом тексте о решении речи не шло, но работа с представленной информацией предложенным методом потребовала осмысления информации, при котором обойтись без вывода оказалось невозможным.

Существует много других методов свертывания информации. Те, кто заинтересовался этим, могут познакомиться с ними по книгам, приведенным в списке литературы на с. 329.

Вопросы и задания

1. Для каких целей может оказаться полезным смысловое свертывание информации?
2. Какое свойство информации позволяет осуществлять ее свертывание?
3. Какие слова называют ключевыми?
4. В чем состоит стратегия магнита?
5. а) Что такое кластер как инструмент представления информации в свернутом виде?

   б) Найдите в словаре другие значения слова «кластер», относящиеся к обработке данных.
6. Прочитайте приведенный ниже текст, взятый из книги «Юношеские субкультуры Урала: мини-энциклопедия» (Екатеринбург: Союз юнкоров, 2005).

   «При существовании в мире мощного слоя базовой культуры у молодого человека вдруг возникает стремление отгородиться, уйти в глубину — «под» культуру (лат. sub — под + культура), искать или создавать автономную, независимую субкультуру. Явление это возникло не вдруг и имеет свою историю и соответственно модели бытования. Еще в начале прошлого века в Екатеринбурге и Перми появились «волчата», «огарки», выстраивающие свое мировоззрение как инакомыслящие, инакочувствующие. Затем были клубы любителей самодеятельной песни и клубы любителей фантастики, стремящиеся встать над суетой, обращавшиеся к вечным ценностям в противовес классовым, хиппи, проповедовавшие любовь и уходившие в противостояние. XXI век усложнил социокультурное пространство, стала приветствоваться толерантность, которая породила множество новых культурных модификаций. Теперь меняется модель субкультуры — она стремится упорядочить окружающий хаос, ограничить свой мир, создать какую-то защищенность, действуя в соответствии с собственными законами».

   а) Взяв слова «молодой человек» в качестве ключевых, сверните эту информацию с помощью стратегии магнита.

   б) Выполните то же задание, взяв слово «инакомыслящие» как ключевое.

в) Выполните то же задание, взяв слова «модель субкультуры» как ключевые.

г) Используя пункты а—в, составьте резюме.

7 Перечитайте еще раз § 4 и попытайтесь построить схему в виде кластера, отражающего содержание этого параграфа.

8 Применяя метод, представленный на рисунке 1.2, постройте схему для следующего текста.
«По результатам исследований, проведенных в России и США, общая продуктивность восприятия экранного текста на 20—30% ниже, чем напечатанного на бумаге. Воображение при таком чтении пассивно. Поэтому возможности эмоционального, эстетического и аналитического чтения при работе с электронными текстами ограниченны».

9* Обучение и самообучение — это информационные процессы, в которых вы участвуете практически всю жизнь. В школе (и не только) для проверки того, как вами воспринят учебный материал, применяются, как вы знаете, вопросы и задания. Так же как этап осмысления, вопросы и задания делятся на три категории: репродуктивные (т. е. предусматривающие простое воспроизведение информации), продуктивные (т. е. требующие применить полученные знания и умения в стандартных ситуациях) и творческие (т. е. те, в которых требуется применить знания в нестандартной ситуации, возможно, модифицировав освоенные методы). Для каждого вопроса и задания к этому параграфу определите, к какой категории оно относится.

§ 6 Моделирование — краеугольный камень информационного мировоззрения

Вы уже не новички в информатике; в предшествующие годы вы знакомились с закономерностями протекания информационных процессов в живой природе, человеческом обществе и различных социотехнических системах, изучали средства информационных технологий. Освоение методов моделирования и формализации играет значительную роль в формировании информационного мировоззрения. Ведь все наши знания и представления об окружающем мире отражены в форме моделей. Одни из них отражают физическую реальность, другие — мир человека, третьи — всего человеческого сообщества. Особую роль играют так называемые глобальные модели (о них мы рассказывали в конце учебника для 10 класса), в которых деятельность человека рассматривается как один из факторов изменений в планетарном масштабе. Именно благодаря моделированию человек имеет возможность объяснять происходящие явления и прогнозировать последствия природных

процессов и собственной деятельности. Решая любую жизненную задачу, человек сначала строит информационную модель, определяя, какие факторы являются существенными для ее решения, каковы исходные данные, что является результатом и как связаны между собой исходные данные и результат. Восприятие человеком окружающего мира разумом, а не только ощущениями — это всегда построение информационной модели. Понимание этого феномена — важная часть информационного мировоззрения. Мы неоднократно будем использовать модельный подход в нашем учебнике, одновременно обогащая ваш арсенал компьютерных технологий по созданию и обработке информационных объектов разной природы. А сейчас напомним основные определения.

Моделированием называется замена одного объекта, процесса или явления другим объектом или процессом, сохраняющим существенные свойства исходного объекта, процесса или явления, с целью изучения этих свойств, прогнозирования возможных изменений и на этой основе использования их в деятельности человека. Сам заменяющий объект или процесс называется **моделью**.

Модель, представляющая объект, процесс или явление формализованным описанием параметров и связей между ними, называется **информационной**.

Построение информационной модели проходит несколько этапов:

- выделение факторов, существенных для достижения той цели, для которой производится построение модели;
- определение средств реализации модели, т. е. языка и исполнителя;
- описание факторов с помощью параметров (формализация);
- установление связей между параметрами;
- проверка полученной модели на адекватность.

Адекватность модели означает, что цель моделирования достигнута и данная модель при решении с ее помощью задач определенного класса дает ответ, удовлетворительный с точки зрения поставленной цели. В частности, адекватность модели означает, что при ее построении действительно были учтены все существенные факторы.

Изучая информатику в 10 классе, вы осваивали построение информационных моделей самого разного вида. Это были модели математические и фактографические, модели статические и динамические, детерминированные и вероятностные, модели объектов и модели процессов. А сейчас вы должны будете применить ваши знания о моделировании к решению задачи, в которой требуется построить модель объекта, оптимизирующего протекание некоторого процесса. Вот эта задача.

Зима. Вы решили во дворе построить снежную горку. С нее хочется скатываться на санях так, чтобы дух захватывало, но все же было безопасно. Какой формы должна быть такая горка?

Цель моделирования сформулирована в самом условии задачи: определить оптимальную форму горки. Два фактора — безопасность и чтобы дух захватывало — также названы в условии. К существенным факторам, влияющим на построение модели, конечно, следует отнести и ограничения на размеры, зависящие, в частности, от размеров участка, на котором будет строиться горка. Пусть длина нашей горки будет ограничена 6 метрами. Условие безопасности диктует ограничение на скорость, которая допустима в конце спуска. Обозначим эту скорость буквой v. Пусть она будет, скажем, 20 км/ч. Пусть h — высота горки. Хорошо известный из физики закон сохранения энергии показывает, что $\frac{mv^2}{2} = mgh$, откуда $h = \frac{v^2}{2g}$. Подставив в эту формулу значение скорости v (не забыв перевести его из км/ч в м/с) и значение g, приближенно равное 9,8 м/с^2, получаем $h \approx 1{,}57$ м. Для простоты будем строить горку высотой 1,5 м. А дух захватывать будет, если спуск происходит как можно быстрее. Иными словами, мы хотим, чтобы спуск осуществлялся за наименьшее время. Тем самым мы провели формализацию, описав существенные факторы посредством нескольких числовых параметров.

Что же можно варьировать для уменьшения времени спуска? Форму горки. Обычно горку делают в форме прямоугольного треугольника (рис. 1.4). А если сначала спуск сделать более крутым, а затем пологим (рис. 1.5)? Как же найти подходящую линию?

Изучая информатику, вы уже знакомились с приемом, который называется **дискретизацией**. Он заключается в том, что промежуток изменения величины разбивается на несколько частей и на каждой из этих частей изучаемая функция, зависящая от этой величины, заменяется линейной функцией (или даже константой). Иными словами, вместо плавной линии, изображенной на рисунке 1.5, мы рассматриваем ломаную (рис. 1.6). При увеличении количества частей, на которые разбивается промежуток изменения исходной величины, получающаяся ломаная все более приближается к искомой плавной линии.

Приступим теперь к установлению связей между параметрами. Прежде всего введем систему координат так, как показано на ри-

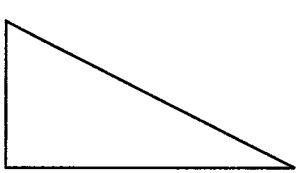


Рис. 1.4

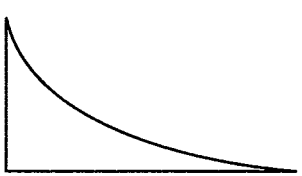


Рис. 1.5

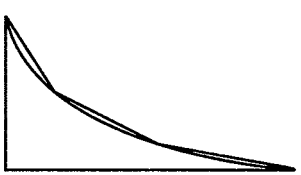


Рис. 1.6

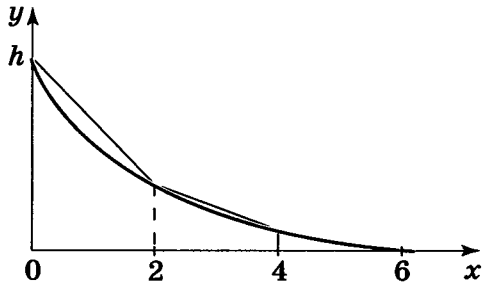


Рис. 1.7

сунке 1.7. Для осуществления дискретизации нам требуется знать количество точек, которыми разбивается отрезок [0; 6], изображающий основание будущей горки. Обозначим это количество буквой n (на рисунке 1.7 количество точек разбиения равно 2). Тогда количество частей, на которые разобьется отрезок, равно $n + 1$.

Обозначим высоту горки в i-й точке разбиения через y_i (левому концу отрезка присвоим номер 0, тогда правый конец отрезка будет иметь номер n, поэтому высота на левом конце отрезка обозначается как y_0, а на правом — как y_{n+1}). Упоминавшийся выше закон сохранения энергии показывает, что скорость v_i, которую будут иметь сани в i-й точке горки, удовлетворяет соотношению $\frac{mv_i^2}{2} = mg(h - y_i)$. Следовательно, $v_i = \sqrt{2g(h - y_i)}$. Поскольку движение по отрезку прямой от одной точки разбиения до другой происходит под действием постоянной силы — силы тяжести, это движение является равноускоренным. А для равноускоренного движения время, затрачиваемое телом на прохождение заданного пути s, равно частному при делении пути s на полусумму скоростей, которые имеет это тело в начале и конце пути. Путь s_i, проходимый санками на i-м участке горки, легко найти по теореме Пифагора; он равен $\sqrt{(y_{i-1} - y_i)^2 + (6/(n+1))^2}$, где $y_0 = h = 1{,}5$, а $y_{n+1} = 0$. Следовательно, время t_i, которое тратится санями на прохождение i-го участка горки, равно $\frac{2s_i}{v_{i-1} + v_i}$, где $v_0 = 0$. Общее время спуска равно, разумеется, сумме t_1, t_2, ..., t_n. Значит, наша цель — найти такие значения для y_1, y_2, ..., y_n, чтобы указанная сумма была наименьшей. Модель построена.

Представим кратко процесс построения этой модели в виде таблицы 1.2.

В дополнительных комментариях, по-видимому, нуждаются только два пункта: выбор исполнителя и проверка адекватности. Мы предлагаем для реализации модели использовать электронную таблицу, ведь построенная нами модель состоит из математических соотношений, связанных посредством тех или иных формул. Впрочем, желающие могут использовать для этих целей любой язык программирования.

Таблица 1.2

Этапы построения модели	Построение модели
Определение цели	Определить оптимальную форму горки
Выделение факторов, существенных для достижения этой цели	Безопасность, чтобы захватывало дух, ограниченность размеров
Определение средств реализации модели	Язык математики, электронная таблица
Описание факторов с помощью параметров	Высота, длина основания, количество точек разбиения
Установление связей между параметрами	Использование законов равноускоренного движения, сохранения энергии
Проверка полученной модели на адекватность	В компьютерном эксперименте

Что касается проверки адекватности, то ее можно было бы производить, строя горки разной конфигурации. Однако до реальной зимы еще далеко, да и много раз повторять натурный эксперимент вам едва ли захочется. Компьютер позволит сэкономить силы и время. Так что проверку адекватности вы будете осуществлять в ходе выполнения лабораторной работы № 1.

Свертывание информации тоже можно рассматривать как процесс построения модели. Убедиться в этом позволяет таблица 1.3.

Если вы внимательно присмотритесь к любой интеллектуальной деятельности человека, то обязательно обнаружите в ней присутствие моделирования в качестве одной из главных составляющих. К сожалению, такой важный компонент информационной культуры, как умение полноценного моделирования, еще не стал достоянием каждого грамотного человека. Довольно стандартными ошибками при решении жизненных задач являются нечеткая формулировка цели, или неуказание существенных факторов, или отсутствие проверки на адекватность. Мы надеемся, что в вашей деятельности такие ошибки будут встречаться все реже.

Таблица 1.3

Этапы построения модели	Построение модели
Определение цели	Представить имеющуюся информацию более компактно
Выделение факторов, существенных для достижения этой цели	Сохранение основного смысла
Определение средств реализации модели	Текстовая или графическая форма; исполнитель — человек
Описание факторов с помощью параметров	Выделение ключевых слов и фраз
Установление связей между параметрами	Установление связей между различными содержательными фрагментами; в результате получается резюме, или кластер, или какое-либо еще свернутое представление информации — модель построена
Проверка полученной модели на адекватность	Анализ получившегося информационного продукта на сохранение в нем основного содержания исходной информации

Вопросы и задания

1. Какую модель называют информационной?
2. Каковы этапы построения информационной модели?
3. Что понимается под адекватностью модели?
4. Какими могут быть причины, по которым построенная модель будет признана неадекватной? (С о в е т. Готовя ответ на этот вопрос, полезно перечитать § 5 из учебника для 10 класса.)
5. а) В чем состоит метод дискретизации?
   б)* Приведите примеры, где используется метод дискретизации.
6. Паспорт — это одна из информационных моделей человека.
   а) Какова, на ваш взгляд, в данном случае цель моделирования?

б) Какие факторы признаны существенными в этой модели? Каковы параметры, посредством которых эти факторы формализованы?

в)* Чем характеризуется адекватность этой модели? Каким образом поддерживается эта адекватность?

7. Вы собираетесь на день рождения к другу и хотите купить ему подарок.
а) Какие факторы, на ваш взгляд, существенны для решения этой жизненной задачи?
б) Попытайтесь описать выделенные вами факторы с помощью формальных параметров. Для всех ли факторов удалось это сделать?

8.* Приведите несколько примеров жизненных задач, для которых при построении модели существенными являются неформализуемые факторы.

9. Перечитайте еще раз задание 6 из § 5 и с позиций модельного подхода объясните различия в выборе ключевых слов.

10. Какие компоненты информационной культуры личности проявляют себя при моделировании?

11. а) Как бы вы объяснили с позиций моделирования фразу «Хотели как лучше, а получилось как всегда»?
б) Как бы вы оценили с позиций моделирования предложение «Давайте сначала ввяжемся в драку, а потом посмотрим, что из этого получится»?

§ 7 Информационные модели в задачах управления

Вспомните: одним из компонентов информационной культуры человека является умение применять полученную информацию для принятия решений (см. § 1). Принимать решения — это фактически определять управление техническими системами или некоторым людским коллективом, а нередко и самим собой (возможно, в составе того или иного коллектива).

Понятие управления мы достаточно подробно обсуждали в главе 5 учебника для 10 класса. Напомним, что **управлением** называется воздействие на объект или процесс, имеющее своей целью получение требуемых значений параметров этого объекта или процесса. Мы надеемся, что никого не смутит то обстоятельство, что под словом «объект» может подразумеваться человек (например, шофер, подчиняющийся сигналам сотрудника ГИБДД), техническое устройство (например, автомобиль), организация (например, школа).

Физически управляющее воздействие может быть реализовано по-разному. Управление телевизором его владелец осуществляет на-

жатием кнопок и вращением ручек настройки, светофор регулирует движение автотранспорта посредством цветовых сигналов с заранее оговоренным смыслом каждого из них, управление собакой хозяин осуществляет голосовыми командами. Эти примеры показывают, что, говоря об управлении вообще, необходимо отвлечься от физической формы управляющего воздействия. Тогда становится ясно, что суть управляющего воздействия — это передача информации объекту управления.

Управляющая информация может быть представлена инструкцией, что должен делать объект управления. Если объект управления — формальный исполнитель, то такая инструкция, как вы знаете, обычно представляет собой алгоритм. Алгоритмическое управление формальным исполнителем является одним из важнейших вариантов реализации управления, особенно часто встречающимся в технических системах. Но в живой природе и особенно в обществе далеко не всякое управление сводится к алгоритмическому; существуют и другие формы управления. О некоторых из них мы рассказывали в учебнике 10 класса; тому, кто забыл этот материал, мы советуем перечитать главу 5 указанного учебника.

В управлении социальными процессами существенную роль играет планирование. Оно далеко не всегда выступает в форме конкретного алгоритма, а обычно представляет собой перечень стадий, которые предстоит пройти, или список задач, которые нужно решить, чтобы достичь поставленной цели. В настоящее время такое планирование нередко осуществляется в форме разработки тех или иных проектов. В современных справочниках **проект** определяется как целенаправленная деятельность временного характера, предназначенная для создания уникального продукта или услуги с заранее определенным уровнем качества и выделенными для этого ресурсами. Осуществление полета на Марс, строительство нового завода, ремонт квартиры — все это примеры возможных проектов. Проектами, а не маниловщиной они будут в том случае, если выполнен ряд условий, отличающих проекты от других видов деятельности. Вот эти условия:

- направленность на достижение конкретной цели;
- ограниченная протяженность во времени, точное определение сроков начала и завершения проекта;
- точное определение ресурсов, выделяемых на осуществление проекта;
- наличие скоординированного плана выполнения проекта.

Как правило, проекты обладают еще одной чертой — они уникальны. Их уникальность может быть вызвана характером решаемой задачи (не каждый день совершаются полеты на Марс), особенностями условий осуществления (например, спецификой используемых ресурсов), индивидуальными требованиями заказчика (скажем, при ремонте квартиры) и т. д.

Необходимость создания систем управления проектами была осознана в середине прошлого века. В США был даже создан Институт управления проектами. Как это нередко случалось в те времена, первые шаги в создании систем управления проектами были осуществлены на военном поприще. В 60-е годы XX века был создан метод анализа и оценки программ PERT (Program Evaluation and Review Technique). Данный метод был разработан корпорацией «Локхид» и консалтинговой фирмой «Буз, Аллен энд Гамильтон» для реализации проекта разработки ракетной системы «Поларис». В проекте участвовало около 3800 основных подрядчиков. Использование метода PERT позволило точно определить, что требуется делать в каждый момент времени и кто именно должен это делать, а также вероятность своевременного завершения отдельных операций. Руководство программой оказалось настолько успешным, что проект удалось завершить на два года раньше запланированного срока. Благодаря такому успешному началу данный метод управления вскоре стал использоваться для планирования проектов во всех вооруженных силах США, а позже и больших гражданских проектов.

Этап бурного развития систем для управления проектами начался с того момента, когда персональный компьютер стал рабочим инструментом для широкого круга руководителей. Управленческие системы нового поколения разрабатываются как средства управления проектом, понятные любому менеджеру, не требующие специальной подготовки и обеспечивающие легкое и быстрое включение в работу. В настоящее время для разработки проектов и управления ими широко используются такие программные продукты, как Open Plan (фирмы Welcom Software Technology) и Microsoft Project. Второй из этих продуктов удобен пользователю, в частности, тем, что его интерфейс во многом схож с интерфейсом привычных программ из Microsoft Office. Тем не менее в стандартный комплект Microsoft Office эта программа не входит, поэтому мы не станем ее рассматривать в данном учебнике.

Разработке конкретного плана нередко предшествует определение стратегии в решении той или иной проблемы или организации процесса по преобразованию имеющейся ситуации. В стратегии, как правило, указываются основные принципы, на основании которых будет разрабатываться план, указываются основные пути и средства для его реализации. Подробнее понятие стратегии мы будем обсуждать в главе 7.

А пока вернемся к обсуждению компонентов, составляющих основу управления теми или иными объектами. Телевизор как объект управления вместе с допустимыми для него воздействиями один, а управлять им могут разные люди, причем с совершенно различными целями: одному хочется смотреть фильм, а другому — спортивную программу; для одного звук слишком громкий, а для другого, наоборот, тихий. Та или иная социальная группа (например, учащиеся одного класса) как объект управления тоже подвер-

жена воздействию разных управляющих объектов (в том числе и находящихся внутри этой группы) с весьма разными целевыми установками. Экономический регион может подвергаться управляющему воздействию со стороны также различных объектов. Отвлекаясь опять-таки от физической реализации объекта управления, мы видим, что каждый объект управления характеризуется *своими функциональными возможностями* (т. е. тем, что он может сделать или в какое состояние перейти) *и допустимыми воздействиями на него.*

Допустимые воздействия для каждого объекта, как правило, предопределены самой его природой. Можно сказать, что воздействие на объект управления — это восприятие им информации через специальные входы. Реакция объекта на входную информацию, т. е. изменение его состояния, тоже описывается соответствующей информацией, которая подается на тот или иной выход — именно благодаря информации на выходах можно судить о том, достигнута ли поставленная цель. Поэтому с информационной точки зрения схему управления естественно изобразить так, как показано на рисунке 1.8. При этом для нас оказывается несущественным внутреннее устройство объекта управления.

Рассмотрим, к примеру, действия человека, переходящего улицу на перекрестке. Он проверит наличие светофора, и если таковой имеется, то согласует свои действия с его сигналами, выберет необходимую скорость перемещения и, наконец, перейдет улицу. Важно ли для нас в этом случае устройство органов зрения, слуха, опорно-двигательного аппарата и т. п.? Конечно, нет. Описывая переход улицы человеком, мы интересуемся только информацией, поступающей из внешней среды, т. е. подаваемой на входы, и действиями человека в соответствии с этой информацией, т. е. результатами на выходах.

Или другой пример, относящийся уже не к живой природе, а техническим устройствам. Пусть перед нами автомат по продаже билетов на пригородные поезда. Мы опускаем в него монеты на нужную сумму (можно считать, что подаем ему на входы информацию), а он на выходе дает нам билет или сообщает, что сумма денег не соответствует стоимости билета. Интересует ли нас при этом, как он устроен внутри? Конечно, нет. Лишь бы он правильно работал.

Как видите, мы отказываемся от изучения внутренней структуры объекта. Объект, внутреннее устройство которого принципи-

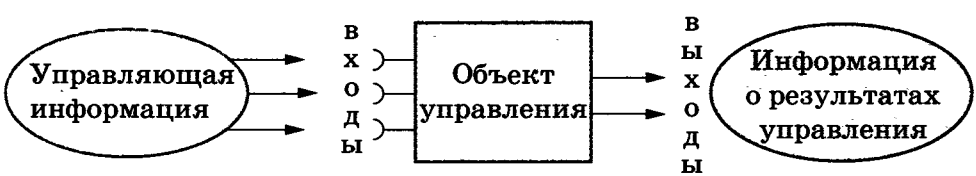


Рис. 1.8 Схема управления

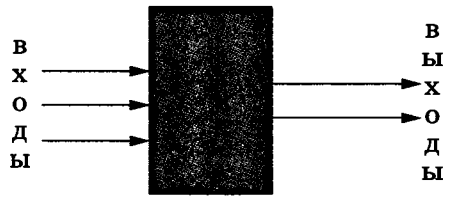


Рис. 1.9 Черный ящик

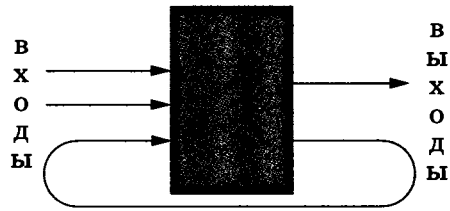


Рис. 1.10

ально скрыто от исследователя, был введен в кибернетике под названием **черный ящик**. Схематично черный ящик можно представлять себе так, как он изображен на рисунке 1.9.

Не зная, как устроен черный ящик, мы можем лишь предполагать, какую информацию он воспринимает на входах (иными словами, что для него существенно) и какой будет его реакция на те или иные входные сигналы. Догадку можно проверить, подавая на входы ту или иную информацию и наблюдая на выходах за реакцией черного ящика на эту информацию. Если наша догадка будет регулярно подтверждаться, то можно считать, что мы построили модель того объекта, который представлен данным черным ящиком.

Важной особенностью живых и социальных систем является свойство саморегуляции. Именно это свойство позволяет природной системе восстанавливать то состояние, из которого ее вывели тем или иным воздействием. Иными словами, возможность саморегуляции — залог устойчивого существования системы. Это возможно только в том случае, когда *состояние, в котором находится система, само является фактором, воздействующим на систему.* Если представлять управляемый объект как систему со входами и выходами, то наличие саморегуляции означает, что информация с некоторых выходов поступает на его входы (рис. 1.10). Напомним, что воздействие выходных параметров системы на ее же входные параметры называют **обратной связью.**

При создании управляемой системы ее устойчивость обеспечивается наличием обратной связи между управляемым и управляющим (регулирующим) объектами такой системы. Управление, использующее обратную связь между управляемым и управляющим объектами, называется **управлением по принципу обратной связи.** Если в такой системе управляемый объект и объект-регулятор изобразить черными ящиками, то мы можем представить ее схемой, изображенной на рисунке 1.11.

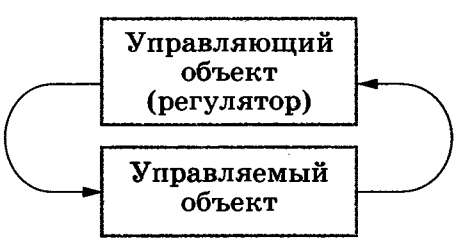


Рис. 1.11

Вопросы и задания

1. Что такое управление объектом или процессом?
2. Что такое проект? Приведите примеры проектов из разных областей человеческой деятельности.
3. Какими основными чертами отличается проект от иной деятельности людей?
4. Ниже приведено несколько ситуаций, в которых осуществляется управление. Укажите, в чем состоит процесс управления: попытайтесь определить цель управления, управляющий и управляемый объекты, допустимые воздействия на управляемый объект.
   а) Движение поезда метро.
   б) Объявление начала посадки в самолет.
   в) Ловля рыбы на удочку.
   г)*Обработка детали на фрезерном станке.
   д)*Лечение пациента.
5. Среди процессов, перечисленных ниже, укажите процессы управления.
   а) Сталевар разливает металл в ковши.
   б) Следователь допрашивает подозреваемого.
   в) Кандидат в депутаты выступает на митинге с призывом голосовать за него.
   г) Художник рисует пейзаж.
   д) Режиссер снимает фильм.
6. Для каждого из следующих объектов управления укажите, что для него является допустимыми воздействиями:
   а) утюг;
   б) автомат по продаже железнодорожных билетов;
   в) продавец мороженого;
   г)*ваш класс.
7. При каких условиях возможно явление саморегуляции? Приведите примеры саморегулирующихся систем.
8. Что такое обратная связь?
9. Среди приведенных ниже примеров взаимодействия укажите те, которые относятся к понятию обратной связи:
   а) раскрытие (или закрытие) зонтика и начало (или прекращение) дождя;
   б) изменение доходности производства и объема выпускаемой продукции;
   в) изменение яркости горения лампочки и подаваемого напряжения;

г) выступление на митинге кандидата в депутаты с призывом голосовать за него;

д) выступление по телевизору кандидата в депутаты с призывом голосовать за него;

е) изменение покупательского спроса и цены на товар.

10 Какое управление называется управлением по принципу обратной связи?

В заданиях 11—13 приведены примеры некоторых черных ящиков. Каждый ящик описывается его реакциями на информацию, подаваемую на его входы. По этим данным требуется определить, что делает данный ящик. На входы могут подаваться последовательности символов, которые могут восприниматься как последовательности любых символов, либо как последовательности букв русского алфавита, либо как натуральные числа, — информацию, представленную другим способом, данные ящики не воспринимают.

11 В этом задании черные ящики имеют один вход и один выход.

а)

№ опыта	Информация на входе	Результат
1	шалаш	шалаш
2	159	951
3	(№%?	?%№(
4	alfa	afla
5	конец	ценок

б)

№ опыта	Информация на входе	Результат
1	поп	оно
2	125	не понимаю
3	аист	язрс
4	(№%?	не понимаю
5	весна	бдрмя

⑫ В этом задании черные ящики имеют два входа и один выход.

а)

№ опыта	Информация на входах		Результат
	1-й вход	2-й вход	
1	раз	два	не понимаю
2	127	148	127
3	364	5423	364
4	2781	658	658
5	465	465	465

б)*

№ опыта	Информация на входах		Результат
	1-й вход	2-й вход	
1	шалаш	физика	не понимаю
2	физика	4	не понимаю
3	4	12	8
4	7	25	16
5	37	12	12
6	25	42	25

⑬ В этом задании черные ящики имеют один вход и два выхода.

а)

№ опыта	Информация на входе	Результат	
		1-й выход	2-й выход
1	шалаш	2	3
2	крокодил	3	5
3	1268	не понимаю	
4	аист	2	2
5	восторг	2	5

б)*

№ опыта	Информация на входе	Результат	
		1-й выход	2-й выход
1	крокодил	не понимаю	
2	123	3	1
3	252	2	2
4	256	2	8
5	49	7	2
6	1	не могу	

§ 8 Модель экономической задачи

Владелец небольшого городского кинотеатра хочет знать, какую цену надо установить на билеты, чтобы выручка была максимальной.

Дирекция авиакомпании хочет установить цену на билеты для пассажиров так, чтобы доход компании был максимальным.

Автопредприятие предоставляет в аренду автобусы. Его руководство хочет установить плату за 1 час аренды так, чтобы доход был максимальным.

Во всех трех случаях ответ на первый взгляд представляется очевидным: чем выше цена (или арендная плата), тем и доход выше. Однако, если назначить слишком высокую цену, то кинозал будет пустым, самолеты будут совершать порожние рейсы, а автобусы простаивать на автобазе. Вместо доходов одни убытки. Если поразмышлять еще немного, то станет ясно, что все три жизненные ситуации описываются той самой моделью управления с обратной связью, которую мы рассмотрели в предыдущем параграфе. В качестве управляющего воздействия выступает цена, а объектом управления будет количество посетителей кинотеатра, или число пассажиров на данный рейс, или количество заказанных автобусов на данное время. Если значение цены обозначить буквой x, то количество проданных билетов или заказанных автобусов будет некоторой функцией $f(x)$ от переменной x. Мы эту функцию не знаем, так что она для нас выступает как черный ящик, и, следовательно, модель рассматриваемой ситуации может быть представлена

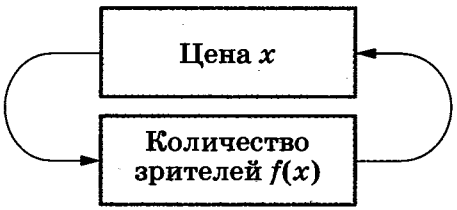


Рис. 1.12

так, как изображено на рисунке 1.12. Обратите внимание: рассматриваемые жизненные задачи касаются, казалось бы, совсем разных областей хозяйствования, а модель получилась одна. Это еще одно важное свойство моделирования — универсальность построенных моделей: одна и та же модель может применяться для решения разных задач.

Как в нашей модели описывается результат через параметры x и $f(x)$? Доход D равен произведению x на $f(x)$. Цель: подобрать такое значение x, при котором функция $D = xf(x)$ принимала бы максимальное значение.

Такая задача была бы совсем несложной, если бы нам была известна формула для функции $f(x)$. Но такую формулу нам никто не предоставит. Однако любой опытный предприниматель, прежде чем предложить свой товар, обязательно изучит спрос на него и, в частности, то, за какую цену готов потребитель покупать его товар (такое исследование называется маркетинговым). Мы взяли у владельца некоторого маленького кинотеатра — всего на 100 мест — отчет о подобном исследовании (таблица 1.4).

Выше 300 рублей владелец цену поднимать не стал — понял, что никто вообще к нему не придет. А от себя мы добавили точку $x = 0$; ясно, что если денег не брать совсем, то зал будет заполнен до отказа, т. е. $f(0) = 100$.

Воспользовавшись электронной таблицей, мы построили график предполагаемой функции $f(x)$. Он представлен на рисунке 1.13.

Такой вид линии вполне объясним: пока цена невелика, количество зрителей убывает незначительно, но потом каждые 10 рублей уже ощутимы для потребителя, и количество посетителей кинотеатра резко убывает. Что касается формы кривой, то даже девятикласснику видно: получившаяся линия очень похожа на параболу. При этом ее вершина, очевидно, приходится на значение $x = 0$, ветви направлены вниз, так что коэффициент при x^2 отри-

Таблица 1.4

Цена, р.	50	80	110	140	170	200	230	260	290
Количество посетителей, чел.	98	94	88	80	71	60	47	32	16

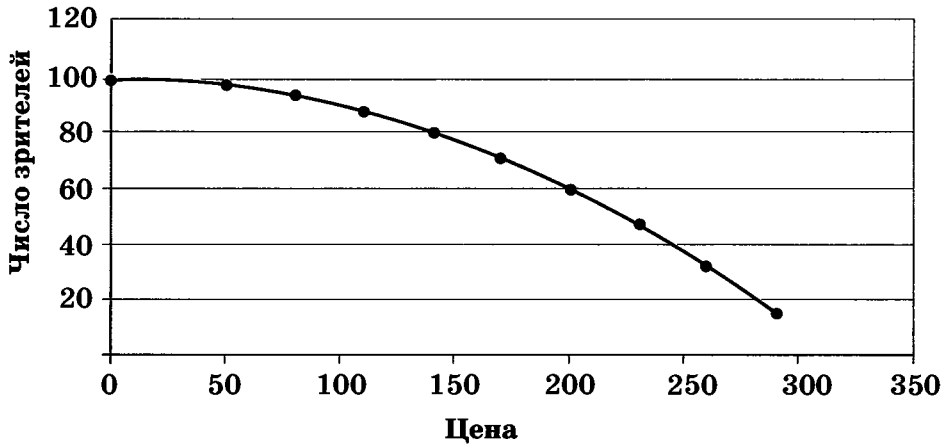


Рис. 1.13 График посещаемости кинотеатра в зависимости от цены билетов

цателен. Следовательно, можно считать, что функция $f(x)$ имеет вид $c - ax^2$, где оба коэффициента a и c положительны. Сами коэффициенты придется, конечно, подбирать так, чтобы график функции как можно ближе подходил к экспериментальным значениям. Впрочем, коэффициент c равен значению функции $f(x)$ при $x = 0$, так что его и искать не надо. Как искать коэффициент a, мы обсудим при выполнении лабораторной работы № 2.

Остается найти неотрицательное значение переменной x, при котором функция $D = x(c - ax^2)$ имеет наибольшее значение. У тех, кто дружен с математикой, уже, наверно, руки чешутся исследовать эту функцию известными им математическими методами. Что ж, пусть попробуют; их ждет удивительный результат. А мы отложим это исследование до выполнения лабораторной работы № 2.

Вопросы и задания

1. При каких условиях, на ваш взгляд, одна и та же модель годится для решения различных задач?
2. Запишите этапы построения модели экономической задачи в форме таблицы по образцу таблиц 1.2 и 1.3.
3. Каков, по вашему мнению, смысл коэффициента a, фигурирующего в формуле для $f(x)$, с экономической точки зрения?

§ 9 Международные исследования PISA

В конце 2005 года в Тунисе состоялся Международный форум по проблемам информационного общества. Одной из главных на этом форуме была признана проблема информационного неравенства между передовыми и слаборазвитыми странами. В докладе Генерального секретаря ООН Кофи Аннана подчеркивалось, что информационное неравенство становится одной из глобальных проблем, для решения которой потребуются совместные усилия многих государств. Эта проблема не впервые прозвучала на международном собрании такого уровня, но прежде она рассматривалась преимущественно как инструментально-технологическая, связанная с обеспечением доступа к современным средствам информатики и информационных технологий. Теперь же внимание было сосредоточено на неравенстве в готовности и умении людей работать с информацией, применяя современные достижения информационной техники. Потребовался инструмент, позволяющий объективно сравнивать такие умения и готовность. В значительной степени роль такого инструмента исполняют исследования PISA — Program for International Student Assessment. Это исследование направлено на получение данных для сравнительной оценки функциональной грамотности 15-летних учащихся — в большинстве стран именно к этому возрасту заканчивается базовое обучение школьников.

Функциональная грамотность — более широкое понятие, чем грамотность информационная. В нее включены также языковая грамотность, компьютерная, правовая, гражданская, экологическая. Особое место в функциональной грамотности занимает **деятельностная грамотность**, т.е. не просто наличие академических знаний, а умение применять их в практической деятельности. Это способность ставить и изменять цели и задачи собственной деятельности, осуществлять различные виды взаимодействия в группе и обществе, действовать в ситуации неопределенности. Ключевой вопрос исследования: «Обладают ли учащиеся 15-летнего возраста, получившие обязательное образование, знаниями и умениями, необходимыми им для полноценного функционирования в обществе?» Поэтому задания PISA направлены не на определение уровня освоения школьных программ, а на оценку способности применять полученные в школе знания и умения в жизненных ситуациях.

Эти исследования проводятся один раз в три года. Россия начала участвовать в них с 2000 года. Ведь сегодня информационная сфера общества становится главной ареной конкурентной борьбы, а значит, уровень информационной грамотности населения впрямую относится к вопросам национальной безопасности. Поэтому важно знать объективную оценку своего положения среди других стран. К сожалению, результаты здесь не так уж радужны —

по многим параметрам (в том числе по информационной культуре) наша страна оказывается в третьем десятке.

Ниже мы приводим два задания из исследований PISA 2000 и 2003 годов, относящихся к информационной грамотности. Попытайтесь выполнить эти задания.

Задание 1. Озеро Чад

На рисунке 1.14 показано изменение уровня глубины озера Чад в североафриканской части пустыни Сахара. Озеро Чад полностью исчезло примерно 20 000 лет до н. э. в течение последнего ледникового периода. Примерно 11 000 лет до н. э. оно появилось вновь. Сегодня уровень его глубины примерно такой же, каким он был в 1000 году н. э.

На диаграмме, представленной на рисунке 1.15, показаны наскальное искусство в Сахаре (древние рисунки или живопись, найденные на стенах пещер) и изменения в животном мире.

Воспользуйтесь информацией об озере Чад, представленной на рисунках 1.14 и 1.15, при ответе на вопросы:

1. Какова глубина озера Чад в наше время?
2. Определите, какой примерно год соответствует начальной точке графика на рисунке 1.14?

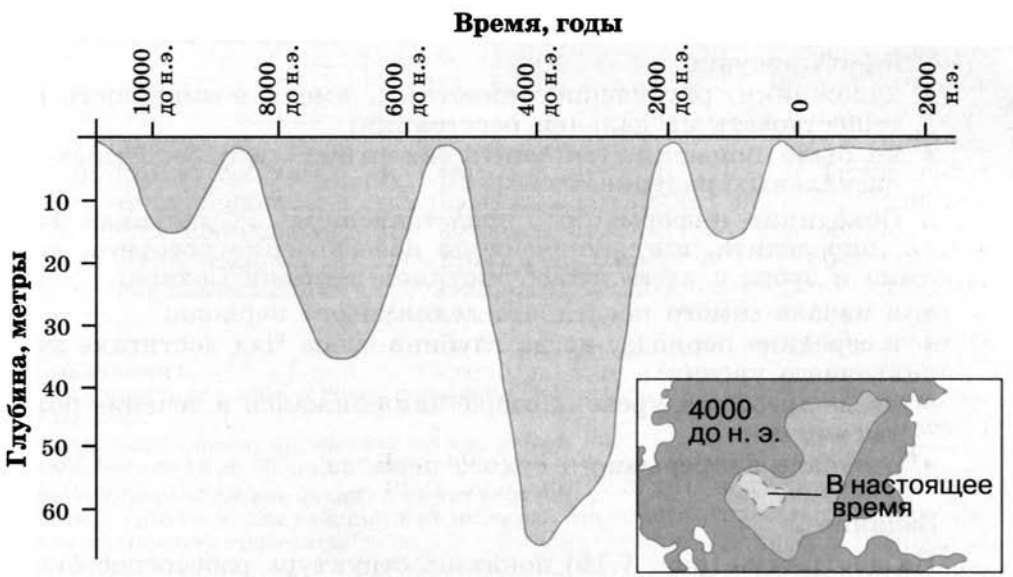


Рис. 1.14 Изменение уровня глубины озера Чад

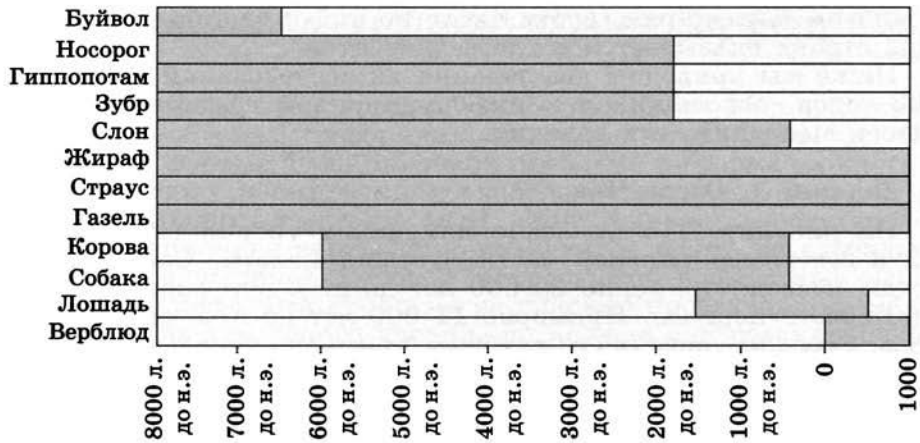


Рис. 1.15 Наскальные рисунки, найденные в Сахаре

3. Почему выбран именно этот год в качестве начальной точки на графике?

4. Диаграмма на рисунке 1.15 основана на предположениях о том, что:

- **животные, изображенные на наскальных рисунках, обитали в районе озера Чад в то время, когда их рисовали;**
- **художники, рисовавшие животных, обладали высокой техникой рисунка;**
- **художники, рисовавшие животных, имели возможность путешествовать на дальние расстояния;**
- **не было попытки приручить животных, изображенных на наскальных рисунках.**

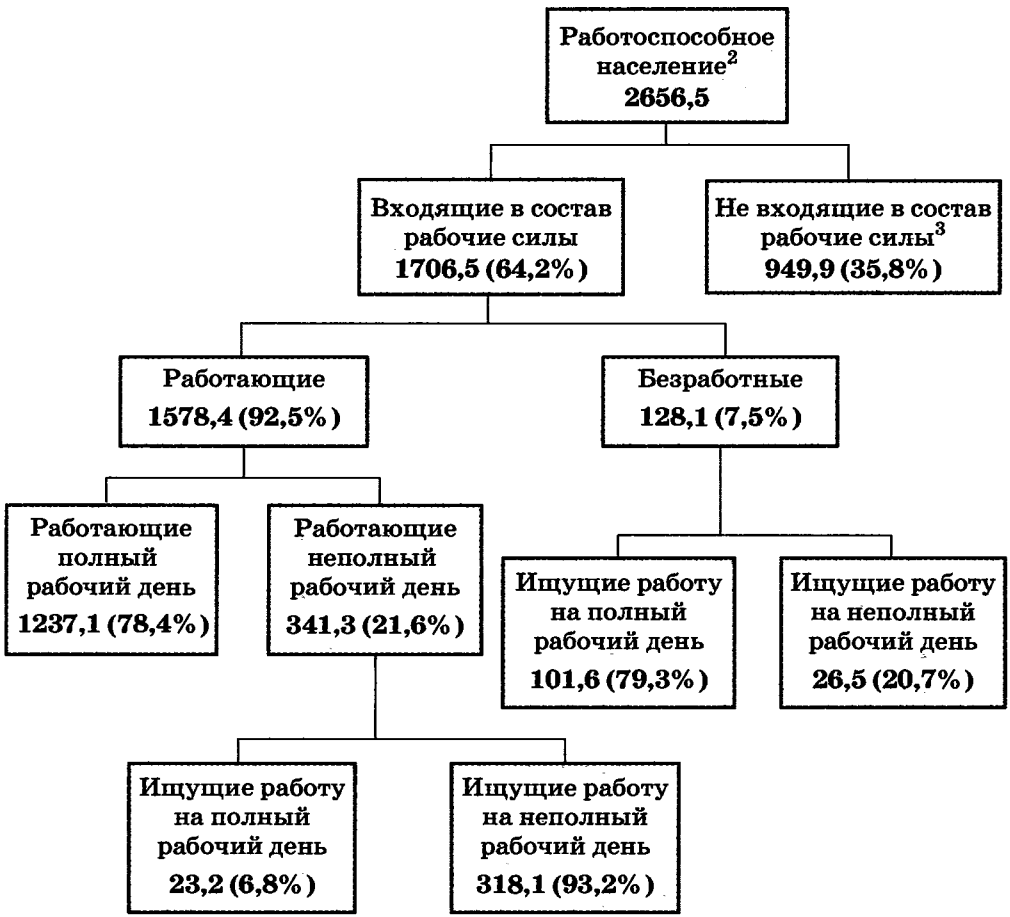
5. Объединив информацию, представленную на рисунках 1.14 и 1.15, определите, когда произошло исчезновение носорога, гиппопотама и зубра с наскальных рисунков пустыни Сахара:

- **в начале самого последнего ледникового периода;**
- **в середине периода, когда глубина озера Чад достигала наивысшего уровня;**
- **после того, как уровень озера Чад снижался в течение более тысячи лет;**
- **в начале непрерывного сухого периода.**

Задание 2

На диаграмме (рис. 1.16) показана структура работоспособного населения в некоторой стране. Численность всего населения этой страны в 1995 году была примерно 3,4 млн.

Структура рабочей силы (в тысячах) к концу 31 марта 1995 года[1]



Примечание:
1. Численность различных групп населения указана в тысячах.
2. К работоспособному населению относят людей в возрасте от 15 до 65 лет.
3. Население, которое не входит в состав рабочей силы, — это те, кто не работает и не ищет работу или кто не может работать.

Рис. 1.16

Используя информацию о рабочей силе, представленную на диаграмме, ответьте на вопросы.

1. На какие две основные группы разделено работоспособное население?

2. Какова численность работоспособного населения, которое не входит в состав рабочей силы? Укажите только число.

3. К какой группе населения можно отнести перечисленных ниже людей? Определите, есть ли среди них те, кого нельзя отнести ни к одной из групп:

- официант, 35 лет, работает неполный рабочий день;
- женщина-предприниматель, 43 года, работает 60 часов в неделю;
- студент дневного отделения, 21 год;
- мужчина, 28 лет, недавно продал свой магазин, ищет работу;
- женщина, 55 лет, никогда не работала и не хотела работать вне дома;
- 80-летняя бабушка, все еще работает несколько часов в день в семейной торговой лавке.

4. Предположим, что сведения о работоспособном населении публикуются ежегодно в форме приведенной ранее диаграммы. Определите, будут ли меняться от года к году перечисленные ниже четыре вида сведений, представленных на диаграмме:

- названия групп населения, представленных на диаграмме (например, «Входящие в состав рабочей силы»);
- проценты (например, 64,2%);
- численность различных групп населения (например, 2656,5);
- примечания, которые даны под диаграммой.

5. Информация о структуре рабочей силы дана в виде приведенной ранее диаграммы, но ее можно представить другими способами, например, с помощью словесного описания, в виде графика, таблицы или диаграммы другого вида, например круговой. Представленная ранее диаграмма выбрана потому, что она наиболее удобна для того, чтобы показать:

- изменения во времени;
- численность всего населения страны;
- категории населения, входящие в состав каждой из выделенных групп;
- численность каждой группы населения.

Вопросы и задания

1. Какой смысл на современном этапе развития общества вкладывается в понятие информационного неравенства?

2. а) Что понимается под деятельностной грамотностью?
   б) В чем, по вашему мнению, заключаются отличия деятельностной грамотности от информационной?
   в) Выделите компоненты, которые, на ваш взгляд, присущи как деятельностной, так и информационной грамотности.

3. Каковы цели международного исследования PISA? В чем вы видите заинтересованность России в участии в этих исследованиях?

4. Ниже приведены два сообщения о забастовке шахтеров. Первый текст взят из газеты консервативной партии, второй — из лейбористской газеты.
   «Сегодня началась забастовка шахтеров. Прекратилась подача угля на электро- и теплостанции. Снабжение теплом и электричеством жилых домов, больниц, школ переведено на пониженный режим в целях поддерживать в них тепло хотя бы на минимальном уровне. Если шахтеры не прекратят забастовку, жители окажутся в смертоносных объятиях холода».
   «Сегодня шахтеры прекратили работу. Они требуют повышения зарплаты, которая оставалась неизменной более 8 лет. Решение о забастовке нелегко далось шахтерам, понимающим, что без их угля в домах будет не так тепло, как к тому привыкли их жители. Но они вынуждены были пойти на забастовку ради своих семей, своих детей. Шахтеры верят, что все с пониманием отнесутся к их борьбе за свое право достойной жизни в обществе».
   а) Оцените качество информации (полнота, достоверность, актуальность, объективность) в каждом из сообщений.
   б) Попытайтесь определить, за счет чего у читателей создается разное отношение к описываемым событиям. (С о в е т. Этот пункт задания полезно выполнять не в одиночку, а небольшой группой.)
   в) Оцените эффективность выполнения пунктов *а* и *б* по следующим параметрам: полученные знания, приобретенные навыки и успешность в решении поставленной задачи.

В 10 классе мы обсуждали, что для хранения и передачи информацию кодируют символами некоторого алфавита. Более того, в технических системах, например компьютерных, обычно используется алфавит, состоящий только из двух символов (как правило, из символов 0 и 1). Могут существовать самые разные способы кодирования информации. О некоторых из них (например, об ASCII-кодировании, кодировании цвета и звука) мы рассказывали в главе 1 учебника для 10 класса. И это далеко не все варианты кодирования информации, используемые в компьютерной технике. Еще об одном способе кодирования — *числовой* информации, правда, известном человечеству задолго до изобретения компьютеров, мы начнем рассказывать в одном из сдедующих параграфов.

§ 10 Системы счисления

Система счисления — это способ записи чисел с помощью заданного набора специальных знаков, называемых цифрами. Еще в 5 классе вы узнали, что привычная нам система счисления — **позиционная** и **десятичная**. Это значит, что для записи любых чисел используется *десять* цифр (обычно 0, 1, 2, 3, 4, 5, 6, 7, 8, 9), а значение каждой цифры (ее «вклад» в обозначаемое число) определяется той *позицией*, которую цифра занимает в записи числа. Так, запись 23 означает, что это число составлено из 3 единиц и 2 десятков. Если мы поменяем позиции цифр, то получим совсем другое число — 32. Это число содержит 3 десятка и 2 единицы. «Вклад» двойки уменьшился в 10 раз, а «вклад» тройки в 10 раз возрос.

Легко понять, что цифры десятичной записи числа — это просто коэффициенты его представления в виде суммы степеней числа 10, которое называют **основанием** данной системы счисления. Например:

$$38\,054 = 3 \cdot 10^4 + 8 \cdot 10^3 + 0 \cdot 10^2 + 5 \cdot 10^1 + 4 \cdot 10^0.$$

На самом деле числа можно записывать как сумму степеней не только числа 10, но и любого другого натурального числа b, большего 1. Например:

$$38\,054_b = 3 \cdot b^4 + 8 \cdot b^3 + 0 \cdot b^2 + 5 \cdot b^1 + 4 \cdot b^0.$$

Такую **запись называют представлением данного числа в системе счисления с основанием b.** Чтобы знать, в какой системе счисления записано число, основание системы (в данном случае это b) указывают нижним индексом справа от этого числа. Легко заметить, что цифры, используемые при записи числа в системе счисления с основанием b, неотрицательны и меньше чем b.

Главное удобство позиционной нумерации состоит в том, что действия над числами в такой системе счисления выполняются поразрядно (вспомните, как вы складываете и умножаете, вычитаете и делите многозначные числа в десятичной системе). Все, что нужно знать, — это таблицы сложения и умножения для однозначных чисел и правила выполнения действий столбиком. Теперь, когда вы знаете слово «алгоритм», каждый сообразит, что упомянутые правила — это просто алгоритмы для любого исполнителя, который умеет выполнять соответствующие действия над однозначными числами.

Самая простая из всех позиционных систем счисления — **двоичная**, ведь в ней всего две цифры: 0 и 1. И значит, имеется только два однозначных числа. Поэтому в этой системе счисления очень просто выглядят таблицы сложения и умножения:

$0 + 0 = 0$	$0 \cdot 0 = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$
$0 + 1 = 1$	$0 \cdot 1 = 0$
$1 + 1 = 10$	$1 \cdot 1 = 1$

В этих равенствах лишь результат 10 выглядит удивительно. Но стоит заметить, что в двоичной системе счисления

$$10 = 1 \cdot 2^1 + 0 \cdot 2^0,$$

и все становится на свои места.

Вспомните, сколько времени ушло у вас в начальной школе, чтобы выучить таблицы сложения и умножения для однозначных чисел в столь привычной десятичной системе счисления. А в двоичной системе счисления, кроме отмеченного выше «удивительного» равенства, и учить-то нечего! Но наша система счисления — десятичная. Скорее всего, так получилось потому, что на наших руках 10 пальцев, а именно они служили той первой материальной основой, посредством которой возник счет. Ведь и сейчас маленькие дети осваивают счет на пальцах.

Вот несколько примеров выполнения действий над многозначными числами в двоичной системе счисления:

$$\begin{array}{r} 101101 \\ +\ 10111 \\ \hline 1000100 \end{array} \qquad \begin{array}{r} 110110 \\ -\ 11101 \\ \hline 11001 \end{array} \qquad \begin{array}{r} 1011 \\ \times\ 101 \\ \hline 1011 \\ 1011 \\ \hline 110111 \end{array} \qquad \begin{array}{r|l} 11011 & 1001 \\ \cline{2-2} -\ 1001 & 11 \\ \cline{1-1} \ 1001 & \\ \cline{1-1} -\ 1001 & \\ \cline{1-1} 0 & \end{array}$$

Проследим, что конкретно пришлось делать при умножении столбиком двух двоичных чисел. Умножая на 1, мы просто переписали число. Во втором множителе две единицы, поэтому первый множитель пришлось переписать дважды, предварительно сдвинув вторую запись влево на необходимое число разрядов. А затем осталось осуществить сложение.

Итак, умножение в двоичной системе счисления сводится к переписыванию одного из множителей несколько раз с необходимыми сдвигами и последующими сложениями этих копий. Можно сказать, что умножение чисел в двоичной системе счисления — это переписывание со сложением.

За простоту действий над числами в двоичной системе счисления приходится расплачиваться тем, что запись даже совсем небольших чисел в ней оказывается весьма длинной. Таблица 2.1 содержит первые 16 натуральных чисел и их представление в двоичной системе счисления.

Вопросы и задания

1. Что такое система счисления? Что называют основанием позиционной системы счисления?
2. Что называют представлением числа в позиционной системе счисления с данным основанием?
3. Сколько цифр используется в позиционной системе счисления:
   а) с основанием 2; б) с основанием 8;
   в) с основанием 10; г) с основанием 16;
   д) с основанием 256; е) с основанием b?
4. Существует ли позиционная система счисления, в которой для записи чисел используется ровно одна цифра?
5. Чем привлекательна двоичная система счисления?
6. Исполнитель умеет сравнивать в некоторой позиционной системе счисления однозначные числа. Составьте для этого исполнителя:

а) алгоритм сравнения двух двузначных чисел;

б) алгоритм сравнения двух n-значных чисел (n — заданное число);

в) алгоритм сравнения двух чисел, имеющих в своей записи одинаковое количество цифр.

7 Число 23 в некоторой позиционной системе счисления записывается как 32. Какое основание у этой системы?

8 Сравните между собой числа, записанные в двоичной системе счисления:

а) 10101 и 101010;

б) 1101 и 1001;

в) 111001 и 110111.

Для пары чисел из пункта *в* определите, на сколько одно число больше другого.

9 Пользуясь приведенными в этом параграфе таблицами сложения и умножения однозначных чисел в двоичной системе счисления, сложите и перемножьте числа 101_2 и 1101_2. Какими будут результаты этих действий в десятичной системе счисления?

Таблица 2.1

Перевод чисел из десятичной системы счисления в двоичную

Десятичная система	Двоичная система
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111
16	10000

10 а) К чему приводит увеличение в 2 раза числа, записанного в двоичной системе счисления?

б) Пользуясь правилом пункта *а* и таблицей представления чисел в двоичной системе счисления, запишите в двоичной системе счисления следующие числа: 32, 64, 128, 20, 30.

в) Пользуясь равенствами 31 = 16 + 15; 52 = 32 + 20; 75 = 64 + 11 и результатами пункта *б*, запишите в двоичной системе счисления числа 31, 52, 75.

11 Замените звездочки цифрами так, чтобы получилась верная запись в двоичной системе счисления:

а)
```
  *0*10*
+  *1*1*
--------
 ***1000
```

б)
```
 **0*1*
-  **10*
--------
   **01
```

в)
```
   **1*
 ×  ***
--------
   ****
+ ****
--------
 *****1
```

г)
```
 *0****  | ***1
-****    | ---
-------  | ***
  ****
- ****
-------
     0
```

§ 11 Перевод целых чисел из одной системы счисления в другую

Внимательное чтение § 10 могло навести вас на грустные мысли: компьютер имеет дело лишь с двоичными числами и теперь нужно обзаводиться словарем, в котором содержатся переводы чисел из одной системы счисления в другую. Как при общении с иностранцем на незнакомом языке!

В заданиях к § 10 вы могли увидеть некоторые приемы превращения десятичных чисел в двоичные, но вряд ли это удовлетворит того, кто хотел бы иметь простой и надежный способ. О некоторых способах перевода чисел из одной системы счисления в другую мы расскажем в этом параграфе.

Начнем с изложения алгоритма перевода целых чисел из десятичной системы счисления в систему счисления с основанием b. Что значит записать число в системе счисления с основанием b? Если вы ответили на вопрос 2 к § 10, то вам ясно, что запись $a_0a_1...a_{n-1}a_n$ является представлением натурального числа c в системе счисления с основанием b, если

$$c = a_0 \cdot b^n + a_1 \cdot b^{n-1} + ... + a_{n-1} \cdot b^1 + a_n \cdot b^0,$$

причем каждый коэффициент a_i неотрицателен и меньше b.

Из этого равенства видно, что последняя цифра a_n представляет собой остаток при делении числа c на b. Частное от такого деления используется для нахождения предпоследней цифры a_{n-1} — она получается как остаток при делении этого частного снова на b и т. д. Вот пример перевода десятичного числа 793 в шестеричную систему счисления:

```
793 | 6
 6  |---
---  132 | 6
 19     12 |---
 18    ---  22 | 6
---      12  18 |---
 13     ---  --  3
 12      12   4
---     ---
  1       0
```

Результат: $793_{10} = 3401_6$.

Чтобы перевести число в b-ичную систему счисления, нужно последовательно делить на b до тех пор, пока частное от деления не будет меньше b. Число в b-ичной системе записывается как последовательность остатков от деления, взятых в обратном порядке.

А как быть, если, наоборот, требуется перевести число из b-ичной системы счисления в десятичную? Указанный алгоритм годится и в этом случае. Однако деление тогда надо производить в b-ичной системе. Но деление уголком — это фактически последовательное выполнение операций умножения и вычитания, и, значит, пришлось бы выучить таблицы сложения и умножения в системе счисления с любым основанием. У кого на такое хватит сил? Нельзя ли найти другой алгоритм для решения этой задачи? Оказывается, можно. Изложим алгоритм перевода числа из b-ичной системы счисления, использующий действия той системы счисления, куда мы собираемся это число переводить. Если, например, мы переводим число в десятичную систему, то и действия будут выполняться в обычной десятичной системе. Этот алгоритм заключается в следующем.

Записываем в одной строке число, которое нужно перевести, а строкой ниже будем получать число в нужной нам системе счисления. Для этого первую цифру перепишем без изменения, а под каждой следующей цифрой будем писать число, полученное сложением этой цифры с произведением слева стоящего числа на основание системы счисления. Число под последней цифрой и будет результатом перевода.

Например, для перевода двоичного числа 100111011 в десятичную запись исполнение этого алгоритма будет выглядеть следующем образом:

1	0	0	1	1	1	0	1	1
1	2	4	9	19	39	78	157	315
	$1\cdot2+0$	$2\cdot2+0$	$4\cdot2+1$	$9\cdot2+1$	$19\cdot2+1$	$39\cdot2+0$	$78\cdot2+1$	$157\cdot2+1$

Результат: $100111011_2 = 315_{10}$.

Как видите, эти вычисления легко выполнить и устно. Называется этот алгоритм **схемой Горнера**.

Двоичное представление чисел (и информации вообще), столь удобное для электронной техники, вряд ли можно признать удобным для человека. У любого программиста — специалиста, который разрабатывает программное обеспечение для компьютеров, — зарябит в глазах от нуликов и единичек, если он попытается прочитать программу прямо в машинном коде. А такая потребность иногда возникает. Поэтому программисты решили сократить количество знаков в записи машинных чисел, использовав для этого шестнадцатеричную систему счисления.

Прежде всего заметим, что для записи чисел в шестнадцатеричной нумерации требуется 16 цифр. Первые десять цифр в ней используются те же, что и в десятичной системе счисления, а дальше обычно пользуются латинскими буквами. Как именно — показано в таблице 2.2.

Таблица 2.2 **Перевод чисел из десятичной системы счисления в двоичную и шестнадцатеричную системы счисления**

Десятичная система	Двоичная система	Шестнадцатеричная система
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Фактически таблица 2.2 — это таблица 2.1, дополненная еще одним столбцом.

Если взять уже встречавшееся нам число 315_{10}, то в шестнадцатеричной системе оно запишется как 13B. Убедиться в этом можно с помощью алгоритма, использующего последовательное деление на 16 с остатком. Но ту же запись 13B можно получить и по-другому: двоичное число 100111011 (равное десятичному 315_{10}) разбивается на четверки справа налево и каждая такая четверка заменяется шестнадцатеричной цифрой согласно приведенной выше таблице:

$$\underbrace{1}_{1}\,\underbrace{0011}_{3}\,\underbrace{1011}_{B}$$

В самой левой четверке оказалась только одна цифра, значит, к ней надо мысленно слева приписать недостающие нули. Такие четверки называются **тетрадами**.

Это правило перевода чисел из двоичной системы счисления в шестнадцатеричную справедливо для любого числа, а не только для рассмотренного нами. Легко выполнять и обратный перевод из шестнадцатеричной системы в двоичную — для этого каждую цифру надо расписать в соответствии все с той же таблицей.

Значит, шестнадцатеричная система по сути своей — это та же двоичная, но в ней каждая группа из четырех цифр заменяется всего одним символом. Такое превращение двоичного кода в шестнадцатеричный легко выполнить в уме, чего никак не скажешь о переводе в десятичную систему и обратно. Понятна теперь любовь программистов к шестнадцатеричной системе (хотя у них, как и у всех других людей, на руках десять пальцев).

Вопросы и задания

1. а) Как переходить от записи числа в шестнадцатеричной системе счисления к записи в двоичной и обратно?
   б)* Придумайте легкий (т. е. устно выполнимый) алгоритм перевода числа из восьмеричной системы счисления в двоичную и обратно. Какой частью таблицы, приведенной в этом параграфе, удобно для этого пользоваться?

2. Переведите числа 579 и 2468 в системы счисления с основанием: а) 5; б) 7; в) 9.

3. Переведите в десятичную систему счисления числа 866_9, 1287_{11}, 45297_{12}, 5423_7, 74268_9.

4. Переведите числа 19, 44, 129, 561, 1322 из десятичной системы счисления в двоичную и шестнадцатеричную.

5. Переведите из двоичной в десятичную систему счисления числа 1001, 10101, 111001, 10111101.

6. Переведите из шестнадцатеричной в десятичную систему счисления числа 25, 4F, 1A7, ABC, D1AE, FFFF.

7. Предположим, что компьютер, работающий в двоичной системе счисления, оперирует с семизначными числами. Какое максимальное число он воспринимает?
   Ответьте на тот же вопрос в случае одиннадцатиразрядного и пятнадцатиразрядного компьютера.

8.* Попробуйте объяснить:
   а) почему схема Горнера дает перевод из одной системы счисления в другую;
   б) почему алгоритм последовательного деления с остатком дает перевод из одной системы счисления в другую;
   в) алгоритм перевода из шестнадцатеричной системы в двоичную и обратно.

§ 12 Перевод дробных чисел из одной системы счисления в другую

Каждый знает, что дроби бывают обыкновенные и десятичные. Обыкновенная дробь представляет собой отношение целого числа к натуральному. Поэтому ее перевод в другую систему счисления трудности не представляет: надо отдельно перевести в новую систему счисления числитель и знаменатель, затем записать их отношение.

Запись числа десятичной дробью — это распространение позиционного принципа вправо от разряда единиц. Вспомните: при переходе на один разряд влево «вклад» цифры увеличивается в 10 раз, а при переходе на один разряд вправо уменьшается в 10 раз. Так что запись 1,38054 обозначает число

$$1 \cdot 10^0 + 3 \cdot 10^{-1} + 8 \cdot 10^{-2} + 0 \cdot 10^{-3} + 5 \cdot 10^{-4} + 4 \cdot 10^{-5}.$$

Легко понять, что и здесь вместо числа 10 можно использовать любое другое натуральное число b, большее 1. Скажем,

$$1{,}38054_b = 1 \cdot b^0 + 3 \cdot b^{-1} + 8 \cdot b^{-2} + 0 \cdot b^{-3} + 5 \cdot b^{-4} + 4 \cdot b^{-5}.$$

По аналогии с десятичными дробями будем называть такую запись дробного числа b-ичной дробью. Так же как и для целых чисел, каждая цифра, используемая в записи b-ичной дроби, должна быть меньше b.

Как же переводить десятичную дробь в b-ичную? Для того чтобы найти алгоритм, запишем b-ичную дробь $c = 0{,}a_1a_2\ldots a_n$ в виде суммы разрядных слагаемых:

$$c = a_1 \cdot b^{-1} + a_2 \cdot b^{-2} + \ldots + a_{n-1} \cdot b^{1-n} + a_n \cdot b^{-n}.$$

Из этой записи видно, что целая часть числа $bc = a_1,a_2\ldots a_n$ дает первую цифру после запятой в указанном представлении числа c. Выделив в bc дробную часть, поступим с ней точно так же — умножим на b. Таким образом мы получим еще одну цифру — a_2. И так далее.

Вот пример перевода десятичной дроби 0,36 в пятеричную систему:

$$\begin{array}{rl|l} & 0, & 36 \\ \times & & 5 \\ \hline & 1, & 80 \\ \times & & 5 \\ \hline & 4, & 00 \end{array}$$

Ответ: $0{,}14_5$.

А теперь попытаемся перевести ту же дробь в семеричную систему счисления:

$$
\begin{array}{rr|l}
\times & 0,36 & \\
 & \multicolumn{2}{r}{7} \\
\hline
\times & 2,52 & \\
 & \multicolumn{2}{r}{7} \\
\hline
\times & 3,64 & \\
 & \multicolumn{2}{r}{7} \\
\hline
\times & 4,48 & \\
 & \multicolumn{2}{r}{7} \\
\hline
\times & 3,36 & \\
 & \multicolumn{2}{r}{7} \\
\hline
\times & 2,52 & \\
 & \multicolumn{2}{r}{7} \\
\hline
 & \ldots & \ldots
\end{array}
$$

Обратите внимание: после четвертого умножения мы снова получили дробь 0,36. Это значит, что дальше процесс будет повторяться и никогда не закончится! Тем самым после перевода числа 0,36 в семеричную систему счисления получается бесконечная периодическая дробь: $0{,}23432343\ldots_7 = 0{,}(2343)_7$.

При переводе конечной b-ичной дроби в десятичную систему тоже может получиться бесконечная дробь. К примеру, запись $0{,}1_3$ представляет одну треть и, следовательно, в десятичной системе будет выглядеть как бесконечная десятичная дробь 0,33333... = 0,(3).

Как вы знаете, бесконечные дроби нередко округляют, оставляя такое количество разрядов, которое обеспечивает необходимую точность. Напомним, что в десятичной системе правило округления таково: если цифра в разряде, с которого производится округление, меньше 5, то цифра в предшествующем разряде не меняется, в противном случае она увеличивается на 1. Для b-ичной дроби правило нужно модифицировать: если цифра в разряде, с которого производится округление, меньше $b/2$, то цифра в предшествующем разряде не меняется, в противном случае она увеличивается на 1. Например, дробь $0{,}23432343\ldots_7$ при округлении до третьего разряда после запятой дает $0{,}234_7$, а при округлении до шестого разряда после запятой дает $0{,}234324_7$.

В конце § 11 мы обсудили удобный алгоритм перевода чисел из двоичной системы счисления в шестнадцатеричную и обратно. Он применяется и для дробей. Только на тетрады дробную часть двоичного числа надо разбивать, двигаясь слева направо, начав с первого после запятой разряда. Вот как с помощью этого алгорит-

ма осуществляется перевод двоичного числа 10,0111101 в шестнадцатеричную систему счисления:

$$\underset{2}{\underbrace{10}},\underset{7}{\underbrace{0111}}\ \underset{A}{\underbrace{1010}}$$

Обратите внимание: последняя тетрада оказалась неполной, и ее пришлось дополнить одним нулем справа. Ответ: $2,7A_{16}$.

Вопросы и задания

1. а) Как переходить от записи дроби в шестнадцатеричной системе счисления к записи в двоичной и обратно?
   б)* Придумайте легкий (т. е. устно выполнимый) алгоритм перевода числа из восьмеричной системы счисления в двоичную и обратно.

2. Переведите в десятичную систему счисления дробные числа $86,6_9$; $12,87_{11}$; $452,96_{12}$; $54,23_7$; $74,263_9$. Для каждого числа ответ запишите как обыкновенной дробью, так и десятичной.

3. Переведите в пятеричную систему счисления дробные числа 86,6; 12,87; 40,96; 54,23; 74,268. Результат округлите до шестого разряда после запятой.

4. Переведите из двоичной в десятичную систему счисления числа 10,01; 1010,1; 11,1001; 1011,1101; 10111,101.

5. Переведите из шестнадцатеричной в десятичную систему счисления числа 2,5; 4,F; 1A,7; A,BC; D,1AE; D1,AE; FF,FF.

6. а) Верно ли, что при переводе дробных чисел из двоичной системы счисления в десятичную всегда будет получаться конечная десятичная дробь?
   б) Ответьте на тот же вопрос при переводе в десятичную систему счисления шестнадцатеричных дробей.

7. Переведите из двоичной в шестнадцатеричную систему счисления числа 11,1001; 1011,1101; 100,1; 10,101; 10111,101.

8. Переведите из шестнадцатеричной системы счисления в двоичную числа 2,5; 4,F; 1,A7; A,BC; D,2BE; D2,BE; FF,FF.

9. Найдите в двоичной системе счисления:
   а) сумму 101,01 + 100,01;
   б) разность 1000 – 1,11;
   в) произведение 111 · 111; 101,01 · 1010,1.

§ 13 Кодовые таблицы

В § 4 учебника для 10 класса мы рассказывали, что каждый символ, используемый для представления текстовой информации в компьютере, кодируется последовательностью нулей и единиц. Первым кодом, получившим всемирное признание, был восьмибитный ASCII. Для кодирования использовались последовательности, состоящие из восьми двоичных символов. Например, нажатие клавиши, на которой написана латинская буква «A», компьютером воспринимается как ввод последовательности 01100001. Это код строчной латинской буквы «a». Если же нажать одновременно клавиши Shift и A (это означает, что вы намерены ввести заглавную латинскую букву «A»), то будет введена последовательность 01000001.

Но как же компьютер узнает, какая именно последовательность кодирует тот или иной символ? Очень просто — в памяти компьютера хранится специальная кодовая таблица, в которой для каждого символа указан его двоичный код. Для ASCII такая таблица содержит 256 символов — ведь именно столько символов можно закодировать восьмибитными последовательностями. Впрочем, история этой таблицы не так уж проста.

Как уже говорилось, в начале компьютерной эры код был семибитным. И использовался он для представления информации не только в компьютере, но и в телеграфных, телетайпных и других системах коммуникаций. Эти 128 символов составили основную часть кодовой таблицы, и в таблицу восьмибитового кодирования они вошли дополненные нулем в самом левом разряде. При этом первые 32 символа являются управляющими, а остальные — изображаемыми, т. е. отвечающими за графическое изображение букв, знаков операций, знаков препинания и т. п. В таблице 2.3 приведены коды некоторых управляющих символов. Согласно сложившейся традиции двоичные коды символов рассматриваются как целые числа и переводятся в десятичную систему счисления. Тем самым каждый символ кодовой таблицы получает порядковый номер в обычной десятичной системе счисления.

Некоторые из этих символов могут вводиться с клавиатуры. Однако, что именно вводится при нажатии той или иной клавиши, зависит от используемого программного обеспечения. В большинстве текстовых редакторов (например, в Microsoft Word) нажатие клавиши Enter передает одновременно коды 10 и 13. При этом код 10 выставляется автоматически (без нажатия клавиши), как только длина строки вводимых символов достигнет определенного предела. В некоторых редакторах (например, Блокнот) этого не происходит.

Особенно отчетливо проявляется разница между управляющими кодами 10 и 13, когда они поступают на принтер. Код 13 предписывает принтеру начать печать с начала строки. При отсутствии

Таблица 2.3 **Кодирование некоторых управляющих символов в ASCII**

Двоичный код	Десятичный код	Действие	Английское название
00000111	7	Стандартный звуковой сигнал	Beep
00001000	8	Удаление предыдущего символа	Back Space (BS)
00001010	10	Перемещает позицию печати на одну строку вниз	Line Feed (LF)
00001101	13	Перемещает позицию печати в крайнее левое положение	Carriage Return (CR)
00011010	26	Признак конца текстового файла	End Of File (EOF)
00011011	27	Отмена предыдущего ввода	Escape (Esc)

перед этим кодом кода 10 печать начнется поверх уже напечатанной строки. Если будет получен только код 10, то произойдет протяжка бумаги на одну строку, а печать будет продолжаться с той позиции, на которой закончилась печать предыдущей строки.

Отметим также, что многие программы используют код 13 как признак выбора пользователем некоторого пункта меню.

Код 27, вводимый нажатием клавиши Esc, также многофункционален. В диалоге с пользователем он нередко используется как сигнал на отмену работы (выход из приложения) или как сигнал возврата на шаг назад в цепочке операций. Однако для принтера этот символ играет совсем другую роль. Дело в том, что для организации печати принтеру нужно много команд: указать шрифт, кегль, межстрочный интервал и т. п. На все это малочисленного множества управляющих символов недостаточно. Поэтому конструкторы придумали так: символ с номером 27 указывает принтеру, что следующий за ним символ надо воспринимать как команду, а не как символ, который требуется напечатать. А тогда в распоряжении конструкторов оказывается более 200 команд!

Программисту иногда приходится пользоваться управляющими символами. Но тогда удобно, чтобы они тоже отображались на экране компьютера с помощью тех или иных графических образов, — иначе как узнать, какой символ стоит в данной позиции? Скажем,

символ с кодом 27 графически на экране компьютера изображается как ←. Однако наличие символа с таким кодом в тексте, который отправлен на принтер, может привести к непредсказуемым последствиям.

В таблице на обороте форзаца учебника приведены коды всех изображаемых символов основной части ASCII.

Судьба остальных 128 символов с номерами от 128 до 255 включительно определялась потребностью кодировать простейшие графические символы, но самое главное — потребностью кодировать символы алфавитов различных национальных языков: русского, французского, португальского и т. д. Каждая страна стала разрабатывать свои так называемые **расширения ASCII**, используя оставшиеся коды по своему усмотрению. Каждое такое расширение было связано с используемой операционной системой, а иногда напрямую с программным обеспечением, разрабатывавшимся в этой стране. Поэтому, например, для русского алфавита было создано около десятка различных расширений ASCII. В те времена человеку, получившему текстовый файл, нередко приходилось подбирать опытным путем ту кодировку, посредством которой этот файл был создан. Переключение на разные кодировки было автоматизировано, но сам по себе такой поиск не доставлял радости в работе.

Со временем в множестве различных расширений был наведен некоторый порядок. Кое-какие расширения были исключены из пользования, а остальным — наиболее употребительным — были присвоены номера. Эти расширения получили название кодовых страниц. Для русского языка это прежде всего кодовые страницы CP-866 и CP-1251 (последнюю называют еще Windows-1251, поскольку именно эта таблица используется приложениями, работающими под ОС Windows). Сокращение CP происходит от английского Code Page — кодовая страница. В свою очередь, кодовая страница CP-866 используется при работе под ОС MS-DOS — предшественницей Windows. В таблицах 2.4—2.5 приведены коды CP-866 и CP-1251. Для экономии места мы не приводим двоичные коды символов, а только указываем их десятичный порядковый номер.

В сети Интернет, наряду с другими, используется таблица КОИ-8 (Код Информационного Обмена 8-битный). Здесь она представлена таблицей 1.6. Еще недавно при работе в сети с помощью операционных систем MS-DOS и Windows это вызывало определенные трудности. Современные браузеры обеспечивают обычно автоматически, иногда вручную соответствующую перекодировку.

К примеру, текст «Привет!» будет иметь следующую кодировку (в десятичном виде):

в CP-866 — 143 224 168 162 165 226 033;
в CP-1251 — 207 240 232 226 229 242 033;
в КОИ-8 — 240 210 201 215 197 212 033.

Таблица 2.4 **Расширение CP-866**

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

Таблица 2.5 **Расширение CP-1251**

Ђ	Ѓ	‚	ѓ	„	…	†	‡	€	‰	Љ	‹	Њ	Ќ	Ћ	Џ
128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
ђ	‘	’	“	”	•	–	—	□	™	љ	›	њ	ќ	ћ	џ
144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
	Ў	ў	J	¤	I	¦	§	Ё	©	Є	«	¬	-	®	Ї
160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
°	±	I	i	ı	µ	¶	·	ё	№	є	»	j	S	s	ї
176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Таблица 2.6 **Расширение КОИ-8 (koi-8r)**

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

Ограниченность восьмибитной кодировки, не позволяющей одновременно пользоваться несколькими языками, привела к разработке нового кода. Таким, как вы знаете, стал разработанный в 1993 г. новый стандарт **Unicode**. В нем на кодирование одного символа отводится 31 бит. Первые 128 номеров, как и прежде, отведены под ASCII, далее размещены основные алфавиты современных языков. Они полностью умещаются в первой части таблицы, их коды не превосходят $65\,535 = 2^{16} - 1$. В целом Unicode описывает алфавиты всех известных языков, в том числе мертвых. Если язык имеет несколько официальных алфавитов или вариантов написания, то в Unicode они все представлены. Он содержит всю математическую и иную научную символику. Более того, он содержит коды некоторых придуманных языков, например языки эльфов и Мордора из трилогии Дж. Р. Р. Толкиена «Властелин колец». Однако сейчас использовано менее одной тысячной части возможных кодов этого стандарта — настолько велика его информационная емкость.

В современных компьютерных операционных системах используется укороченная 16-битная версия Unicode. Она называется базовой многоязыковой страницей — Base Multilingual Plane (BMP). В Unix-подобных ОС работа с Unicode-текстами напрямую невозможна ввиду особенностей архитектуры. Поэтому в таких системах

используются особые формы данного стандарта, которые называются UTF — Unicode Transformation Form. В них символы кодируются переменным количеством байтов. Например, в UTF-8 коды символов занимают от 1 до 6 байтов.

Вопросы и задания

1. Какие десятичные номера имеют символы, содержащиеся в основной части кодовой таблицы ASCII?
2. Зависит ли десятичный код цифры от того, какое выбрано расширение таблицы ASCII? Ответ «да» надо подтвердить примером такой цифры, ответ «нет» обосновать.
3. Представьте десятичным кодом в трех различных кодировках текст «Никогда не говори никогда!».
4. Получено несколько русскоязычных сообщений, которые не удается прочитать. Для каждого сообщения попытайтесь определить, в какой кодировке пришло сообщение и в какой было отправлено. Раскодируйте каждое сообщение (стоящий в конце пунктов символ «;» в сообщение не входит).

   а) = р±≥≤яшыр ю±хэ^{n};

   б) с ЪОБА ЙОЖПТНБФЙЛХ ОБ ПФМЙЮОП;

   в) ▒Б╝╚Ф╒╞ ▒╝АА┐┐ О╒╩О╒БАО ■╝А

   г) Ќ Ґ §- п Ўа®¤ , -Ґ бг©бп ў ў®¤г;

   д) оПХУНДХ, бЮМЪ, 1-ЦН ЮОПЕКЪ ЙН ЛМЕ Б ЦНЯРХ!
5. Подсчитайте информационную емкость Unicode.

§ 14 Кодирование цветовой информации

Прежде всего кратко повторим то, что вы узнали о кодировании графической информации в 10 классе. Во-первых, для кодирования любое изображение подвергается дискретизации, т. е. разбиению на маленькие части. Во-вторых, цвет каждой такой части описывается количеством каждого из основных цветов — красного, зеленого и синего, при смешивании которых получается нужный цвет.

Когда художник рисует картину, оттенки цветов он выбирает по своему вкусу. Но в любом технологическом процессе цвет необходимо стандартизировать, чтобы воспроизводить его совершенно точно и однозначно. Поэтому надо определить, что такое красный,

Таблица 2.7 **Стандарты цвета CIE**

Цвет	Красный	Зеленый	Синий
Длина волны, мкм	0,7	0,5641	0,4351

зеленый и синий цвета. Как вы знаете из физики, цвет определяется длиной волны. В 1931 г. в качестве международного стандарта была принята система CIE (Commission Internationale d'Eclairage). В этой системе три основных цвета стандартизированы так, как показано в таблице 2.7. В таблице 2.8 указано, какие цвета получаются при смешивании этих цветов в одинаковых пропорциях. Символ 1 обозначает наличие цвета, символ 0 —его отсутствие

Впрочем, для получения того или иного цветового оттенка можно комбинировать и другие цвета, отличные от указанных выше. Этот факт был обнаружен М. В. Ломоносовым в его экспериментах по производству цветных стекол для мозаики и теоретически обобщен Германом Грассманом в виде законов **аддитивного синтеза цвета**. Слово «аддитивный» означает, что речь идет о смешивании, или сложении, цветов. Мы сформулируем два из этих законов, наиболее важные для понимания сути цветовоспроизведения и цветового кодирования.

Таблица 2.8 **Кодирование основных цветов**

Красный	Зеленый	Синий	Цвет
0	0	0	Черный
0	0	1	Синий
0	1	0	Зеленый
0	1	1	Голубой
1	0	0	Красный
1	0	1	Пурпурный
1	1	0	Желтый
1	1	1	Белый

Закон трехмерности. С помощью трех независимых цветов можно, смешивая их в однозначно определенной пропорции, выразить любой цвет.

Цвета некоторого набора цветов называются **независимыми**, если никакой из них нельзя получить, смешивая остальные цвета этого набора.

Закон непрерывности. При непрерывном изменении пропорции, в которой взяты компоненты цветовой смеси, получаемый цвет также меняется непрерывно.

Подчеркнем, что эти законы отражают *восприятие* цвета человеком. С физической точки зрения цвет характеризуется длиной волны. Как вам известно из курса физики, еще И. Ньютон разложил белый свет на спектральные составляющие и выделил из них семь наиболее заметных: красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый. Так что свет желтого цвета — это не смесь красного и зеленого!

Почему же в качестве основных цветов для синтеза цвета на экранах компьютера, телевизора и других устройств отображения графической информации выбраны именно красный, зеленый и синий цвета? Ответ снова кроется в физиологии человеческого зрения. Вы знаете, что рецепторы человеческого глаза делятся на две группы: палочки и колбочки. Палочки более чувствительны к интенсивности поступающего света, а колбочки — к длине волны. Поэтому восприятием цвета человек обязан именно колбочкам. Если посмотреть, как распределяется количество колбочек по тому, на какую длину волны они «настроены», то окажется, что наибольшие доли имеют как раз колбочки, воспринимающие синий, зеленый и красный цвета. Естественно эти цвета взять основными и, смешивая их, получать остальные цвета. Согласно закону трехмерности этих цветов достаточно, а любой цвет задается тройкой чисел (a, b, c), показывающих, в каком соотношении нужно взять эти цвета. Можно при этом считать, что каждое из чисел меняется в диапазоне от 0 до 1, где числу 1 соответствует максимально возможная яркость источника света, передающего данный цвет, а числу 0 соответствует отсутствие света, несущего данный цвет. Обычно при технической реализации данной модели цветопередачи стремятся к тому, чтобы исходная яркость каждого из основных цветов была одинаковой, иначе возникнут искажения (многие, наверно, наблюдали подобные искажения на экранах своих стареньких телевизоров, у которых от времени уже произошло рассогласование яркости трех основных цветов).

Указанные три числа можно рассматривать как код любого цвета — ведь они его однозначно определяют. Такой способ кодирования называют **RGB-кодированием** по первым буквам английских названий трех основных цветов: Red — красный, Green — зеленый и Blue — синий. Еще раз подчеркнем, что данный способ кодирования цвета связан именно с особенностями человеческого зрения. У собак оно, например, смещено в сине-фиолетовую область (захватывая немножко невидимый для человека ультрафиолетовый спектр). Поэтому человек долгое время считал, что у собак вообще нет цветного зрения. У других животных восприятие цвета смещено, наоборот, в красную и инфракрасную область — они даже способны видеть тепловые лучи. Если бы видеотехнологию разрабатывали такие животные, то у них было бы иное цветовое кодирование.

Описание цвета тройкой чисел наводит на мысль считать эти числа координатами точки в пространстве. Тогда получается, что коды всех цветов заполняют куб с ребром 1. Такой куб изображен на рисунке 2.1 и на форзаце учебника.

Яркость цвета определяется тем, насколько близка к 1 хотя бы одна из координат точки, соответствующей данному цвету. А чем ближе точка к диагонали, соединяющей черный и белый цвета, тем меньше насыщенность цвета, обозначенного этой точкой.

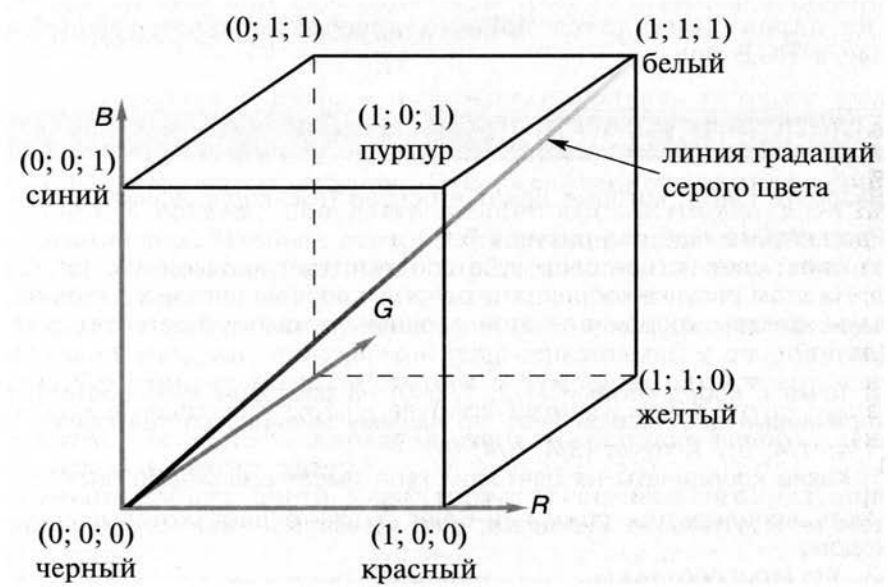


Рис. 2.1 Цветовой куб для RGB-кодирования

Вы уже знаете, что конечность разрядной сетки не позволяет использовать для кодирования информации любые числа. К счастью, закон непрерывности позволяет для каждого цвета построить весьма близкое приближение.

В современных компьютерах используется 16-битное (режим Hi-Color) и 24-битное (режим True-Color) кодирование. В первом случае оказывается возможным закодировать $2^{16} = 65\,536$ цветов, во втором — $2^{24} = 16\,777\,216$ цветов. В режиме True-Color на кодирование градаций яркости каждого из основных цветов отводится 1 байт: код 00000000 показывает, что данного цвета нет вообще, а код 11111111 соответствует наибольшей интенсивности (яркости) кодируемого цвета. При 16-битном кодировании ситуация иная: на кодирование яркости красного и синего цвета отводится по 5 бит, а на кодирование зеленого цвета — оставшиеся 6 бит. Поэтому в данном режиме шкала яркости для зеленого цвета содержит в 2 раза больше градаций, чем для красного и синего.

При использовании кода True-Color изменение значения одного бита дает настолько незначительное изменение цвета фигуры, что человеческий глаз его не улавливает.

Поскольку именно модель RGB-кодирования соответствует механизму формирования цветного изображения на экране монитора, практически все форматы графических файлов хранят изображение в этой кодировке. Если же используется другая цветовая модель (например, в формате JPEG), то компьютеру при выводе изображения на экран приходится спешно преобразовывать графические данные в RGB-код.

Вопросы и задания

1. Назовите цвета, которые лежат в основе RGB-кодирования.
2. Рассмотрите еще раз рисунок 2.1.
   а) Какой цвет на цветовом кубе соответствует вершине (0; 1; 1)?
   б) На этом рисунке координаты одной из вершин цветового куба не указаны. Каковы координаты этой вершины и какому цвету она соответствует?
3. а) Точке с координатами (1/2; 1/2; 0) на цветовом кубе соответствует коричневый цвет. Какой цвет, по вашему мнению, соответствует точке (1/4; 1/4; 0)? А точке (3/4; 3/4; 0)?
   б) Какие координаты на цветовом кубе имеет оранжевый цвет?
4. Пусть используется режим Hi-Color. Укажите цвет, который задается кодом:
   а) 1111100000011111;
   б) 0111101111101111;
   в)* 1111101111100000.

5. Пусть используется режим True-Color. Укажите цвет, который задается кодом:
   а) 000000001111111111111111;
   б) 011111110111111101111111;
   в)* 011111110000000001111111.

6. Сформулируйте законы восприятия цвета, благодаря которым возможна дискретизация цвета при его кодировании. Почему трехбайтового кодирования оказывается достаточно для полноценного цветовоспроизведения?

7. а) Вы хотите работать с разрешением 800 × 600 пикселей, используя одновременно 65 536 цветов (16-битное кодирование). В магазине продаются видеокарты с памятью 256 Кб, 512 Кб, 1 Мб, 2 Мб, 4 Мб. Какие карты подходят для вашей работы?
   б) Если вам необходимо разрешение 1600 × 1200 пикселей и работа с 16 777 216 цветами (24-битное кодирование), какие тогда видеокарты подходят для вашей работы?
   в) Подсчитайте объем памяти, необходимый для записи 1 секунды видеофильма (25 кадров с разрешением 1024 × 768 пикселей) при использовании режима True-Color.

§ 15 Цветовая модель HSB

Попросите любого художника назвать оттенки красного цвета — и вы услышите: красный, темно-красный, алый, розовый, бледно-розовый и т. п. Еще, говоря о цвете, обычно отмечают его яркость, насыщенность. Эти характеристики цвета, в отличие от RGB-модели, передают именно субъективное восприятие цвета человеком. Чтобы с этими характеристиками можно было работать на компьютере, их надо формализовать и описать числовыми параметрами.

С физической точки зрения яркость — это количественная мера световой энергии, излучаемой или отражаемой в сторону наблюдателя. Так, при солнечном свете и в сумерках один и тот же цвет выглядит по-разному, хотя цветовой оттенок один и тот же. Можно сказать, что яркость определяется тем, сколько серого цвета добавлено к основному цвету.

Насыщенность цвета характеризует степень его разбавления белым цветом. Чем больше белого цвета вы добавите, тем меньше насыщенность.

Что касается цветового оттенка, то он определяется расположением цвета в световом спектре. Как вы знаете из физики, видимые человеком цвета располагаются в спектре следующим образом:

Рис. 2.2 Круговое расположение цветов

красный — оранжевый — желтый — зеленый — голубой — синий — фиолетовый. Нередко их располагают по кругу, который называют **кругом Манселла**. Круг Манселла изображен на рисунке 2.2 и на форзаце учебника.

Тогда цветовой оттенок кодируется либо величиной угла, либо длиной дуги, считая, что длина всей окружности равна 1. При этом 0° или дуга нулевой длины соответствует красному цвету.

Чтобы описать две другие характеристики, используют пространственное изображение этой модели в виде конуса (рис. 2.3). Угол между осью конуса и образующей характеризует насыщенность цвета, а удаленность от вершины —яркость.

Таким образом, и в этой модели цветопередача характеризуется тремя числами. Сама модель получила название HSB — по первым буквам слов Hue (цветовой оттенок), Saturation (насыщенность) и Brightness (яркость).

Вопросы и задания

1. Какие факторы принимаются как существенные при построении RGB-модели цветопередачи и какие — при построении HSB-модели?
2. Какими характеристиками цвета оперирует HSB-модель цветопередачи?
3. Как кодируется цветовой оттенок в HSB-модели?
4. Укажите, где на цветовом круге располагаются коричневый, оранжевый и фиолетовый цвета. Какими числами, на ваш взгляд, эти цвета кодиру-

Рис. 2.3 Цветовая модель HSB

§ 16 Получение изображений на бумаге

До этого момента мы говорили о видеоизображении на экране компьютера или другого электронного устройства. А как возникает изображение при печати?

Давайте вооружимся лупой и посмотрим на какую-нибудь цветную иллюстрацию, напечатанную в типографии. Видите: изображение распалось на мельчайшие кляксы всего лишь четырех цветов, причем очень знакомых: черного, пурпурного, голубого и желтого. Они же фигурируют в таблице 2.8. Давайте порассуждаем и постараемся объяснить картину, представшую перед нами под лупой.

Итак, белый цвет можно рассматривать как смесь трех основных цветов. И когда мы видим белый лист бумаги, в наш глаз попадают лучи всех цветов, отраженные от ее поверхности, как это изображено на рисунке 2.4.

Если мы видим на бумаге красный цвет, это означает, что:

1) либо из трех составляющих белого цвета, которым освещается лист бумаги, осталась только одна — красная:
2) либо краска, нанесенная на лист бумаги, отразила только красную составляющую.

Казалось бы, какая разница? Однако в этих двух случаях мы имеем дело с принципиально разными типами красок.

Первый тип, к которому и относится типографская краска, можно рассматривать как фильтр, задерживающий зеленый и синий лучи. Любую типографскую краску надо рассматривать как фильтр, задерживающий те или иные цвета. Нанесение типографской краски на черный лист оставит его черным.

Второй тип — это плотная отражающая краска типа гуаши или масляной. Ее можно наносить на бумагу любого цвета и получать при этом изображения необходимых цветов.

Рассмотрим более пристально типографскую краску. Допустим, что в качестве основных красок мы снова взяли красную, синюю и зеленую. Это значит, что красная краска поглощает синий и зеленый цвета, синяя — красный и зеленый цвета, а зеленая — красный и синий. Иными словами, любая из этих трех красок не даст нам более одной составляющей. В случае соединения двух или

Рис. 2.4 Отражение лучей от листа белой бумаги

более красок задерживается еще больше цветов. Поэтому если соединить две из указанных нами красок, то получившийся «светофильтр» никаких лучей уже не пропустит, и мы увидим... Все, наверно, догадались, что мы увидим. Ясно теперь, что, к примеру, желтый цвет с помощью таких красок уже не получить — для него требуются две составляющие: красная и зеленая.

Поэтому в случае цветной печати типографскими красками изображение приходится формировать из таких красок, каждая из которых задерживает только *одну* составляющую.

Какой же цвет на бумаге поглощает красную составляющую? Конечно, голубой. Синий цвет поглощается желтым, а зеленый — пурпурным. Черный цвет мы получим, если нанесем на бумагу все три поглощающих цвета.

Взаимодействие основных цветов при печати отражено в таблице 2.9.

Модель цветопередачи, при которой основными являются не излучающие, а поглощающие цвета, называется **субтрактивной** или **вычитательной.**

В вычитательной модели цветовой куб оказался как бы перевернутым: отсчет в нем начинается из точки, которой соответствует белый цвет, а оси направлены по ребрам, сходящимся в этой точке.

Таблица 2.9 **Кодирование основных цветов при печати**

Голубой (нет красного)	Пурпурный (нет зеленого)	Желтый (нет синего)	Цвет
0	0	0	Белый
0	0	1	Желтый
0	1	0	Пурпурный
0	1	1	Красный
1	0	0	Голубой
1	0	1	Зеленый
1	1	0	Синий
1	1	1	Черный

Кодировка, которую мы только что рассмотрели, называется **CMY-кодировкой** — по первым буквам английского названия основных цветов — Cyan, Magenta, Yellow.

Осталось объяснить, откуда же взялись черные кляксы на цветной картинке, которую мы рассматривали.

Если вы внимательно полистаете любое цветное издание, то заметите, что все же в основном оно печатается черной краской — это цвет текста. Черного цвета хватает и в иллюстрациях. Получать его на бумаге смешением трех основных цветов неудобно по трем причинам:

- невозможно произвести идеально чистые пурпурные, голубые и желтые краски, и из-за этого получается не чисто черный, а темно-темно-коричневый цвет;
- на черный цвет при CMY-кодировке тратится в три раза больше краски;
- любые цветные чернила дороже обычных черных.

Поэтому на практике к базовому набору из трех красок добавляется — для качественной и более экономной печати — черный цвет. Такая кодировка называется **CMYK-кодировкой** (от слова blacK взяли последнюю букву, чтобы не путать с сокращением Blue).

Естественно, необходимо учитывать особенности CMYK-кодировки и уметь переводить картинки из RGB в CMYK, если вы занимаетесь подготовкой какой-либо печатной продукции.

Если вы занимаетесь подготовкой печатной продукции, необходимо учитывать особенности CMYK-кодировки и уметь переводить картинки из RGB в CMYK. В задании 3 к этому параграфу предлагается вывести формулы для такого перекодирования. Чтобы это сделать, надо изобразить координатный куб, в котором по осям откладываются основные цвета CMY. Если внимательно рассмотреть цветовой куб RGB, станет ясно, где разместится начало координат и как будут направлены оси.

Вопросы и задания

1. Почему при печати на принтере приходится использовать иные цвета, нежели для цветного воспроизведения на экране компьютера?
2. Какие цвета являются основными при CMY-кодировке? А при CMYK-кодировке?
3. Напишите формулы перехода из RGB-кодировки в CMY-кодирование.
4. Напишите CMY-код коричневого цвета. Типографскую краску каких цветов и в какой пропорции надо нанести на бумагу, чтобы получить коричневый цвет? А оранжевый? (С о в е т . Воспользуйтесь информацией, полученной вами при выполнении задания 3 из § 14.)

§ 17 Коды, обнаруживающие и исправляющие ошибки

Вы уже знаете, что часто для передачи сообщений используется двоичное кодирование, т. е. каждому символу алфавита, с помощью которого записывается сообщение, сопоставляется последовательность, состоящая из 0 и 1. Вспомните, именно такое сопоставление представлено кодовыми таблицами, задающими ASCII-кодирование (см. § 13). К примеру, в расширении CP-1251 символы «а», «и», «п», «р» задаются кодами 11100000, 11101000, 11101111, 11110000 соответственно. Слово «пир» будет закодировано последовательностью

111011111110100011110000.

Но представьте себе, что при передаче этого кода один из символов оказался переданным ошибочно — вместо 1 оказался 0 — и на приемник информации поступила последовательность

111011111110<u>0</u>000011110000.

Ошибочный символ в ней подчеркнут. Тогда эта последовательность будет декодирована как «пар».

От ошибок не застрахованы не только люди, но и технические системы. К подобной ошибке мог привести обыкновенный технический сбой, например перепад в напряжении. Воздействие, приводящее к искажению передаваемой информации, обычно называют **шумом**. На рисунке 2.5 схематично показано воздействие шума на процесс передачи информации.

Рис. 2.5

Если сообщение носит «бытовой» характер, подобная ошибка может не привести к тяжелым последствиям. А если речь идет о передаче команд управления космическим кораблем или атомной электростанцией? Важно, чтобы ошибки, возникающие при кодировании и передаче информации, могли распознаваться автоматически. Оказывается, что это вполне возможно. Только кодирование должно обладать подходящими свойствами.

В жизни, если кто-то не услышал, что вы сказали, вас просто попросят повторить фразу. При передаче можно было бы поступать так же: дублировать каждый передаваемый символ, и тогда простое сравнение двух последовательных кодов легко выявляет ошибку. В приведенном нами примере получилось бы так:

11111100111111111111110001000000111111100000000.

Достаточно разбить все символы на последовательность пар, чтобы увидеть, что в подчеркнутой паре символы не совпали. Значит, в этом месте произошла передача ошибочного символа.

Если задуматься над использованным нами приемом, то станет ясно, что мы вместо восьмибитового кодирования каждого символа стали использовать шестнадцатибитовое кодирование. При этом все наши сообщения стали в два раза длиннее, и на их передачу требуется в два раза больше времени. Довольно высокая цена, если учесть, что ошибки не встречаются часто. В нашем примере на 24 символа оказался всего лишь один неверно переданный символ.

Чтобы продемонстрировать идею более экономного кодирования, позволяющего обнаруживать ошибки, предположим, что все передаваемые сообщения — это числовые данные, записываемые цифрами обычной десятичной системы счисления. Каждую цифру будем кодировать ее представлением в двоичной системе счисления. Результат такого кодирования можно увидеть в первых двух столбцах таблицы 2.10.

Таблица 2.10

Цифра	Двоичный код	Расширенный код
0	0000	00000
1	0001	00011
2	0010	00101
3	0011	00110
4	0100	01001
5	0101	01010
6	0110	01100
7	0111	01111
8	1000	10001
9	1001	10010

Легко понять, что предложенное кодирование не позволяет обнаружить ошибку. Если вместо 0010 пришло ошибочное 0011, то в такое сообщение можно поверить как в правильное.

Давайте добавим в код каждой цифры справа еще один символ. Сделаем это так, как показано в третьем столбце таблицы 2.10. Правило, по которому приписывается еще один символ в код, очень простое: если в исходном четырехсимвольном коде четное число единиц, то пишем 0; если нечетное, то пишем 1. В получившемся коде для любого символа количество единиц всегда четно. Поэтому если вдруг при передаче сообщения в каком-то месте произошла ошибка, т. е. 0 заменился на 1 или наоборот, то количество единиц в таком коде соответствующего символа стало нечетным, и это мгновенно обнаруживается.

Например, пришло сообщение

100100011010011.

Разбиваем его на 3 группы по 5 символов:

10010 00110 10011.

Сразу видно: первые две группы вне подозрений, а третья наверняка неправильная.

Чтобы описать способность кода к распознаванию ошибок, используют понятие расстояния между словами. Пусть даны слова $a_1a_2 \ldots a_n$ и $b_1b_2 \ldots b_n$ над одним и тем же алфавитом. **Расстоянием между словами** называют количество позиций, в которых символы одного слова не совпадают с символами другого. Например, расстояние между словами «стог» и «снег» равно 2 — они различаются во второй и третьей позициях, а между словами 1001001 и 0100001 равно 3 — они различаются в первой, второй и четвертой позициях. По-другому расстояние между словами называют **расстоянием Хэмминга.**

Давайте еще немного «нарастим» код для цифр, который мы рассмотрели в таблице 2.10, — увеличим количество символов в каждом кодовом слове до 7 таким образом, чтобы минимальное расстояние между кодовыми словами было равно 3. Это можно сделать, например, так, как показано в третьем столбце таблицы 2.11.

Если при таком кодировании при передаче сообщения произошло 2 ошибки, то все равно принятое слово не совпадет ни с одним кодовым словом. Иными словами, будет выявлено ошибочно переданное слово. Более того, весьма маловероятно, чтобы в семибитовом слове оказалось сразу 2 ошибки, поэтому, получив слово с ошибкой, мы найдем ближайшее к нему слово (т. е. отличающееся только на один символ) и можем с уверенностью исправить ошибку.

Отметим, что, поскольку расстояние между любыми двумя кодовыми словами не меньше 3, кодовое слово, ближайшее к ошибочному, будет единственным. Пусть, к примеру, получено сообщение

0010111011001110101 11.

Разбиваем это сообщение на группы по 7 символов и получаем

0010111 0110011 1010111.

Таблица 2.11

Цифра	Двоичный код	Расширенный код
0	0000	0000000
1	0001	0001111
2	0010	0010110
3	0011	0011001
4	0100	0100101
5	0101	0101010
6	0110	0110011
7	0111	0111100
8	1000	1000011
9	1001	1001100

В таблице 2.11 нет кодового слова, соответствующего первой группе символов. Но на расстоянии 1 от него находится код 0010110, значит, допущена одна ошибка. Исходно был передан код 0010110, или цифра 2. Вторая группа является кодовым словом, она соответствует цифре 6, а третья группа снова ошибочна. Ближайшее к ней кодовое слово — 1000011. Это код цифры 8. Значит, было передано число 268.

Таким образом, построенный код — он называется **кодом Хэмминга** — гарантированно исправляет одну ошибку. Более того, исправление производится по вполне очевидному алгоритму, и, следовательно, исправлять такую ошибку можно поручить компьютеру.

Разработанная математиками теория кодирования позволяет строить коды с заданным минимальным расстоянием между кодовыми словами. Так, в европейских системах связи широко используется 235-битовый код, расширенный с помощью дополнительных 20 символов. Минимальное расстояние между словами этого кода равно 7. Такой код гарантированно обнаруживает 6 ошибок и исправляет слова, в которых допущено не более 3 ошибок. В течение многих лет эксплуатации этих систем не было случая, чтобы ошибка прошла незамеченной.

Вопросы и задания

1. Что такое шум с точки зрения передачи информации?
2. Что такое расстояние между словами?
3. Найдите расстояние между словами каждой пары:
   а) собака и корова; б) паровоз и самовар; в) 10010110 и 10110100.
4. Рассматривается множество всех пятисимвольных слов над алфавитом {0, 1}.
   а) Перечислите все слова, находящиеся на расстоянии 1 от слова 10101.
   б) Перечислите все слова, находящиеся на расстоянии 2 от слова 01010.
5. * Рассматривается множество всех n-символьных слов над алфавитом {0, 1}. Сколько имеется слов, находящихся на заданном расстоянии d от некоторого слова из этого множества? Зависит ли это количество от выбранного слова?
6. На рисунке 2.6 линиями соединены те символы, закодированные кодом Хэмминга (см. таблицу 2.11), расстояние между которыми равно 3. Проверьте, что расстояние между символами любой другой пары больше 3.
7. а) Получено сообщение, закодированное семибитовым кодом Хэмминга: 0010111001000010000001000001. Декодируйте его, исправив, если необходимо, ошибки.
   б) Выполните такое же задание для сообщения
   10011010110100010111000010 11.
8. Символы a, b, c, d закодированы следующим образом:
   $a \to 000000$, $b \to 010101$, $c \to 101010$, $d \to 111111$.
   а) Каково минимальное расстояние между кодовыми словами?
   б) Получено сообщение 110101101011001111110000. Попытайтесь его декодировать, исправив, если необходимо, ошибки.
9. * Для кодирования 15 букв русского алфавита и пробела использовался следующий код:
   А→0001111; Б→0010110; В→0011001; Г→0100101; Д→0101010; Е→0110011; Ж→0111100; З→1000011; И→1001100; Й→1100110; К→1110000; Л→1011010; М→1010101; Н→1101001; О→1111111; пробел→0000000.
   а) Найдите кодовое расстояние. Сколько ошибок находит и сколько исправляет этот код?
   б) Декодируйте следующее сообщение:
   0011011001111101110001100001111101101110110001000010001001101 1110110001111011100000001000100001110110000111111011

⑩ Приведенное в таблице 2.11 кодирование десяти цифр — это только часть кода, изобретенного Хэммингом для кодирования всевозможных четырехбитовых последовательностей. Минимальное расстояние между кодовыми словами в этом коде равно 3. Попытайтесь найти кодовые слова для остальных шести четырехбитовых последовательностей так, чтобы минимальное расстояние между словами по-прежнему было равно 3.

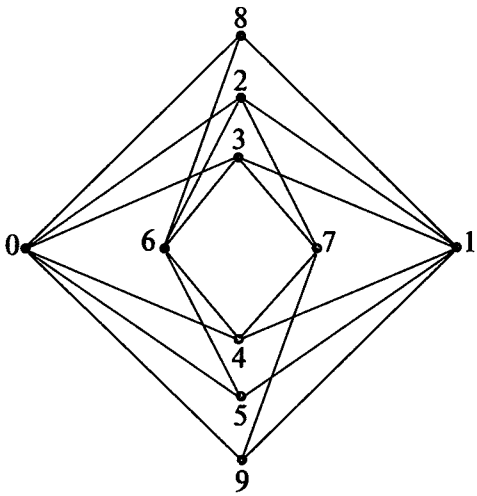


Рис. 2.6 Символы с кодовым расстоянием 3

⑪ Обозначим через $d(a, b)$ расстояние между словами a и b. Докажите, что для любых слов a, b и c выполняется неравенство $d(a, b) \leqslant d(a, c) + d(c, b)$. Это неравенство называют **неравенством треугольника** за его аналогичность известному геометрическому неравенству «Сторона треугольника меньше суммы двух других его сторон».

⑫ Докажите, что при кодировании кодом Хэмминга у каждого семибитового слова, содержащего одну ошибку, имеется ровно одно ближайшее к нему правильное кодовое слово. (С о в е т. Воспользуйтесь неравенством, сформулированным в задании 11.)

⑬ Пусть минимальное расстояние между кодовыми словами некоторого кода равно 7.

а) Докажите, что такой код обнаруживает до 6 ошибок.

б)* Докажите, что любое слово, содержащее не более 3 ошибок, имеет ровно одно ближайшее к нему кодовое слово. Это означает, что данный код исправляет 3 ошибки.

⑭ Напомним, что **графом** называется совокупность точек и линий, соединяющих некоторые из точек. Точки называют **вершинами** графа, а соединяющие линии — **ребрами**.

а) Рассматривается множество всех трехсимвольных слов над алфавитом {0, 1}. Изобразите граф, вершинами которого являются эти 8 слов, а две его вершины соединены ребром тогда и только тогда, когда расстояние между словами равно 1.

б) Выполнив задание 10, вы нашли 16 кодовых слов кода Хэмминга. Изобразите граф, вершинами которого являются эти 16 слов, а две его вершины соединены ребром тогда и только тогда, когда расстояние между словами равно 3.

§ 18 Экономные коды. Алгоритмы сжатия

В предыдущем параграфе мы обсудили проблему надежности сохранения информации при ее передаче по каналам связи или записи на какой-либо носитель. Но кодирование информации можно рассматривать еще с одной стороны — экономической. Поясним суть вопроса на примере.

Допустим, что по каналу связи передается только числовая информация. Тогда нет никакого резона тратить на кодирование каждой цифры 8 бит, как это происходит, если мы пользуемся ASCII. Вполне достаточно 4 бит, даже если потребуется отдельно закодировать символы «пробел», знаки «плюс» и «минус», а также десятичную запятую. Можно, к примеру, такое кодирование произвести так, как показано в таблице 2.12 (в ней для удобства символ «пробел» обозначен символом «_»).

Тогда сообщение

«0,258 23,5 −67,08 2478 2,5 −0,0012»

кодируется как

0000110100100101100010100010001111010101101011000110011111010000100010100010010001111000101000101101010110101100000011010000000000010010.

Экономия двукратная: в ASCII исходное сообщение займет 34 байта, а в нашем коде всего лишь 17 байт.

Конечно декодирующему устройству надо сообщить о способе кодирования, т. е. переслать коды символов и количество бит в коде одного символа. Иными словами, перед самим сообщением придется записать еще сведения, содержащиеся в таблице 2.12. Выглядеть это может, например, так, как показано на рисунке 2.7.

Сначала мы указали длину кода каждого символа, а затем попарно записали ASCII-код символа и его код в нашей системе. В конце записан код символа, который в ASCII является управляющим, а не текстовым. Нетрудно подсчитать, что для передачи этой информации требуется 23 байта. И при передаче нашего сообщения мы не только ничего не выиграли, но даже и проиграли. Однако реально подобные методы применяются, когда речь идет об объемах в сотни килобайт и десятки мегабайт. На этом фоне потратить 23 байта, чтобы в два раза сжать информацию, весьма высокая эффективность.

Таблица 2.12

0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0101
7 0111	8 1000	9 1001	– 1010	+ 1011	− 1100	, 1101

00110100	00110000	0000	00110001	0010	...	00111001	1001	00100000	1010
ASCII код числа 4	ASCII код числа 0	код	ASCII код числа 1	код		ASCII код числа 9	код	ASCII код пробела	код

00101011	1011	10010110	1100	00101100	1101	00001101
ASCII код знака «+»	код	ASCII код знака «–»	код	ASCII код знака «,»	код	служебный код конца кодовой таблицы

Рис. 2.7

Этот метод хорош для любого сообщения, в котором присутствует лишь небольшая часть используемого алфавита. К примеру, вам требуется передать сообщение: «КОЛ ОКОЛО КОЛОКОЛА, А КОЛОКОЛ ОКОЛО КОЛА». Здесь использовано всего лишь 6 символов: «А», «К», «Л», «О», «,» и «пробел». Значит, каждый символ можно закодировать, используя 3 бита. Тогда все это сообщение уместится в $3 \cdot 40 = 120$ бит, т. е. 15 байт. В ASCII это сообщение занимает 40 байт. Даже с таблицей кодирования, которая занимает 11 байт, получается полуторакратная экономия. Подсчет числа различных символов в сообщении, определение длины кода, подготовка таблицы кодирования, перекодирование сообщения — все это автоматически выполняет программа-упаковщик. А после получения сообщения программа-распаковщик превращает его в исходный текст.

Описанный **метод упаковки данных** не только не единственный способ сжатия информации, но нередко и не самый эффективный.

Пусть, к примеру, речь идет о передаче сообщений на каком-либо естественном языке. В словах такого языка одни буквы встречаются чаще, другие реже. Если все буквы закодированы двоичными последовательностями одной и той же длины, то при передаче любого сообщения, содержащего n букв, будет всегда тратиться одно и то же время. Именно так обстоит дело при ASCII-кодировании. Другое дело, если буквы, встречающиеся чаще, кодировать более короткими двоичными последовательностями. Экономия времени и энергии может оказаться весьма значительной.

В определенном смысле эта идея реализована в коде азбуки Морзе, где односимвольными и двухсимвольными последовательностями из точек и тире закодированы наиболее часто встречающиеся буквы. Однако при декодировании сообщений, записанных таким кодом, возникает проблема определения того, где кончился код одного символа и начался код следующего.

Проблема кодирования символов последовательностями переменной длины теоретически была решена американскими учеными К. Шенноном и Р. М. Фано. Поэтому условие, достаточное для однозначного декодирования сообщений с переменной длиной кодовых слов, обычно называют **условием Фано**: никакое кодовое слово не является началом другого кодового слова. По-другому условие Фано называют **свойством префиксности**, а код, удовлетворяющий этому условию, называют **префиксным кодом**. Вот пример префиксного кода для 8 символов: 00, 10, 010, 110, 0110, 0111, 1110, 1111. Его эффективно использовать, если есть 2 часто встречающихся символа, 2 символа, встречающихся со средней частотой, и 4 редко встречающихся символа.

Чтобы лучше понять, как строятся префиксные коды, покажем сначала, как каждому набору кодовых слов сопоставить ориентированный граф, определяющий этот код. Напомним, что граф называется **ориентированным** (сокращенно **орграфом**), если на линиях, соединяющих вершины графа, указано направление. Соединяющие линии при этом называются **дугами**.

Пусть, к примеру, код состоит из слов 00, 01, 10, 011, 100, 101, 1001, 1010, 1111, кодирующих соответственно первые 9 букв латинского алфавита: *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, *i* (такой код не является префиксным). Граф этого кода представлен на рисунке 2.8. Он построен следующим образом. Из начальной вершины, которую называют **корнем**, выходят две дуги, помеченные символами алфавита кодирования — в нашем случае 0 и 1. Затем из конца каждой такой дуги выходят новые дуги, помеченные символами кодирующего алфавита так, чтобы, идя по этим дугам от корня, читалось начало какого-либо кодового слова. Если при этом какое-то кодовое слово оказывается прочитанным полностью, то у конца послед-

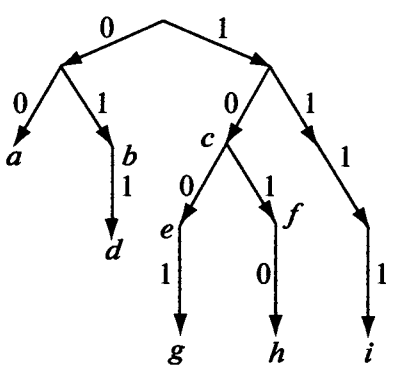


Рис. 2.8

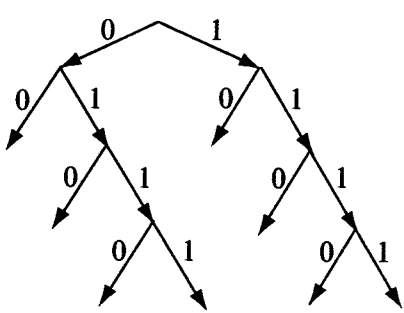


Рис. 2.9

ней дуги пишется кодируемый символ. В нашем случае на этом шаге построения орграфа такими символами оказались *a*, *b* и *c*. Из получившихся вершин снова проводятся дуги и так до тех пор, пока не будут исчерпаны все кодовые слова.

А на рисунке 2.9 представлен орграф для префиксного кода 00, 10, 010, 110, 0110, 0111, 1110, 1111. В чем характерная особенность орграфа префиксного кода? В нем кодируемые символы располагаются только в таких вершинах, из которых не выходят новые дуги. Такие вершины называют **листьями**. Поэтому говорят, что префиксный код задается орграфом с размеченными листьями.

Если вам известен орграф, созданный по префиксному коду, то по этому орграфу легко восстанавливается код каждого символа — надо просто, идя от корня к листу, помеченному данным символом, выписать 0 и 1 в порядке их прочтения.

Идея префиксного кодирования была использована американским ученым Д. Хаффманом для создания эффективного алгоритма сжатия символьной информации. Алгоритм Хаффмана читает данные дважды. При первом считывании он подсчитывает частоту встречаемости каждого символа. Затем по этим частотам строится орграф кодирования, а по нему — коды символов. После этого еще раз прочитываются исходные данные, и они переписываются в новых кодах.

Алгоритм построения орграфа Хаффмана таков:

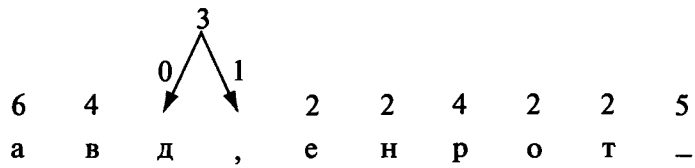
1. Все символы кодируемой информации образуют вершины-листья. Каждой вершине приписывается вес, равный количеству вхождений данного символа в сжимаемое сообщение.
2. Среди вершин, которым приписаны веса, выбираются две с наименьшими весами (если таких пар несколько, выбирается любая из них).
3. Создается следующая вершина графа, из которой выходят две дуги к выбранным на предыдущем шаге вершинам; одна дуга помечается символом 0, другая — символом 1. Созданной вершине приписывается вес, равный сумме весов выбранных вершин, а веса этих двух вершин стираются.
4. К вершинам, которым приписаны веса, применяются шаги 2 и 3 до тех пор, пока не останется одна вершина с весом, равным сумме весов исходных символов.

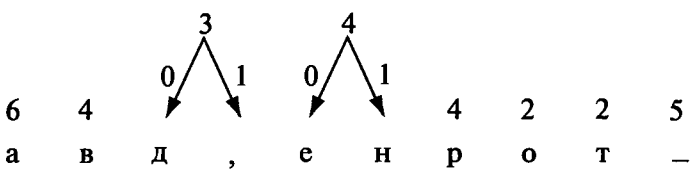
Ниже показано построение кодирующего орграфа Хаффмана для фразы «на_дворе_трава,_на_траве_дрова»; для удобства пробелы обозначены символом «_».

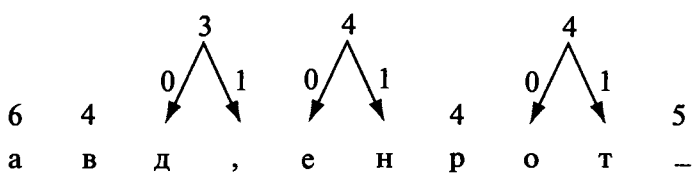
Шаг 1.

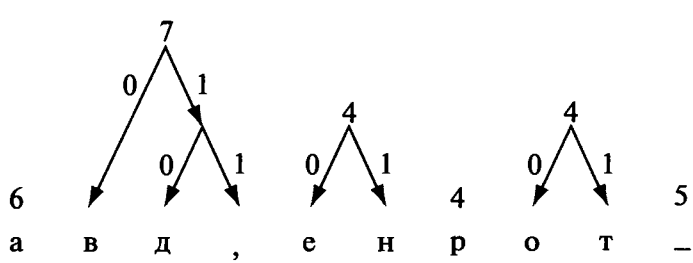
6	4	2	1	2	2	4	2	2	5
а	в	д	,	е	н	р	о	т	–

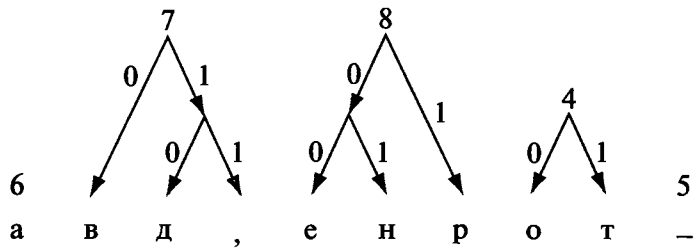
Повторяющиеся шаги 2—3.

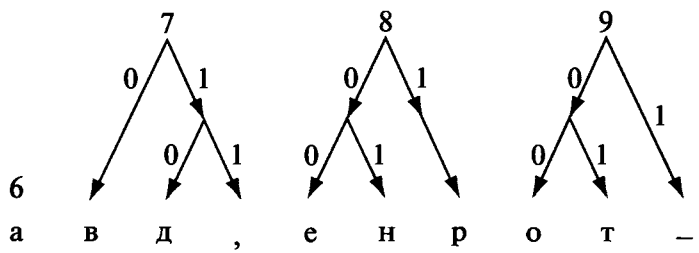


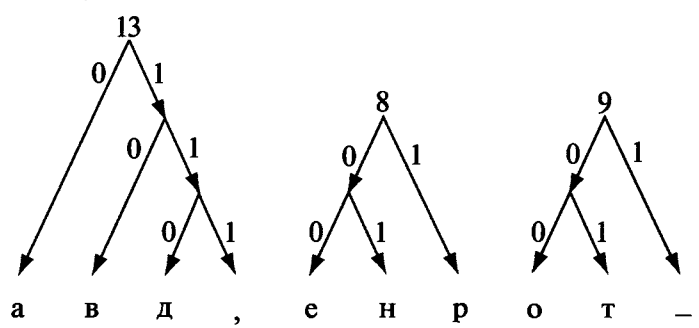


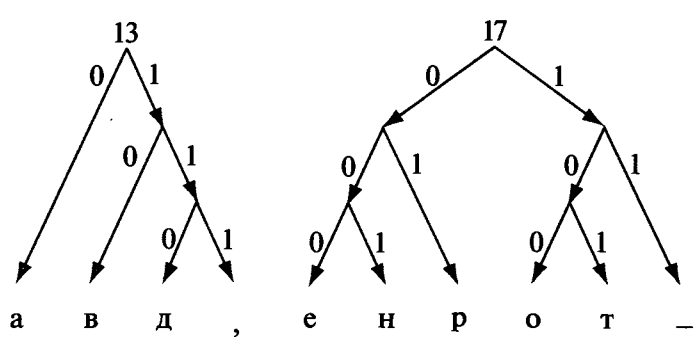


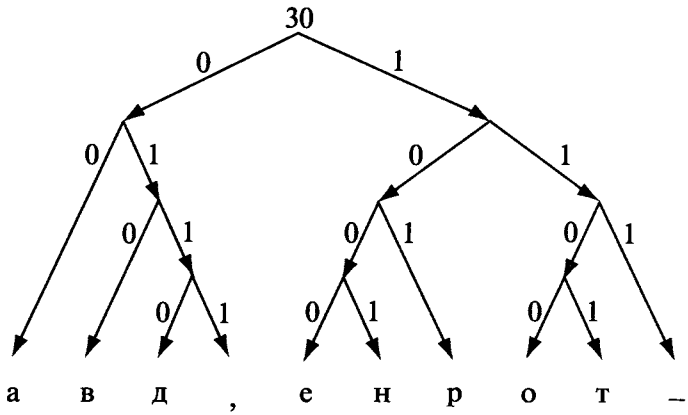



7
8
4
0
1
6
5
а в д , е н р о т –


7
8
9
0
1
6
а в д , е н р о т –


13
8
9
0
1
а в д , е н р о т –


13
17
0
1
а в д , е н р о т –



Теперь определяем коды символов, двигаясь от корня к соответствующему символу. Получившиеся коды приведены в таблице 2.13. В третьей строке таблицы мы указали, сколько раз встречается данная буква в сообщении.

Подсчитаем, сколько двоичных символов окажется в сообщении «на_дворе_трава,_на_траве_дрова» после кодирования его построенным нами кодом Хаффмана. Для этого надо найти произведение числа символов в коде каждой буквы на количество раз, которое эта буква встречается в сообщении, а затем полученные произведения сложить. Получаем

$$2 \cdot 6 + 3 \cdot 4 + 4 \cdot 2 + 4 \cdot 1 + 4 \cdot 2 + 4 \cdot 2 + 3 \cdot 4 + 4 \cdot 2 + 4 \cdot 2 + 3 \cdot 5 = 95.$$

Теперь подсчитаем, сколько символов оказалось бы в двоичном коде этого сообщения, если каждый его символ кодировать цепочкой из 0 и 1 одной и той же длины. Поскольку в сообщении используется 10 различных символов, для их кодирования требуются, как минимум, четырехбитовые цепочки. Поэтому после кодирования данного сообщения получится цепочка объемом 120 бит. Напомним, что коэффициентом сжатия называется отношение объема исходного сообщения к объему сжатого. В нашем случае это отношение равно $120/95 \approx 1{,}26$. Но на самом деле данное сооб-

Таблица 2.13

а	в	д	,	е	н	р	о	т	–
00	010	0110	0111	1000	1001	101	1100	1101	111
6	4	2	1	2	2	4	2	2	5

щение в памяти компьютера закодировано с помощью ASCII, поэтому на каждый символ отведено 8 бит. Тем самым объем исходного сообщения 240 бит, а коэффициент сжатия составляет $240/95 \approx 2{,}53$. Весьма впечатляющий выигрыш, если это сообщение нужно передать по каналу связи или сохранить на каком-либо носителе.

Для декодирования сжатого сообщения вместе с ним обычно пересылают не коды исходных символов (т. е. первые две строки таблицы 2.13), а сам орграф Хаффмана (без указания веса корня и разметки на дугах, ибо она стандартна: дуга, идущая влево, размечается 0, а идущая вправо — 1). На этом, оказывается, тоже можно сэкономить.

Математики доказали, что среди алгоритмов, которые каждый символ кодируют по отдельности целым количеством бит, алгоритм Хаффмана обеспечивает наилучшее сжатие.

Вопросы и задания

1. За счет чего достигается эффект сжатия данных в методе упаковки?
2. Пусть для записи текста используются только заглавные английские буквы и пробел. Каким будет коэффициент сжатия при использовании метода упаковки для передачи такого текста?
3. Какой код называется префиксным?
4. Дан набор кодовых слов: 00, 10, 011, 101, 110, 1001, 1010, 1101, 1111. Постройте для этого кода соответствующий ему орграф. Является ли этот код префиксным?
5. Ниже приведен код азбуки Морзе. Постройте для этого кода соответствующий ему орграф. Определите, является ли этот код префиксным.

А	·–	И	··	Р	·–·	Ш	––––
Б	–···	Й	·–––	С	···	Щ	––·–
В	·––	К	–·–	Т	–	Ъ	––·––·
Г	––·	Л	·–··	У	··–	Ы	–·––
Д	–··	М	––	Ф	··–·	Ь	–··–
Е	·	Н	–·	Х	····	Э	··–··
Ж	···–	О	–––	Ц	–·–·	Ю	··––
З	––··	П	·––·	Ч	–––·	Я	·–·–

6. а) Постройте код Хаффмана для фразы «КАРЛ_У_КЛАРЫ_УКРАЛ_КОРАЛЛЫ,_А_КЛАРА_У_КАРЛА_УКРАЛА_КЛАРНЕТ».

   б) Определите коэффициент сжатия для данной фразы, считая, что для кодирования каждого символа используется 4 бит. Найдите коэффициент сжатия, если каждый символ кодируется в ASCII.

❼ а) Постройте код Хаффмана для фразы
«ШЛА САША ПО ШОССЕ И СОСАЛА СУШКУ».

б) Определите коэффициент сжатия для данной фразы, считая, что для кодирования каждого символа используется 4 бит. Найдите коэффициент сжатия, если каждый символ кодируется в ASCII.

§ 19 Необратимые алгоритмы сжатия

Алгоритмы сжатия, рассмотренные в предыдущем параграфе, обладали важным свойством — они сжимали данные так, что после декодирования исходное сообщение восстанавливалось в первоначальном виде. Такие алгоритмы называют **обратимыми** или **алгоритмами сжатия без потери информации.** Однако такое буквальное воспроизведение исходной информации важно, как правило, лишь для текстовых сообщений. Если же речь идет о звуковой или видеоинформации, то при разработке алгоритмов сжатия можно учесть особенности человеческого восприятия звука и изображения.

Рассмотрим сначала методы сжатия графических данных. Одним из наиболее применяемых методов и фактически признанным сегодня стандартом сжатия является алгоритм, созданный группой экспертов по фотоизображениям. Он получил название **JPEG** по первым буквам английского названия этой организации: Joint Photographic Experts Group.

Алгоритм JPEG основан на учете целого ряда особенностей зрительного аппарата человека. Во-первых, как мы уже рассказывали в учебнике для 10 класса, человеческое зрение обладает некоторой инерцией, т. е. изображение не исчезает мгновенно, а на некоторое время сохраняется. Именно этот эффект привел в свое время к созданию кинематографа. Именно этот эффект позволяет осуществлять дискретизацию изображения с последующей его оцифровкой. Во-вторых, чувствительность человеческого глаза к зеленому цвету почти в четыре раза выше, чем к красному, и почти в десять раз выше, чем к синему. Значит, и для передачи информации о красной и синей составляющих можно использовать меньший объем памяти.

Само сжатие осуществляется алгоритмом JPEG в несколько этапов. Прежде всего производится перекодировка из RGB-модели в так называемую **YCbCr-модель.** В этой модели цвет представлен характеристиками Y — яркость зеленого цвета, Cb — цветоразность зеленый — синий, Cr — цветоразность зеленый — красный (мы не даем точного определения понятию «цветоразность», поскольку не планируем абсолютно точно описывать алгоритм JPEG, а интуитивное понимание этого слова есть, как мы надеемся,

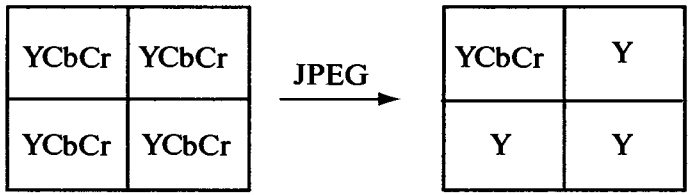


Рис. 2.10 Сжатие по алгоритму JPEG

у каждого читателя). Затем в каждом втором столбце и в каждой второй строке таблицы пикселей, заполняющих экран, информация о Cb и Cr стирается. Иными словами, для каждого квадратика из 4 пикселей только в одном остается информация о цветоразностях (рис. 2.10).

Это преобразование уменьшает объем данных в два раза: на каждые 4 пикселя вместо 12 значений передается только 6. После этого получившиеся числовые данные сжимаются еще и алгоритмом Хаффмана.

При декодировании непередаваемые характеристики цвета восстанавливаются так, чтобы у зрителя создавалось ощущение непрерывного изменения цвета. На рисунке 2.11 схематично показано, как определяются значения непередаваемых характеристик при декодировании. Сначала приближенно восстанавливаются значения для пикселей, расположенных по диагонали от пикселей с полной цветовой информацией, а затем значения для остальных пикселей.

Самый простой способ определения непередаваемых характеристик — вычисление среднего арифметического 4 известных значений. Однако если изобразить график изменения какой-либо восстанавливаемой характеристики (например, Cb) вдоль ряда пикселей с номерами 1, 2, 3, 4, 5, то вполне вероятно, что он в этом

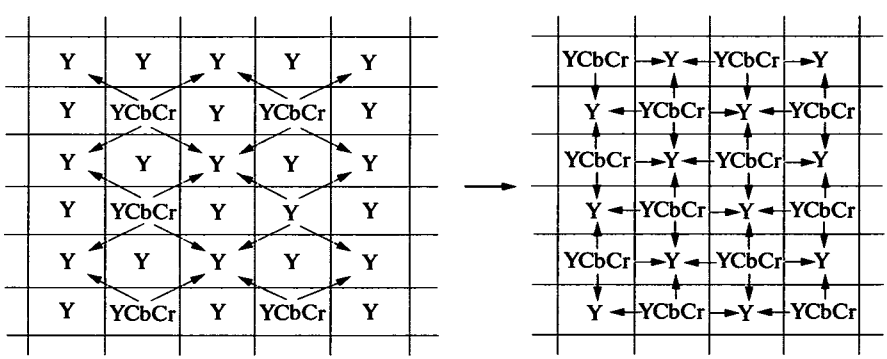


Рис. 2.11 Схема восстановления характеристик при декодировании

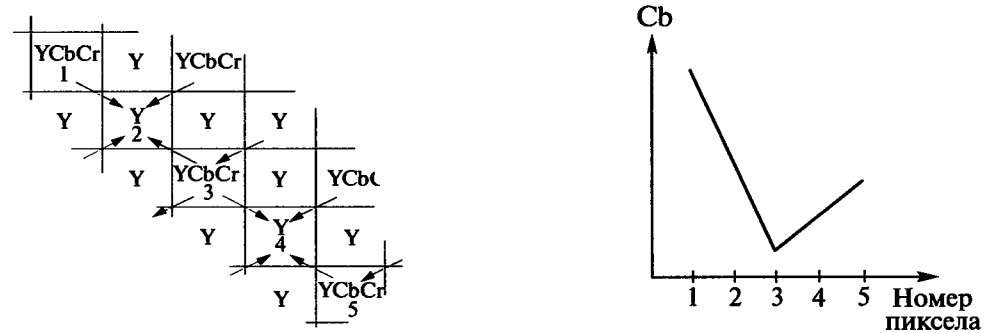


Рис. 2.12 График значений характеристики Cb вдоль ряда 1—2—3—4—5

случае будет выглядеть, например, так, как показано на рисунке 2.12. Резкая смена направления изменения значений в точке 3 характеристики Cb вряд ли имела место в оригинале изображения. Математиками разработаны специальные методы, позволяющие, как говорят, сглаживать такие переходы (рис. 2.13). На математическом языке это означает, что функция, описывающая значения характеристики, не только непрерывна, но и дифференцируема, т. е. в каждой точке имеет производную. Для построения такой функции учитываются значения как в четырех ближайших точках, так и на гораздо более обширной области, а иногда и на всем множестве точек с известными значениями функции.

Стандартизацией алгоритмов сжатия видеоданных занимается группа экспертов по видеоизображениям — Motion Picture Experts Group. Поэтому алгоритмы, удовлетворяющие стандартам, принятым этой группой, называют общим именем MPEG. Среди них и алгоритм MPEG-1 Layer 3, более известный по своему сокращенному названию MP3. На самом деле этот алгоритм предназначен для сжатия аудиоинформации. Одним из параметров, регулирующих степень сжатия, является так называемый **битрейт** (от английского bitrate) — количество бит, используемых для кодирования одной секунды звука.

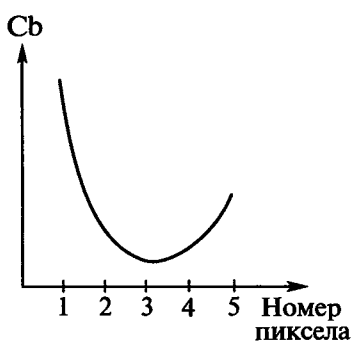


Рис. 2.13 Сглаживание функции

Звук, как вы знаете, — это колебательный процесс. Математики научились любое сложное колебание представлять в виде суммы простых колебаний, каждое из которых описывается амплитудой, частотой и фазой. Оказывается, что некоторыми составляющими можно просто пренебречь — даже самый музыкальный

слух не уловит их отсутствие. Ответственность за то, какие составляющие и в каком количестве оставить, как раз и несет битрейт. Но даже при самом большом допустимом битрейте стандарта MP3 — 320 Кбит/с — данный алгоритм обеспечивает четырехкратное сжатие информации по сравнению с форматом Audio CD при том же субъективном восприятии качества звука. На заключительном этапе к полученным данным снова применяется алгоритм Хаффмана.

Алгоритмы сжатия собственно видеоинформации используют самые разнообразные подходы. Одним из базовых является метод опорного кадра. Дело в том, что при переходе от одного кадра к следующему нередко бóльшая часть изображения остается неизменной. Если, к примеру, происходит встреча героев фильма в некоторой комнате, то ее обстановка, занимающая основную часть кадра, останется той же самой. Поэтому можно сохранять не целиком следующий кадр, а только изменения по сравнению с предыдущим кадром.

С другой стороны, если в кадре есть быстросменяемые участки, то их можно кодировать с более низким качеством — человек привык к тому, что он не может рассмотреть детали быстро изменяющихся объектов. При этом статичное изображение и динамично меняющееся изображение можно отделить друг от друга и применить к ним разные алгоритмы сжатия. И только после декодирования снова соединить их в одно изображение.

Назовем имена нескольких алгоритмов из семейства MPEG.

MPEG-1 использовался в первых Video CD (VCD-I).

MPEG-2 используется в DVD и Super Video CD (SVCD, VCD-II).

MJPEG — формат сжатия видеоизображений, в котором каждый кадр сжимается по алгоритму JPEG.

MPEG-4 — один из самых эффективных форматов сжатия видео.

DivX, XviD — улучшенные модификации формата MPEG-4.

Вопросы и задания

1. Какой алгоритм сжатия данных называется обратимым?
2. В чем состоит отличие обратимых алгоритмов сжатия от необратимых?
3. Какие особенности человеческого зрения позволяют применять необратимые алгоритмы сжатия графических изображений без потери качества?
4. Пусть V_0 — исходный объем данных, V — объем данных после обработки их сжимающим алгоритмом. Как, на ваш взгляд, обычно меняется отношение V_0 / V при увеличении V_0: возрастает, убывает или остается примерно одним и тем же?

§ 20 Обработка информации при помощи компьютера

Мы уже неоднократно говорили, что информация в памяти компьютера представлена последовательностями битов. Формальная обработка информации, на которую, собственно, только и способен компьютер, — это преобразование одной битовой последовательности в другую. Уже потом полученную результирующую последовательность битов устройство вывода декодирует в удобный для нас вид, и мы начинаем вникать в смысл полученного сообщения или любоваться графическим изображением.

Рис. 2.14 Схема электромагнитного реле

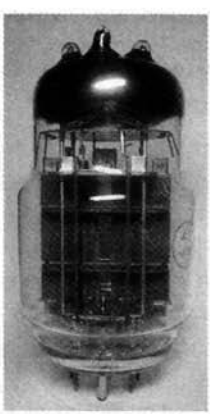


Рис. 2.15 Электронная лампа

Физическая подоплека использования двоичного кодирования состоит в том, что 2 символа легко реализуются при помощи технических средств, например электрического тока или светового луча. Цифра 0 двоичной системы счисления может означать, что ток (луч) не проходит, а цифра 1 тогда будет означать, что ток (луч) проходит. При таком представлении цифр действия над числами производятся подходящими комбинациями включений и выключений тока или света. Поэтому любую электронную вычислительную машину можно представлять себе как совокупность соединенных между собой выключателей тока (или света). Отличие электронного выключателя от выключателя настольной лампы состоит в том, что в электронном выключателе нет механических движущихся частей и переключается он не рукой человека, а электрическим сигналом от другого выключателя. Время переключения поэтому оказывается очень малым, порядка 10^{-9} с.

Самый простой прибор, осуществляющий такую операцию, — электромагнитное реле — схематично изображен на рисунке 2.14. Цепь разомкнута до тех пор, пока на обмотку железного сердечника не подано напряжение. В этот момент в сердечнике создается магнитное поле, притягивающее один конец вращающегося на шарнире рычажка. Другой его конец в этот момент сжимает контакты: цепь замыкает-

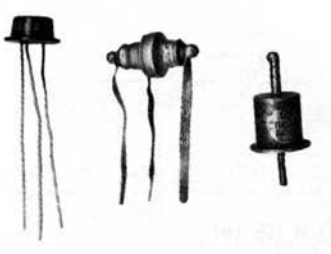


Рис. 2.16 Полупроводниковые триоды

Рис. 2.17 Микросхемы

ся. На принципах такого переключения работал один из первых компьютеров — МАРК-1.

В дальнейшем в качестве переключателей стали использовать электронные лампы-триоды (рис. 2.15). Их действие вы, вероятно, изучали на уроках физики. Потом настала пора полупроводниковых триодов (рис. 2.16). А затем и планарных транзисторов, они идентичны по своему действию плоскостным, но не превышают в длину сотой доли сантиметра. Современная технология изготовления этих приборов позволяет размещать на поверхности одной микросхемы несколько миллионов транзисторов. Внешний вид микросхем можно увидеть на рисунке 2.17.

Важно не только иметь миниатюрные электронные переключатели, но и знать, как их соединить между собой, чтобы с их помощью выполнять арифметические действия. Если в вашем распоряжении оказались два переключателя, то имеется ровно два способа соединить их между собой (рис. 2.18).

Первый вариант соединения называется **последовательным**, а второй — **параллельным**. В первом случае ток в цепи идет (лампочка горит) тогда и только тогда, когда включены оба переключателя. Во втором случае для прохождения тока в цепи (лампочка горит) достаточно, чтобы включен был хотя бы один переключатель. Кроме того, рассматривают еще один вариант: лампочка горит тогда и только тогда, когда переключатель выключен (рис. 2.19).

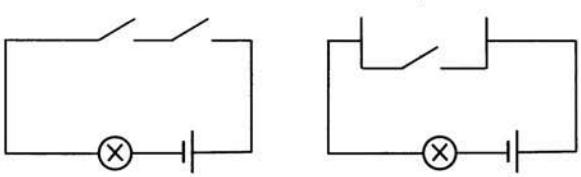


Рис. 2.18 Последовательное (*а*) и параллельное (*б*) соединение переключателей

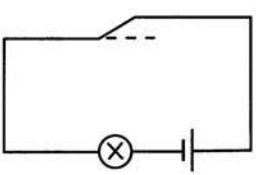


Рис. 2.19

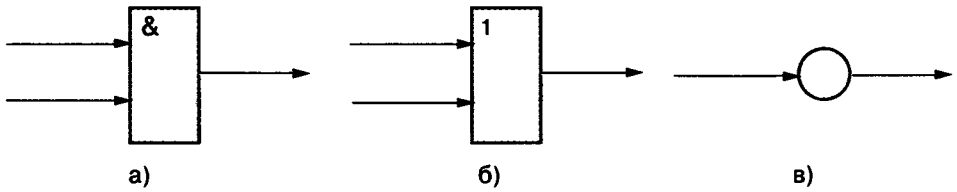


Рис. 2.20 Условное изображение вентилей И (*а*), ИЛИ (*б*) и НЕ (*в*)

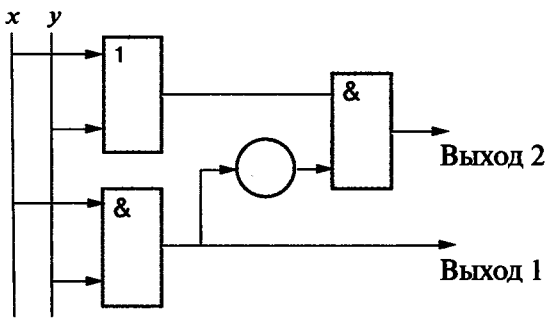


Рис. 2.21

Таблица 2.14

x	*y*	Выход 1	Выход 2
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Конструкции, изображенные на рисунках 2.18 и 2.19, называются **вентилями.** Можно условиться называть их «вентиль первого типа», «вентиль второго типа» и т. д. Но столь обезличенные названия не отражают специфику их работы. Поскольку первый вентиль зажигает лампочку только тогда, когда замкнут *и* первый переключатель, *и* второй, его назвали вентилем И. Второй вентиль зажигает лампочку, когда замкнут *или* первый переключатель, *или* второй, его назвали вентилем ИЛИ. Третий вентиль зажигает лампочку тогда и только тогда, когда переключатель *не* замкнут, он называется вентиль НЕ. Будем условно изображать вентили так, как показано на рисунке 2.20.

Из вентилей соберем схему, приведенную на рисунке 2.21. И для этой схемы составим таблицу результатов (см. табл. 2.14).

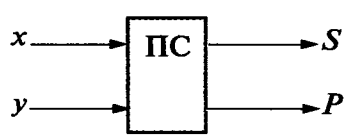


Рис. 2.22 Полусумматор

Сравнивая получившуюся таблицу с таблицей сложения однозначных чисел в двоичной системе счисления, приходим к выводу, что наше устройство на выходах дает два сигнала, которые поразрядно кодируют сумму двух однозначных чисел в двоичной системе счисления. А поскольку действия над числами, записанными в позиционной системе, выпол-

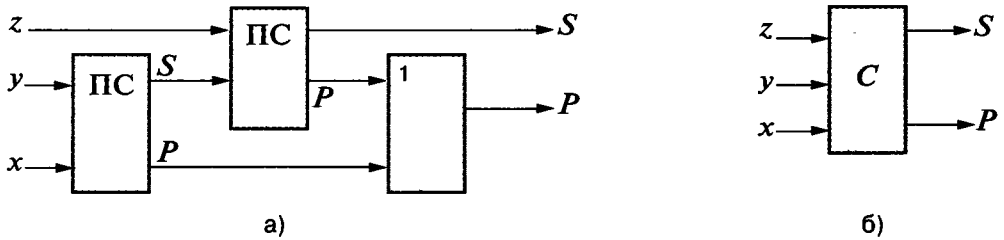


Рис. 2.23 Схема сумматора (*а*) и его обозначение в виде блока (*б*)

няются поразрядно, то аналогичным образом можно построить электронные схемы для сложения многозначных чисел, представленных в двоичной системе счисления. Электронную схему, изображенную на рисунке 2.21, называют **полусумматором**. В дальнейшем для краткости полусумматор будем изображать одним блоком (рис. 2.22). В нем буквой S обозначен младший разряд суммы, а буквой P — старший разряд, или, по-другому, перенос единицы в следующий разряд суммы.

При сложении многозначных чисел в старших разрядах приходится учитывать появление так называемой единицы переноса. Это означает, что в этих разрядах складываются не два однозначных числа, а три. Схему сумматора для сложения чисел в двоичной системе счисления можно изобразить так, как показано на рисунке 2.23. Здесь x и y — единицы разрядов слагаемых, а z — перенос из суммы предыдущих разрядов; выходы S и P имеют тот же смысл, что и для полусумматора.

Чтобы сложить два многозначных числа, нужно выстроить батарею сумматоров так, как показано на рисунке 2.24. А на рисунке 2.25 показана работа такой батареи для чисел 100101 и 1011.

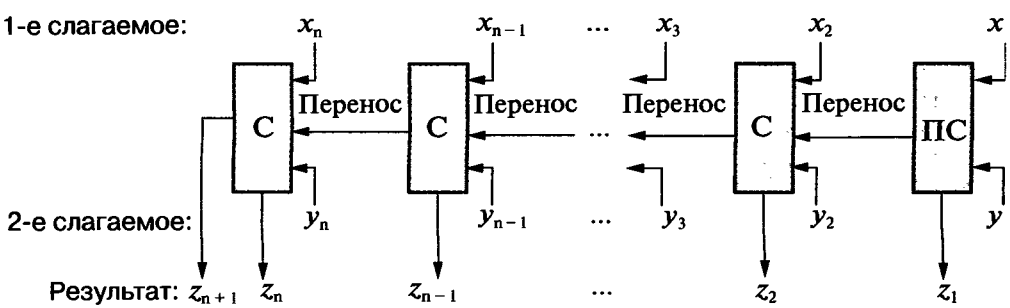


Рис. 2.24 Батарея сумматоров для сложения двух n-разрядных чисел $x_n x_{n-1} x_3 x_2 x_1$ и $y_n y_{n-1} y_3 y_2 y_1$

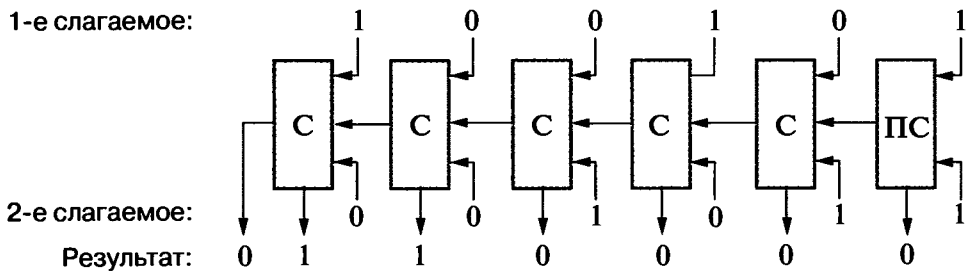


Рис. 2.25 Сложение чисел 100101 и 1011

В современном компьютере никто, конечно, не выстраивает подобных батарей. Они входят составной частью в ту или иную микросхему, которая обеспечивает выполнение не только операции сложения, но и целого комплекса операций по обработке двоично-закодированной информации.

Вопросы и задания

1. Почему в электронной вычислительной технике обычно используется двухсимвольное кодирование?

2. а) Обозначим в вентиле И (см. рис. 2.20, *а*) один вход буквой *x*, другой — буквой *y*, а выход буквой *z*. Заполните таблицу 2.15, показывающую, как значение *z* зависит от значений *x* и *y*.

   б) Выполните такое же задание для вентилей ИЛИ и НЕ.

   в) Как, по-вашему, выглядит «двоичный мультипликатор» — устройство для перемножения двух однозначных двоичных чисел?

3. Для схемы, изображенной на рисунке 2.26, составьте таблицу, показывающую зависимость значения *z* от значений *x* и *y*.

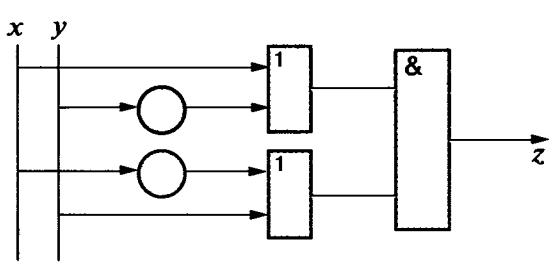


Рис. 2.26

Таблица 2.15

x	*y*	*z*
0	0	
1	0	
0	1	
1	1	

4 Чем сумматор отличается от полусумматора?

5 Используя вентили, сумматоры и полусумматоры, постройте схему мультипликатора для умножения двух двузначных двоичных чисел. (Совет. Перечитайте в § 10 правило умножения многозначных чисел в двоичной системе счисления.)

§ 21 Булевы функции

Теоретической базой для описания работы компьютера служит математическая теория булевых функций. Как уже было сказано, вся информация кодируется последовательностями из двух символов; можно считать, что это 0 и 1. Если длина исходной последовательности n, то можно считать, что мы имеем дело с n переменными x_1, x_2, ..., x_n, каждая из которых независимо от другой принимает значение 0 или 1. Обработка такой последовательности состоит в том, что ей ставится в соответствие другая последовательность из нулей и единиц. Каждый элемент новой последовательности можно рассматривать как значение подходящей функции от переменных x_1, x_2, ..., x_n. Если в результирующей последовательности m элементов, то, значит, у нас m функций. В таблицах 2.16 и 2.17 приведены примеры, в которых исходная последовательность двухэлементна.

Функцию $f(x_1, x_2)$, представленную таблицей 2.16, записывают обычно как $x_1 \vee x_2$ и называют **дизъюнкцией** переменных x_1, x_2. Другое название для нее — **логическая операция ИЛИ**. Именно такой будет эта операция, если считать, что истинное высказывание кодируется единицей, а ложное — нулем.

Функции $f_1(x_1, x_2)$ и $f_2(x_1, x_2)$, описанные в таблице 2.17, реализуют сложение двух однозначных чисел x_1 и x_2 в двоичной арифметике, при этом функция $f_1(x_1, x_2)$ отвечает за старший разряд суммы, а функция $f_2(x_1, x_2)$ — за ее младший разряд.

Таблица 2.16

x_1x_2	$f(x_1, x_2)$
00	0
01	1
10	1
11	1

Таблица 2.17

x_1x_2	$f_1(x_1, x_2)$	$f_2(x_1, x_2)$
00	0	0
01	0	1
10	0	1
11	1	0

Функции, аргументы которых принимают значения только 0 и 1 и которые сами принимают только такие же значения, называются **булевыми** (в честь Дж. Буля, заложившего основы математической логики).

Легко видеть, что булевых функций от одной переменной только четыре; все они представлены в таблице 2.18.

Первая из этих функций называется **тождественной единицей**, вторая — просто **тождественной**, третья — **отрицанием** или **логической функцией НЕ**, четвертая — **тождественным нулем**. Отметим, что обычно вместо $\varepsilon(x)$ пишут просто x, а функцию $\nu(x)$ обозначают как $\bar{x}$ или $\neg x$.

Функций от двух переменных уже шестнадцать. Действительно, двухсимвольных последовательностей, составленных из 0 и 1, имеется четыре (все они записаны в первом столбце таблиц 2.16 и 2.17). Две функции на двухсимвольных последовательностях, т. е. от двух аргументов, различаются, если они принимают разные значения хотя бы на одной такой последовательности. Всего же различных способов сопоставить четырем элементам один из двух символов — 0 или 1 — имеется 2^4, т. е. 16. В таблице 2.19 приведены все шестнадцать булевых функций от двух аргументов; вместо безликой буквы f даны употребительные обозначения этих функций. Для функций двух переменных знак функции обычно пишут не слева от аргументов, а между ними — мы ведь издавна пишем $x + y$, а не $+(x, y)$. Впрочем, такая запись, которую обычно называют **префиксной**, хотя и непривычна, весьма удобна. Скобки в ней необязательны — можно просто писать $+x, y$. И даже если написать более длинное выражение, скажем $: + x, y, z$, то оно однозначно расшифровывается как $(x + y) : z$. Выражению же $x + y : z$ будет соответствовать запись $+ x, : y, z$. Можно избрать и другой вариант — записывать знак операции после аргументов. Такую запись, как правило, называют **обратной польской записью**. Она особенно удобна для анализа и оптимизации процессов вычислений.

Приведем общепринятые названия некоторых функций из таблицы 2.19 (не повторяя тех, которые названы ранее).

Функция $x_1 \& x_2$ называется **конъюнкцией** переменных x_1, x_2. Другое название для нее — **логическая операция И.**

Функция $x_1 \to x_2$ называется **импликацией.**

Функция $x_1 \sim x_2$ называется **эквиваленцией.**

Функция $x_1 \oplus x_2$ называется **сложением по модулю 2.**

Функция $x_1 | x_2$ называется **операцией Шеффера.**

Таблица 2.18

x	λ	$\varepsilon(x)$	$\nu(x)$	o
0	1	0	1	0
1	1	1	0	0

Таблица 2.19

x_1x_2	λ	$x_1 \vee x_2$	$x_2 \to x_1$	$x_1 \to x_2$	$x_1 \mid x_2$	x_1	x_2	$x_1 \oplus x_2$
00	1	0	1	1	1	0	0	0
01	1	1	0	1	1	0	1	1
10	1	1	1	0	1	1	0	1
11	1	1	1	1	0	1	1	0

x_1x_2	$x_1 \sim x_2$	$\bar{x}_2$	$\bar{x}_1$	$x_1 \& x_2$	$x_1 \not\to x_2$	$x_2 \not\to x_1$	$x_1 \uparrow x_2$	0
00	1	1	1	0	0	0	1	0
01	0	0	1	0	0	1	0	0
10	0	1	0	0	1	0	0	0
11	1	0	0	1	0	0	0	0

Функция $x_1 \uparrow x_2$ называется **операцией Пирса.**

Мы привели эти названия не для того, чтобы вы их заучивали. Просто само наличие названий показывает, что эти функции попали под пристальное внимание математиков. Употребление термина «операция» тоже не случайно — функцию двух переменных нередко называют операцией над этими переменными (вспомните: операция вычитания, операция умножения и т. д.).

Обсудим теперь, как можно использовать функции, кроме того, что вычислять их значение по заданным значениям аргументов.

Каждую функцию (совсем необязательно булеву) можно представлять себе как некое устройство по переработке значений аргументов в значение функции. Как именно работает данное устройство, нас, вообще говоря, не интересует. Функцию двух переменных можно схематически изобразить, например, так, как показано на рисунке 2.27. Аргументы в этом случае называют **входами** данного устройства, а значение функции подается на его **выход.**

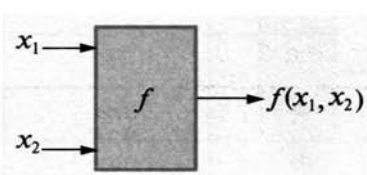


Рис. 2.27 Функция как устройство по обработке данных

Функции, физически реализованные в виде таких устройств, называ-

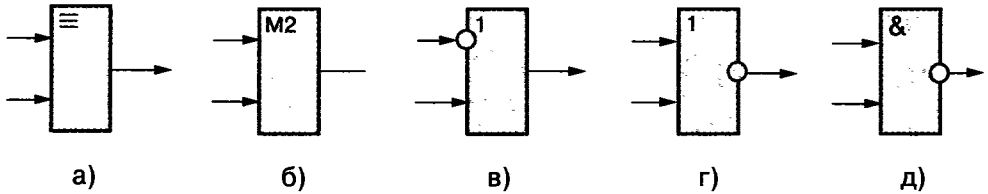


Рис. 2.28 Стандартные изображения вентилей: а) эквиваленция; б) сравнение по модулю 2; в) импликация; г) операция Пирса; д) операция Шеффера

ют вентилями; функции И, ИЛИ и НЕ, как было сказано в § 4, называют соответственно вентилем НЕ, вентилем И, вентилем ИЛИ. Их стандартные изображения были приведены на рисунке 2.20. На рисунке 2.28 мы приводим стандартные (ГОСТ 2.745—91) изображения остальных упомянутых выше функций.

Обратите внимание на небольшие кружочки в трех последних изображениях устройств. Они означают, что данное устройство эквивалентно комбинации двух устройств (рис. 2.29), одно из которых отрицание.

Различные вентили можно комбинировать, подавая на входы одной функции то, что получилось на выходах других функций. Пример такой комбинации приведен на рисунке 2.30.

Получившееся устройство естественно рассматривать как функцию трех переменных x_1, x_2, x_3. Эту функцию записывают как $f_2(f_1(x_1, x_2), x_3)$. Подстановку одной функции в другую вместо аргумента называют **композицией** данных функций.

Если в функцию от одной переменной подставить снова функцию от одной переменной, то получится функция от одной переменной. Если же есть хотя бы одна функция от двух переменных, то, как показано выше, можно получить функцию с тремя и вообще с любым числом переменных.

Оказывается, что любую булеву функцию можно получить при помощи композиции некоторого конечного набора булевых функций. Иными словами, любую обработку двоично закодированной информации можно произвести, собирая различные конструкции из раз и навсегда заданного конечного набора элементов.

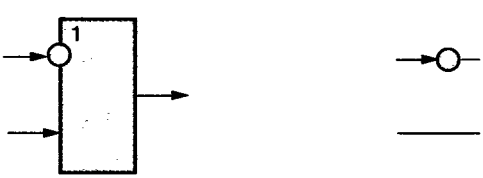


Рис. 2.29 Вентиль и эквивалентное ему сочетание устройств

Более того, в качестве таких элементов можно взять отрицание, конъюнкцию и дизъюнкцию. Но прежде чем дать обоснование утверждению о

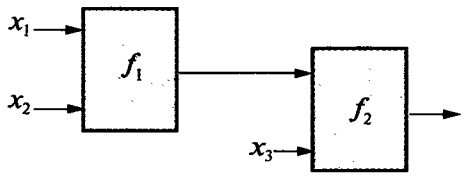


Рис. 2.30 Композиция $f_2(f_1(x_1, x_2), x_3)$ функций f_1 и f_2

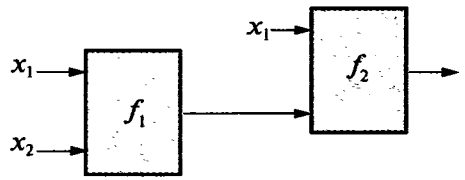


Рис. 2.31 Композиция $f_2(x_1, f_1(x_2, x_3))$ функций f_1 и f_2

том, что этих функций достаточно для получения всех булевых функций, необходимо сделать одно замечание.

Результат композиции двух функций, как правило, зависит от того, вместо какого аргумента делается подстановка: наряду с композицией, изображенной на рисунке 2.30, можно рассматривать композицию, представленную на рисунке 2.31.

Даже когда f_1 и f_2 — это одна и та же функция, результат подстановки может зависеть от выбора аргумента, на место которого осуществляется подстановка. Легко убедиться, например, что $(x_1 \to x_2) \to x_3$ и $x_1 \to (x_2 \to x_3)$ — различные функции: они принимают разные значения на последовательности 010. А вот если f_1 и f_2 — это конъюнкции, то результат подстановки не зависит от выбора аргумента: получающиеся функции $(x_1 \,\&\, x_2) \,\&\, x_3$ и $x_1 \,\&\, (x_2 \,\&\, x_3)$ совпадают, в чем тоже несложно убедиться, вычислив значения каждой из них для всевозможных трехэлементных последовательностей из чисел 0 и 1 (см. табл. 2.20). Аналогично можно убедиться в совпадении функций $(x_1 \vee x_2) \vee x_3$ и $x_1 \vee (x_2 \vee x_3)$. Это позволяет не писать вообще скобки в выражениях $x_1 \,\&\, x_2 \,\&\, x_3$ и $x_1 \vee x_2 \vee x_3$, поскольку любая их расстановка дает один и тот же результат. Именно так мы и будем поступать в дальнейшем.

Таблица 2.20

$x_1x_2x_3$	$(x_1 \,\&\, x_2) \,\&\, x_3$	$x_1 \,\&\, (x_2 \,\&\, x_3)$
000	0	0
001	0	0
010	0	0
011	0	0
100	0	0
101	0	0
110	0	0
111	1	1

Вернемся к обоснованию высказанного утверждения о том, что любая булева функция может быть получена композицией тож-

Таблица 2.21

$x_1x_2...x_n$	$f(x_1, x_2, ..., x_n)$
00...0	...
00...1	...
...	...
11...0	...
11...1	...

дественной функции, отрицания, конъюнкции и дизъюнкции. Фактически для этого надо показать, что каждая булева функция может быть записана как выражение, в которое входят лишь эти четыре операции.

Пусть $f(x_1, x_2, ..., x_n)$ — какая-либо булева функция от переменных $x_1, x_2, ..., x_n$. Мы построим нужное выражение в три шага.

Шаг 1. Построим таблицу значений данной функции для всевозможных значений переменных по образцу таблицы 2.21.

Шаг 2. Выберем в этой таблице те строки, в которых значение функции равно 1. Для каждой выбранной строки записываем конъюнкцию, составленную из функций ε и ∨ по следующему правилу: если значение x_k равно 1, то пишем x_k; если значение x_k равно 0, то пишем $\bar{x}_k$. Например, для последовательности 100101 запишем такое выражение:

$$x_1 \& \bar{x}_2 \& \bar{x}_3 \& x_4 \& \bar{x}_5 \& x_6.$$

Если на этом шаге выбрано более одной строки, то каждое полученное выражение заключаем в скобки, после чего переходим к шагу 3. Если была выбрана только одна строка, то полученное выражение и есть нужное представление функции $f(x_1, x_2, ..., x_n)$.

Шаг 3. Все выражения, составленные на шаге 2, соединяются знаком дизъюнкции. Полученное выражение представляет исходную функцию $f(x_1, x_2, ..., x_n)$.

Из записи выражения для функции $f(x_1, x_2, ..., x_n)$ видно, что она получается композицией некоторого количества тождественных функций, отрицаний, конъюнкций и дизъюнкций.

Таким образом, промышленности достаточно освоить выпуск всего трех логических элементов — элемента НЕ, элемента И, элемента ИЛИ — и уже из них можно собирать схему для вычисления любой булевой функции. На самом деле можно еще уменьшить разнообразие необходимых элементов: любая функция может быть получена из одной только операции Пирса или операции Шеффера. Иными словами, можно выпускать лишь один какой-то логический элемент и из него конструировать все вычислительные устройства.

Но уменьшение разнообразия выпускаемых элементов вовсе не так экономично, как это может показаться на первый взгляд. Дело в том, что для получения нужной схемы с использованием

одного универсального элемента может потребоваться гораздо больше элементов, чем при применении элементов нескольких видов. Гораздо эффективнее оказывается иметь готовые логические элементы (микросхемы) разных видов и уже из них собирать сложные схемы. Кроме того, одна и та же функция может быть реализована разными выражениями и, значит, разными схемами. Общей теории построения схем с наименьшим числом элементов на сегодняшний день не существует, хотя и есть отдельные алгоритмы, позволяющие упрощать схемы. Большой вклад в создание таких алгоритмов внесли российские ученые Ю. И. Журавлев, С. В. Яблонский и др.

Вопросы и задания

1. Какую функцию называют булевой?

2. а) Ассоциативность операции & проверена в объяснительном тексте параграфа (см. табл. 2.21). На форзаце приведено еще несколько законов для булевых операций. Проверьте их, составив аналогичные таблицы значений.

   б) На форзаце приведены только наиболее часто применяемые законы, выражающие свойства булевых операций. Ниже мы приводим еще несколько формул. Проверьте их справедливость.

   $\bar{x} = \lambda \oplus x$; $\quad x \to y = \bar{x} \vee y$;

   $x \oplus x = 0$; $\quad x \vee y = x \oplus y \oplus x \,\&\, y$.

3. Из вентилей И, ИЛИ, НЕ постройте схему по заданному выражению булевой функции:

   а) $\overline{X \to Y} \vee ((\bar{X} \vee Y) \,\&\, X)$;

   б) $\overline{Z \to Y} \,\&\, (\bar{X} \vee Y) \,\&\, Z$;

   в) $\overline{Z \,\&\, Y} \vee \bar{X} \,\&\, (Y \vee \bar{Z})$.

4. Составьте формулу, описывающую схему, изображенную на рисунке 2.32.

5. а) Составьте таблицу значений булевой функции, реализованной схемой, изображенной на рисунке 2.32.

   б) Какая булева функция из перечисленных в таблице 2.19 эквивалентна булевой функции, реализованной данной схемой?

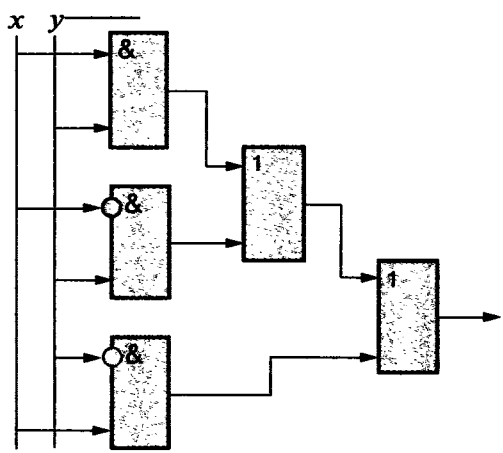


Рис. 2.32

6 Составьте формулу, описывающую схему, изображенную на рисунке 2.33.

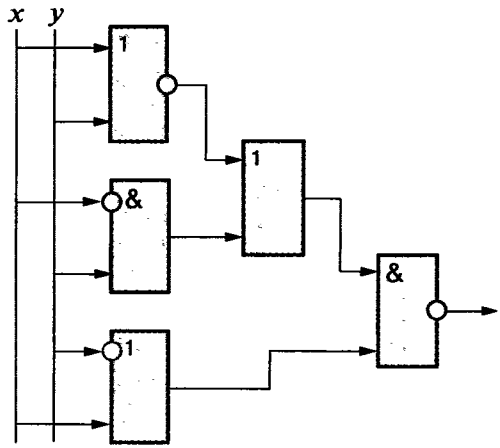


Рис. 2.33

7 а) Составьте таблицу значений булевой функции, реализованной схемой, изображенной на рисунке 2.33.

б) Какая булева функция из перечисленных в таблице 2.19 эквивалентна булевой функции, реализованной данной схемой?

8 Таблица 2.22 демонстрирует логику работы сумматора: x и y — разрядные единицы слагаемых, z — единица переноса из предыдущего разряда.

а) Составьте формулы для пары булевых функций (разрядной единицы суммы и единицы переноса) от трех переменных x, y и z, описывающих работу сумматора.

б) Изобразите схему, содержащую вентили И, ИЛИ и НЕ, реализующую работу сумматора. Попытайтесь составить схему, содержащую как можно меньше указанных вентилей.

Таблица 2.22

x	y	z	Единица переноса в следующий разряд	Разрядная единица результата
0	0	0	0	0
0	1	0	0	1
0	0	1	0	1
0	1	1	1	0
1	0	0	0	1
1	1	0	1	0
1	0	1	1	0
1	1	1	1	1

в)* Попытайтесь составить самую экономную схему сумматора, в которой разрешается использовать еще и вентили, изображенные на рисунке 2.28.

9 а) Проверьте, что $x \uparrow x = \bar{x}$ и $(x_1 \uparrow x_2) \uparrow (x_1 \uparrow x_2) = x_1 \vee x_2$.
б) Выразите через операцию Пирса функцию $x_1 \& x_2$.

10 Выразите через операцию Шеффера отрицание, конъюнкцию и дизъюнкцию.

§ 22 Логика оперативной памяти

Основу оперативной памяти компьютера составляют элементы, обладающие двумя устойчивыми состояниями и меняющие их по внешнему сигналу. В первых ЭВМ такие элементы представляли собой ферритовые кольца, которые перемагничивались при подаче на их обмотку соответствующего напряжения. В современных компьютерах оперативная память состоит из электронных элементов, которые по-прежнему называются **триггерами** (от английского trigger — защелка); впрочем, в английском языке чаще употребляется термин *flip-flop*, звукоподражательно обозначающий что-то вроде *щелчок-хлопок*, которым как бы сопровождается мгновенный переход из одного состояния в другое. Конечно, в электронных схемах никакого хлопка услышать невозможно.

Мы расскажем об одном из самых распространенных типов триггера — так называемом ***SR*-триггере** (от английских Set — установка и Reset — сброс). На рисунке 2.34 изображена логическая схема *SR*-триггера.

Физически такой триггер имеет два входа — S и R, а также два выхода — Q и $\bar{Q}$. Такое обозначение выходов не случайно: они всегда имеют противоположные значения. Так что фактически результатом является одна независимая величина — Q.

Для триггера процесс управления является дискретным: напряжение на его входы подается не непрерывно, а импульсами. В реальной схеме, для того чтобы на каждом такте на входы (в том числе и с выходов Q) импульсы подавались одновременно, после НЕ-элементов ставится синхронизирующий элемент задержки. Каждый импульс кодирует символ 1.

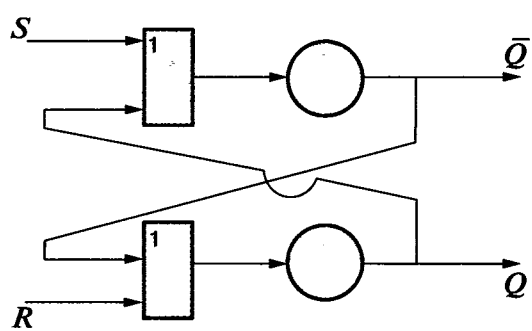


Рис. 2.34 Логическая схема *SR*-триггера

Составим таблицу значений выходов Q и $\bar{Q}$ в зависимости от значений на входах (см. табл. 2.23).

Тут-то и возникает противоречие: в двух последних строках значения Q и $\bar{Q}$ оказались равными, вместо того чтобы быть противоположными. Поэтому на входы SR-триггера запрещается одновременно подавать значения $S = 1$ и $R = 1$. Из приведенной таблицы следует, что подача 1 на вход S устанавливает триггер в положение 1, а подача 1 на вход R сбрасывает его значение до 0. Если же никаких импульсов не поступает или продолжает поступать импульс на тот же вход (R или S), то состояние триггера не меняется. Это и означает, что триггер помнит информацию.

Другую разновидность триггеров составляют так называемые ***JK*-триггеры**. Схема JK-триггера приведена на рисунке 2.35.

В таблице 2.24 приведены значения выходов Q и $\bar{Q}$ в зависимости от значений на входах. Как видно из таблицы, выходы Q и $\bar{Q}$ здесь всегда согласованы, но теперь при одновременной подаче импульсов (т. е. 1) на входы J и K значения Q и $\bar{Q}$ меняются на противоположные. Следовательно, на следующем такте работы триггера они снова меняются на противоположные и т. д. Так что при подаче импульсов на оба входа J и K данное устройство ведет себя неустойчиво. Поэтому и для такого триггера одновременная

Таблица 2.23

Входы			Выходы	
S	R	Q	Q	$\bar{Q}$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	0
1	1	1	0	0

Таблица 2.24

Входы			Выходы	
K	J	Q	Q	$\bar{Q}$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

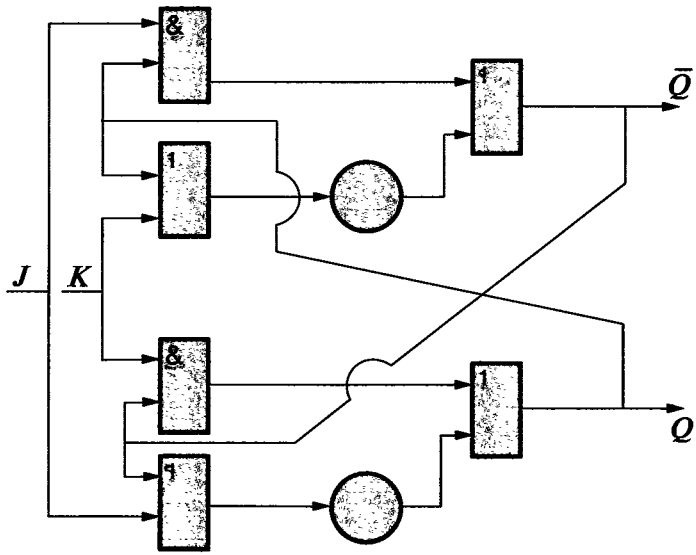


Рис. 2.35 Логическая схема *JK*-триггера

подача 1 на оба входа запрещена. В остальном *JK*-триггер ведет себя так же, как *SR*-триггер.

Поскольку триггер запоминает лишь 1 бит, оперативная память представляет собой батареи таких триггеров, к которым и обращается за информацией процессор. Довольно редко обращение идет к 1 биту; как правило, для хранения информации предоставляется последовательность определенной длины, состоящая из триггеров и образующая **ячейку** компьютерной памяти. Наибольшая длина последовательности, которая может быть сохранена в ячейке, называется **разрядностью** ячейки.

Все ячейки пронумерованы, начиная с нуля; номер ячейки называется ее **адресом**. Любое обращение к памяти компьютера для занесения туда информации или извлечения ее оттуда процессор осуществляет, манипулируя с адресами ячеек. В любую ячейку записан некоторый набор нулей и единиц — так называемое **машинное слово**. Введение этого термина обусловлено тем, что компьютер сам по себе не знает, что именно хранится в ячейке памяти: число в том или ином формате, команда или какая-то служебная информация. Более того, одно и то же машинное слово в разные моменты работы компьютера может им по-разному интерпретироваться.

Однородность памяти — в этом состоит один из принципов фон Неймана, сформулированных в историческом докладе в 1945 г. Именно в этом докладе трое авторов (идейным лидером авторского

коллектива и был Дж. фон Нейман), отвлекшись от радиоламп и электрических схем, сумели описать формальную организацию компьютера.

Процессор тоже имеет несколько собственных ячеек памяти. Эти ячейки называют **регистрами**. Они служат для хранения и обработки информации, необходимой при выполнении процессором очередной операции. **Разрядностью процессора** называют максимальную длину машинного слова, которое может обрабатываться и передаваться процессором как единое целое. Разрядность процессора определяется разрядностью его регистров. В настоящее время используются 8-, 16-, 32- и 64-разрядные процессоры.

Вопросы и задания

1. Для чего предназначен триггер?
2. Почему запрещена одновременная подача импульса на оба входа триггера?
3. Что такое ячейка памяти? Что такое регистр процессора?
4. Чем определяется разрядность процессора?

§ 23 Представление целых чисел в памяти компьютера

Физически память компьютера — это пронумерованная совокупность ячеек. В свою очередь, каждая ячейка состоит из восьми устройств, каждое из которых может находиться в одном из двух состояний. Одно из состояний кодируется символом 0, а другое — символом 1. Тем самым в одной ячейке размещается 1 байт информации.

Содержимое каждой ячейки можно рассматривать как запись натурального числа в двоичной системе счисления. При этом коду 00000000 сопоставляется число 0.

Легко подсчитать, что такое кодирование позволяет записать целые числа от 0 до 255 включительно. Ведь самое большое число — это $11111111_2 = 2^8 - 1 = 255$.

Кроме положительных целых чисел, существуют отрицательные. Чтобы указать знак числа, нужен еще один бит. При этом договариваются, что его нулевое значение соответствует знаку «+», а единичное значение — знаку «−». Обычно под знак выделяют са-

мый левый бит ячейки. Тогда код наибольшего натурального числа, которое можно записать в одной ячейке, — это 01111111; ему соответствует десятичное число +127. А код наименьшего отрицательного числа — это 11111111; в десятичной записи ему соответствует число −127. Рассмотренный код называют **прямым.**

Чтобы найти сумму чисел 101_2 и 1011_2, с их кодами можно действовать по правилам сложения двоичных чисел:

$$00000101 + 00001011 = 00010000.$$

Попытаемся теперь сложить положительное число с отрицательным. Например, число 1010_2 с числом -101_2. Их коды соответственно таковы: 00001010 и 10000101. Результатом должно быть натуральное число 101_2. Даже алгоритм получения кода результата сформулировать не так уж просто (попытайтесь такое проделать), а тем более реализовать простое устройство, которое бы этот алгоритм исполняло. А ведь компьютеру приходится выполнять сложение довольно часто, так что все должно быть как можно проще и быстрее.

Для того чтобы выполнение операции сложения было проще и не зависело от знака слагаемых, отрицательные целые числа кодируют другим способом. Но сначала поговорим о нуле.

Что будет, если в восьмиразрядную ячейку попытаться записать натуральное число 100000000_2? Все восемь разрядов окажутся нулями. Значит, компьютер воспринимает это число как 0. Этим-то мы и воспользуемся. Вычтем из числа 100000000_2 число 101_2. Получится число 11111011_2. Если теперь его сложить с числом 101_2, то получим результат 100000000_2, компьютер воспримет его как 0. Поэтому число 11111011_2 объявляют кодом отрицательного числа -101_2. Его называют **дополнительным кодом** данного отрицательного числа.

Рассмотрим код 10000000. Во-первых, это код отрицательного числа, поскольку в самом левом разряде стоит 1. Компьютер воспринимает его как дополнительный код. Тогда прямой код противоположного ему положительного числа совпадает с разностью $100000000_2 - 10000000_2$. Она равна 10000000_2, т. е. 128_{10}. Таким образом, наименьшее отрицательное число, которое можно записать в восьмибитовую ячейку, — это −128.

Кодом числа 0 является 00000000_2. А что произойдет, если записать дополнительный код для числа −0? Вычитаем из 100000000_2 число 0_2 и получаем 100000000_2. При записи в ячейку снова окажется 00000000_2. Так что компьютер, как и мы с вами, не различает числа +0 и −0.

Для вычисления дополнительного кода отрицательного числа n можно каждый раз вычитать из 100000000_2 модуль числа n.

Но есть и другой способ. Вот как можно поступать, чтобы получить дополнительный код отрицательного числа:

1. Записать двоичный код модуля числа.
2. В полученной записи каждую цифру 1 заменить цифрой 0, а каждую цифру 0 — цифрой 1.
3. К полученному коду, рассматриваемому как натуральное число в двоичной системе, прибавить 1.

Как видите, построение дополнительного кода осуществляется легко. И поскольку вычитание можно заменить сложением с противоположным числом, переход к дополнительному коду числа позволяет обойтись без вычитания.

Диапазон от −128 до +127 во многих случаях мал для решения возникающих задач. Но вовсе необязательно хранить целое число ровно в одной восьмибитовой ячейке. Обычно для целых чисел отводится две ячейки — именно так происходит, если вы объявляете тип переменной как Integer. Если же объявить тип LongInt, то под целое число будет отведено 4 восьмибитовые ячейки. А вот одна ячейка отводится, если объявлен тип ShortInt.

Если под запись числа отведено две ячейки, то они воспринимаются как единое целое. Принцип же кодирования тот же: самый левый разряд отводится под знак числа, остальные пятнадцать разрядов — для кода абсолютной величины числа. Для положительных чисел используется прямой код, для отрицательных — дополнительный. Тем самым диапазон целых чисел таков: от −32768 до +32767 = $2^{15} - 1$.

Если для записи целых чисел используется m-битный двоичный код, то диапазон кодируемых чисел от -2^{m-1} до $2^{m-1}-1$, при этом дополнительный код отрицательного числа n из этого диапазона совпадает с записью в двоичной системе числа $2^m + n$.

Вопросы и задания

1. Что такое дополнительный код? Каковы преимущества использования дополнительного кода?

2. Каков диапазон целых чисел, для кодирования которых используются 4 ячейки?

3. Объясните, почему приведенный в объяснительном тексте параграфа способ получения дополнительного кода отрицательного числа действительно дает верный результат.

4 а) Следующие числа записаны в прямом восьмибитном коде: 01101010; 10111011; 10001001; 01010111; 11111111. Найдите среди них отрицательные числа и запишите их в дополнительном коде.

б) Следующие числа записаны в прямом шестнадцатибитном коде: 0110101010111011; 1011101110111011; 1011101110001001; 0101011110111011; 1111111111111111. Найдите среди них отрицательные числа и запишите их в дополнительном коде.

5 а) Переведите в двоичную систему счисления и запишите в дополнительном восьмибитном коде числа –10; –101; –68.

б) Переведите в двоичную систему счисления и запишите в дополнительном шестнадцатибитном коде числа –351; –7843; –26 175.

6 а) Запишите в десятичной системе счисления отрицательные числа, заданные в дополнительном восьмибитном коде: 10101101; 11011010; 11110000.

б) Запишите в десятичной системе счисления отрицательные числа, заданные в дополнительном шестнадцатибитном коде: 1010110110101101; 1101101101101010; 1111110110100000.

7 Как по дополнительному коду отрицательного числа узнать, четно оно или нечетно?

8 Числа 01001010; 10010101; 11010111; 00110110 записаны в прямом восьмибитном коде.

а) Найдите восьмибитные коды всевозможных попарных сумм этих чисел, используя для отрицательных чисел дополнительный код.

б) Найдите восьмибитные коды всех попарных разностей этих чисел.

§ 24 Представление вещественных чисел в памяти компьютера

Вы давно знаете, что, кроме целых чисел, человек активно пользуется дробями. И любое вещественное число может быть записано конечной или бесконечной десятичной дробью. Впрочем, такое представление чисел бывает полезным, как правило, только в теоретических построениях. А на практике...

Попробуйте ответить на следующие вопросы:

1. Каково население Земли?
2. Каково население вашего города (или города, ближайшего к населенному пункту вашего проживания)?
3. Сколько человек учится в вашем классе?

Но прежде чем отвечать, давайте подумаем, какими должны быть ответы. Совершенно ясно, что никто не в состоянии дать от-

вет на первый вопрос с точностью до одного человека. Как правило, ответ типа 6,5 миллиарда будет признан подходящим.

Ответы типа 1,3 миллиона или 56 тысяч на второй вопрос вполне будут удовлетворительными. А вот в третьем случае нужен и может быть дан точный ответ, например 24 человека.

Несмотря на совершенно разную точность — до ста миллионов в первом случае, до ста тысяч во втором, до единиц в третьем, в них есть нечто общее. А именно: они приведены с двумя значащими цифрами.

На практике в большинстве случаев приходится иметь дело именно с приближенными значениями величин. Приближенные значения возникают при измерениях, при подсчете больших величин, да и во многих других случаях. Поэтому исследователь или инженер, решая ту или иную задачу, должен изначально оценить, сколько значащих цифр следует иметь в ходе проводимых им вычислений и сколько оставить в результате.

Допустим, мы решили, что нам достаточно трех значащих цифр. Тогда в числах 37 200 000; −370; 3,72; 0,000372 нам требуется знать лишь эти три цифры и то, начиная с какого десятичного разряда они записаны. Именно такую информацию и надо сообщить компьютеру. Для этого представим числа единообразно: пишем 0 (перед ним знак «−», если число отрицательное), затем запятую, сразу после запятой — значащие цифры и получившуюся десятичную дробь умножаем на подходящую степень числа 10. Вот как преобразуются четыре указанных выше числа:

$37\,200\,000 = 0{,}372 \cdot 10^{8}$; $\quad -372 = -0{,}372 \cdot 10^{3}$;

$3{,}72 = 0{,}372 \cdot 10^{1}$; $\quad 0{,}000372 = 0{,}372 \cdot 10^{-3}$.

Абсолютную величину любого числа можно представить как произведение числа, заключенного между 0,1 и 1, и степени числа 10 с целым показателем. Дробная часть первого множителя в таком представлении называется **мантиссой** числа, а показатель степени числа 10 — **порядком** числа. Само представление числа в виде такого произведения называется **нормализованной записью** числа. По-другому такое представление чисел называют **записью чисел с плавающей запятой**. Максимально допустимое количество разрядов в мантиссе числа определяет точность, с которой данное число может быть представлено.

С нормализованной записью чисел вы встречаетесь, когда пользуетесь любым калькулятором. Правда, она несколько отличается от той, которую мы описали выше: мантисса числа, выводимого на табло калькулятора в нормализованном виде, заключена от 1 до 10. А порядок отделяется от мантиссы символом «e». Например, число 345,687249 в нормализованном виде выглядит на табло калькулятора так: 3,45687249e+2. А число 0,000345687249 будет выглядеть так: 3,45687249e−4.

Использование нормализованной записи чисел существенно расширяет диапазон чисел, которые можно обрабатывать. Стандартный инженерный калькулятор, работающий под Windows, в формате с фиксированной запятой позволяет записывать не более чем тринадцатиразрядные числа. Это значит, что максимальное число в этом формате 9 999 999 999 999. А минимальное положительное число — это 0,000000000001. Но даже если для записи порядка отвести всего лишь два разряда, то максимальное число, воспринимаемое калькулятором, будет 9,999999999999e+99. Это 100-значное число, которое в 10^{87} раз больше, чем число 9 999 999 999 999. Точно так же число 1,0e–99 в 10^{87} раз меньше числа 0,000000000001.

В памяти компьютера числа представлены в двоичной системе счисления. Для двоичной системы нормализованный вид числа — это представление его в виде $\pm m \cdot 2^p$, где $0{,}1_2 \le m < 1$, а p — целое число. Например:

$$0{,}1_{10} = 0{,}0(0011)_2 = 0{,}11(0011)_2 \cdot 2^{-3}.$$

Число с плавающей запятой может занимать в памяти компьютера разное количество байтов. Обычно для записи числа отводится 4 байта, а числа двойной точности занимают 8 байтов. Впрочем, встречаются и другие варианты, например когда под запись числа отводится 10 байтов.

В большинстве практических случаев того, что называют обычной точностью представления действительных чисел, оказывается вполне достаточно. Поэтому и мы дальше рассмотрим только этот случай. В нем под знак числа и порядок отводится 1 байт, и это первые 8 разрядов. Остальные 24 разряда отводятся под мантиссу. Это, например, означает, что мантисса компьютерного представления числа $0{,}1_{10}$ (с учетом округления) будет такова:

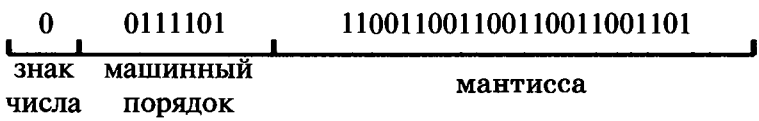
$$11001100110011001100 1101.$$

Знак числа кодируется обычным образом: «+» кодируется символом 0, «–» — символом 1. Что касается порядка числа, то там записывается так называемый **машинный порядок.** Под него отведено семь разрядов, и он равен целому неотрицательному числу, для которого данный код является прямым двоичным кодом. Например, коду 0101011 соответствует машинный порядок 43. Машинный порядок связан с порядком числа следующим образом:

$$\text{порядок числа} = \text{машинный порядок} - 2^6.$$

Тем самым цепочка 0101011 является кодом порядка, равного –21. А последовательность 0000000 кодирует порядок –64. Нетрудно видеть, что нулевой порядок кодируется как 1000000. Наибольший положительный порядок имеет код 1111111 и равен 63.

Вот как выглядит полный компьютерный код числа $0,1_{10}$:



Заметим еще раз, что представление чисел с плавающей запятой в памяти компьютера может осуществляться весьма по-разному: в зависимости от языка программирования, особенностей архитектуры и т. д. Например, в некоторых языках программирования под мантиссу отводится всего лишь 1 байт и знак числа записывается при мантиссе, а не выносится в крайний левый разряд. Мы здесь продемонстрировали только основные принципы представления таких чисел.

Вопросы и задания

1. Какую запись числа называют нормализованной? Что такое мантисса и порядок числа?

2. а) Запишите в нормализованном десятичном виде числа 123,456; 1,01001; –987654321; 0,002000300004.
   б) Запишите в виде числа с фиксированной запятой следующие числа, записанные в нормализованном десятичном виде: $0,35421 \cdot 10^{3}$; $0,12457 \cdot 10^{-4}$; $-0,327 \cdot 10^{6}$.

3. Каким будет первый символ кода машинного порядка числа, если порядок числа, представленного в двоичном нормализованном виде, неотрицателен?

4. Запишите следующие числа, представленные в десятичной системе счисления, в четырехбайтном машинном коде для чисел с плавающей запятой: а) 375; б) –2,75; в) 0,15; г) –0,0125.

5. В объяснительном тексте приведена четырехбайтная запись числа $0,1_{10}$.
   а) Какому десятичному числу на самом деле соответствует эта запись?
   б) Каковы абсолютная и относительная погрешности представления числа $0,1_{10}$?
   в) Объясните, почему последней цифрой мантиссы оказалась цифра 1.

6. Укажите, какие из нижеприведенных действительных чисел могут быть представлены четырехбайтным кодом числа с плавающей запятой, описанным в объяснительном тексте:
   а) $0,19 \cdot 10^{18}$; б) $-0,9 \cdot 10^{19}$; в) $0,95 \cdot 10^{19}$;
   г) $-0,37 \cdot 10^{-19}$; д) $0,37 \cdot 10^{-20}$; е) $-0,25 \cdot 10^{-19}$.

§ 25 Особенности компьютерной арифметики

Ограниченность числа разрядов в каждой ячейке памяти компьютера — причина того, что операции над числами выполняются компьютером не совсем так, как вы привыкли выполнять их еще в начальной школе. Рассмотрим сначала, как выполняются операции над целыми числами. Пусть, к примеру, вычисляется сумма двух положительных чисел 1010101_2 и 111101_2. Записываем их коды и получаем 01010101 + 00111101 = 10010010. Но 1 в старшем разряде показывает, что результат оказался отрицательным числом!

Ошибка возникла из-за того, что разрядная сетка ячейки содержит всего семь мест. Впрочем, если ячейка имеет и больше разрядов, все равно такая ошибка будет возникать каждый раз при нахождении суммы достаточно больших чисел. Конечно, такая ошибка может возникнуть не только при сложении, но и при умножении, причем при умножении даже чаще, чем при сложении.

Обнаруженное нами явление называется **эффектом переполнения**. И о нем надо помнить, когда приходится иметь дело с достаточно большими целыми числами.

А как на это реагирует компьютер? **Переполнение при выполнении операций с числами, записанными с фиксированной запятой, не вызывает прерывания работы процессора.** В зависимости от используемого языка проверка переполнения может отсутствовать (так, например, происходит в большинстве версий языка Pascal). Компьютер может сообщать пользователю о переполнении и ждать его реакции, а может просто переходить к записи числа с плавающей запятой (так бывает в различных версиях языка Basic).

Рассмотрим сложение чисел с плавающей запятой. Если у чисел в нормализованном виде порядки одинаковы, то достаточно сложить мантиссы этих чисел, а затем нормализовать полученный результат, если эта сумма окажется больше 1 или меньше 0,1. Например:

$$0{,}101 \cdot 2^{-3} + 0{,}1101 \cdot 2^{-3} = 1{,}0111 \cdot 2^{-3} = 0{,}10111 \cdot 2^{-2};$$
$$0{,}101 \cdot 2^{-3} + (-0{,}1101 \cdot 2^{-3}) = -0{,}0011 \cdot 2^{-3} = -0{,}11 \cdot 2^{-5}.$$

Хотя примеры приведены в двоичной системе счисления, ясно, что высказанное правило действует в любой позиционной системе.

Если же порядки нормализованных чисел различны, то при сложении чисел с плавающей запятой предварительно выполняется **операция выравнивания порядков** слагаемых. Заметим, что величина числа не меняется при сдвиге мантиссы на один разряд вправо с одновременным увеличением порядка на 1. Поэтому в том слагаемом, у которого порядок меньше, производится сдвиг ман-

тиссы вправо на разность между порядками, после чего мантиссы складываются, и результат снова нормализуется. Например:

$$0{,}101 \cdot 2^{-3} + 0{,}1101 \cdot 2^{-4} = 0{,}101 \cdot 2^{-3} + 0{,}01101 \cdot 2^{-3} = \\ = 1{,}00001 \cdot 2^{-3} = 0{,}100001 \cdot 2^{-2};$$

$$0{,}101 \cdot 2^{-3} + (-0{,}1101 \cdot 2^{-4}) = 0{,}101 \cdot 2^{-3} + (-0{,}01101 \cdot 2^{-3}) = \\ = 0{,}00111 \cdot 2^{-3} = 0{,}111 \cdot 2^{-5}.$$

Можно сформулировать следующее правило сложения чисел с плавающей запятой:

> Чтобы сложить два числа в нормализованном виде, их порядки выравнивают, мантиссы складывают, после чего результат, если необходимо, нормализуют.

Намного проще выполняется умножение и деление чисел в нормализованном виде.

> Чтобы умножить два числа в нормализованном виде, их порядки складывают, а мантиссы перемножают, после чего результат, если необходимо, нормализуют.

Например:

$$0{,}101 \cdot 2^{-3} \times 0{,}1011 \cdot 2^{4} = (0{,}101 \times 0{,}1011) \cdot 2^{-3+4} = \\ = 0{,}0110111 \cdot 2^{1} = 0{,}110111 \cdot 2^{0}.$$

> Чтобы в нормализованном виде разделить одно число на другое, из порядка делимого вычитают порядок делителя, а мантиссу делимого делят на мантиссу делителя, после чего полученный результат нормализуют.

Например:

$$0{,}1011 \cdot 2^{-3} : 0{,}101 \cdot 2^{4} = (0{,}1011 : 0{,}101) \cdot 2^{-3-4} = \\ = 1{,}0001100110011\ldots \cdot 2^{-7} = 0{,}10001100110011\ldots \cdot 2^{-6}.$$

Последний пример показывает, что на практике даже при обычных вычислениях мы будем вынуждены довольствоваться лишь приближенным значением дроби. Пусть мы складываем числа $a = 0{,}1 \cdot 2^{13}$ и $b = 0{,}1 \cdot 2^{-12}$. После выравнивания порядков получаем:

$$\begin{array}{r} 0{,}1 \\ +\ 0{,}00000000000000000000000001 \\ \hline 0{,}10000000000000000000000001 \end{array}$$

Но в разрядную сетку мантиссы помещается лишь 24 цифры, а не 26. Поэтому два последних разряда будут утеряны, и результат совпадет с первым слагаемым. Получается, что в компьютерной арифметике вполне может оказаться $a + b = a$, хотя $b \neq 0$. И это далеко не единственная особенность компьютерной арифметики. В задании 6 приведен пример, показывающий, что **может не выполняться сочетательный закон для сложения.**

При умножении и делении нормализованных чисел одним из возможных эффектов является переполнение разрядной сетки порядка числа. Вот пример:

$$0{,}1001 \cdot 2^{32} \times 0{,}11 \cdot 2^{33} = 0{,}011011 \cdot 2^{65} = 0{,}11011 \cdot 2^{64}.$$

Но наивысший возможный порядок — это 63. Следовательно, данный результат непредставим в компьютерной арифметике. Обычно операционная система в таком случае диагностирует переполнение. Переполнение для нормализованных чисел может возникнуть и при сложении; в задании 8 вам предлагается построить соответствующий пример.

Кроме рассмотренных случаев, когда при вычислении на компьютере получается неверный результат или результат вообще не получается, надо иметь в виду эффекты, связанные с округлением чисел. Эти эффекты также обусловлены ограниченностью разрядной сетки. Одним из них является потеря значащих цифр. Возьмем, к примеру, числа 1/1000 и 1/1004. В двоичной системе счисления в нормализованном виде они с учетом округления равны соответственно $0{,}10000011000100100110111 \cdot 2^{-9}$ и $0{,}10000010100001100101100000 \cdot 2^{-9}$. После вычитания получится $0{,}1000010110111111 \cdot 2^{-17}$. Точное значение разности равно 4/1004000. При записи в нормализованном виде с 24-разрядной мантиссой получаем $0{,}10000101101011101110110 0 \cdot 2^{-17}$. Как видите, только первые 11 цифр после запятой оказались точными.

Итак, компьютерная арифметика может давать эффекты:

1. Ошибки округления, которые возникают при записи чисел с округлением и накапливаются при выполнении операций.
2. Переполнение разрядной сетки порядка чисел, что приводит к получению невоспроизводимого в компьютере числа.
3. Потеря значащих цифр при вычитании близких чисел или при переполнении разрядной сетки мантиссы.
4. Игнорирование компонента операции при большой разнице в порядках.

В качестве практической рекомендации отметим, что не следует сравнивать вещественные числа на точное равенство; вместо этого надо сравнивать абсолютную величину их разности с подходящим маленьким положительным числом, соответствующим погрешности представления.

Вопросы и задания

1. Какова основная причина эффектов компьютерной арифметики?

2. а) Выполните сложение чисел 100_{10} и 50_{10} в однобайтовом коде представления целых чисел с фиксированной запятой. Кодом какого числа (в десятичной системе счисления) будет полученный результат? (Совет. Не забудьте, что отрицательные числа представляются в дополнительном коде.)
   б) Выполните сложение чисел -80_{10} и -64_{10} в однобайтовом коде представления целых чисел с фиксированной запятой. Кодом какого числа (в десятичной системе счисления) будет полученный результат?

3.* Легко понять, что при компьютерном сложении целых чисел выполняется переместительный закон: $a + b = b + a$. Будет ли при компьютерном сложении целых чисел обязательно выполняться сочетательный закон $(a + b) + c = a + (b + c)$?

4. Выполните сложение двоичных чисел, представленных в нормализованном виде, результат нормализуйте:
   а) $0{,}101_2 \cdot 2^3 + 0{,}1_2 \cdot 2^3$; б) $0{,}1011_2 \cdot 2^2 + 0{,}1_2 \cdot 2^{-1}$;
   в) $-0{,}1011_2 \cdot 2^{-1} + 0{,}1_2 \cdot 2^{-2}$.

5. Выполните умножение двоичных чисел, представленных в нормализованном виде, результат нормализуйте:
   а) $0{,}101_2 \cdot 2^3 \times 0{,}1_2 \cdot 2^3$; б) $0{,}1011_2 \cdot 2^2 \times 0{,}1_2 \cdot 2^{-1}$;
   в) $-0{,}1011_2 \cdot 2^{-1} \times 0{,}1_2 \cdot 2^{-2}$.

6.* Объясните, почему при умножении чисел с плавающей запятой может нарушаться сочетательный закон. Приведите подтверждающие примеры.

7. Приведите пример двух нормализованных двоичных чисел, при сложении которых может произойти переполнение.

8. Петя и Коля для решения одной и той же задачи составили алгоритмы:

```
Алгоритм Пети
цел: N, K; вещ: S;
{ Запросить N;
  S := 0;
  Делать от K := 1 до N
  { S := S + 1/K; }
  Сообщить S;
}
```

```
Алгоритм Коли
цел: N, K; вещ: S;
{ Запросить N;
  S := 0;
  Делать от K := N до 1 с шагом -1
  { S := S + 1/K; }
  Сообщить S;
}
```

   а) Верно ли, что оба алгоритма решают одну и ту же задачу?
   б)* Верно ли, что при $N = 1\,000\,000\,000$ программы, реализующие данные алгоритмы, дадут одинаковый результат, если используется четырехбайтное представление нормализованных чисел?

3 ГЛАВА Основные информационные объекты. Их создание и компьютерная обработка

Информация, чтобы с ней можно было как-то работать, должна быть зафиксирована подходящим способом. Напомним, что зафиксированную каким-либо способом информацию мы в учебнике 10 класса назвали **информационным объектом**. Таким объектом является и текст книги, и картина художника, и фильм, снятый на фотопленку. Но в центре нашего внимания будут информационные объекты, которые можно обрабатывать с помощью компьютера. Такие объекты обычно называют **оцифрованными**. В предыдущей главе, говоря о кодировании информации, мы обсуждали различные способы оцифровывания информации различного вида. В этой главе речь пойдет о компьютерной обработке оцифрованных информационных объектов.

§ 26 Создание и форматирование текста

В «Словаре русского языка» С. И. Ожегова сказано: «Текст — всякая записанная речь». И совершенно неважно, будет ли она записана буквами, или иероглифами, или стенографическими значками, или еще как-нибудь. С позиций информатики такая формулировка означает, что текст — это просто последовательность знаков некоторого алфавита.

Что же следует понимать под термином «компьютерная обработка текста»? Под обработкой текста обычно понимают такое его преобразование, которое не меняет по существу его информационное содержание. Такое преобразование может быть направлено на то, чтобы облегчить смысловое восприятие информации, заключенной в данном тексте. Вот только один пример: «Боря ударил палкой по табуретке и сломал ее». Что сломал Боря: палку или табуретку?

Компьютер, как вы помните, — формальный исполнитель, и поэтому ему доступна только формальная обработка текста. Он не в состоянии решить вопрос, что сломал Боря, — понимание

смысла информации за пределами его возможностей, а изменение смысла вне его компетенции.

В чем же состоит формальное преобразование текста? В том, что исправляются орфографические, синтаксические и стилистические ошибки, удаляются ненужные повторы, вводятся уточняющие слова и т. п. Текст структурируется, т. е. разбивается на разделы — главы, параграфы, пункты и абзацы. Наконец, в текст вставляются иллюстрации — чертежи, рисунки, диаграммы, графики и т. п. Структурные элементы текста тем или иным образом располагаются на странице — центрируются заголовки, определенным образом располагаются эпиграфы к главам или параграфам, выравниваются (или, наоборот, не выравниваются) края текста, выделяются каким-либо образом основные положения и т. д.

Подобные преобразования текста обычно называют его **редактированием**, а компьютерное программное обеспечение, используемое для обработки текстов в электронном виде, называют в зависимости от предоставляемых возможностей текстовыми редакторами, текстовыми процессорами или программами верстки.

Каждый текстовый редактор позволяет вставлять новые слова, удалять отдельные буквы, переставлять целые абзацы, автоматически заменять во всем тексте одно слово другим и т. д. Многие текстовые редакторы умеют автоматически разбивать текст на страницы и нумеровать их. Они могут следить за размером полей и выравнивать текст. Им можно даже поручить обнаружение и исправление орфографических ошибок!

В базовом курсе информатики вы наверняка знакомились с текстовым редактором Блокнот и процессором Microsoft Word. Возможности редактора Блокнот во много раз меньше, чем возможности процессора Word. Поэтому дальше мы будем говорить преимущественно о возможностях Word. Этот программный продукт неоднократно модернизировался фирмой Microsoft — разработчиком этого текстового редактора. Между версиями разных лет имеются отличия. Мы будем рассказывать об основных функциях Microsoft Word — они у всех версий общие, а о конкретной их реализации вы можете узнать из инструкции пользователю данным продуктом или с помощью встроенной справки и всплывающих подсказок.

Информационные объекты, создаваемые и обрабатываемые текстовым процессором Word, называются документами. Создание документа начинается с выбора **шаблона**. Дело в том, что многие документы должны иметь стандартный вид, которым строго определено, что и где размещается в создаваемом тексте, например кому адресован документ, от кого, дата и другие реквизиты. По умолчанию загружается заготовка так называемого обычного документа, в котором реквизиты отсутствуют. Шаблон обычного документа хранится в файле normal.dot. Расширение dot показывает, что мы имеем дело именно с шаблоном документа, созданного посредством Microsoft Word.

Вы можете создать и собственный шаблон документа. Для этого полезно воспользоваться услугами *Мастера шаблонов*. Вы познакомитесь с ним, выполняя лабораторную работу № 7.

Текст состоит из символов. Символы складываются в слова, слова — в предложения, предложения — в абзацы. Сам текст распределяется по страницам. Страница, абзац и символ — вот те объекты, о задании параметров которых надо прежде всего задуматься, создавая текст.

Расположение текста на странице определяется в первую очередь размером страницы и полями. По умолчанию страница имеет размер 210 × 297 мм. Это так называемый формат А4. Половина такой страницы называется форматом А5. Если же две страницы формата А4 соединить по длинной стороне, получится лист формата А3. Нетрудно догадаться, как получается лист формата А2 или А6. Кроме того, применяются листы других размеров, их параметры и маркировку вы посмотрите, выполняя лабораторную работу № 7.

Сама страница может быть в книжной (вертикальной) или в альбомной (горизонтальной) ориентации. Размеры страницы при этом, конечно, не меняются.

Поля страницы определяют расположение текста относительно краев листа. Это не означает, что вы не сможете ничего разместить на полях. На боковых полях текст можно разместить, если раздвинуть ограничители текстового поля, расположенные на горизонтальной линейке (они имеют форму треугольничков). Текстовое поле можно расширить и с помощью меню *Формат*, выбрав в нем пункт *Абзац*. На вкладке *Отступы и интервалы* достаточно указать левый (или правый) отступ отрицательным, если надо, чтобы текст «вылез» на левое (или правое) поле.

Текст, выносимый на верхнее и нижнее поля, называют **колонтитулом**. Его создают путем обращения к пункту меню *Вид*. Обычно в колонтитуле указывают название текущего раздела, главы или параграфа. Этот текст будет воспроизводиться на всех страницах в пределах одного раздела. Предусмотрено, что колонтитулы на четных и нечетных страницах могут быть различными. Номера страниц также выставляются в колонтитул. Нумерацию страниц можно организовать независимо, используя меню *Вставка*.

Текст на странице можно расположить в несколько колонок. Назначить количество колонок, их ширину, расстояние между ними можно, выбрав в меню *Формат* пункт *Колонки*.

Теперь поговорим о следующем объекте — абзаце. Об одном параметре абзаца — величине отступов от левого и правого полей — мы уже сказали выше. Наверняка вы знаете и о таком параметре, как выравнивание текста. Оно отражает вид границы текста в абзаце. **Выравнивание по левому краю** означает, что первые символы всех строк абзаца (кроме, быть может, первой) находятся на одной вертикальной прямой. Аналогично **выравнивание по правому**

краю означает, что последние символы всех строк абзаца, кроме последней, располагаются вдоль одной вертикальной прямой. **Выравнивание по ширине** — это одновременное выравнивание по левому и правому краям. Наконец, **выравнивание по центру** означает, что каждая строка размещается симметрично относительно вертикальной прямой, делящей страницу пополам. Выравнивание по центру применяется обычно для заголовков, формул и рисунков, вынесенных в отдельную строку элементов текста, к которым требуется привлечь особое внимание.

Для первой строки абзаца может быть применено отдельное форматирование. Эта строка может быть сделана **красной**, т. е. набираться с отступом вправо, или **висячей**, т. е. с отступом влево относительно расположения остальных строк абзаца.

Строки внутри абзаца могут располагаться на разном расстоянии друг от друга. Это расстояние называется **межстрочным интервалом**. Величина интервала измеряется в **пунктах** (здесь пунктом называется не раздел текста, а специальная единица длины, применяемая в издательском деле: 1 пункт = 0,376 мм). Интервал может быть одинарным (т. е. равным по величине размеру используемых заглавных букв), полуторным (т. е. в 1,5 раза больше одинарного), двойным (т. е. в 2 раза больше одинарного) и вообще любым. В последнем случае просто устанавливают величину межстрочного интервала прямо в пунктах на вкладке *Абзац* в меню *Формат*.

Интервал между абзацами также можно менять. Любое натуральное значение этого параметра показывает, на сколько пунктов интервал между абзацами больше межстрочного интервала внутри абзаца (предыдущего или последующего).

Расположение текста на странице называют его **форматом**. Поэтому задание (или изменение) тех или иных рассмотренных выше параметров текста называют **форматированием**.

Иногда требуется пронумеровать некоторую совокупность абзацев. Эту формальную процедуру тоже можно поручить компьютеру. Для этого, выделив подлежащие нумерации абзацы, включаем режим Нумерованный список (соответствующая кнопка обычно располагается на панели инструментов; можно также воспользоваться вкладкой *Список* в меню *Формат*). Если из выделенной совокупности какой-то абзац надо исключить из нумерации, то выделите его или установите курсор внутри этого абзаца и еще раз щелкните по кнопке *Нумерованный список*. Этот абзац окажется ненумерованным, зато последующие получат номер, на 1 меньший.

Вместо номера в начале абзаца можно поставить тот или иной символ, называемый **маркером**. В качестве маркера могут выступать разные символы, например: –, •, ♦, ⇒.

Напомним основные параметры, которыми характеризуются символы текста. Их три: гарнитура, кегль и начертание. Гарнитурой называется вид шрифта. Наиболее часто используемые —

это гарнитура Times и гарнитура Arial. Они представляют два разных типа шрифтов: шрифты с засечками и рубленые шрифты. Например:

Этот текст набран шрифтом с засечками.

А этот текст набран рубленым шрифтом.

Шрифты с засечками как бы визуально объединяют слово в единое целое, и это увеличивает скорость чтения на 10—15%. Рубленые шрифты, как правило, используются в заголовках и подписях к рисункам.

Кроме названий, определяющих вид букв, шрифты имеют размер, называемый **кеглем**. Стандартный шрифт, к которому стараются приблизиться, печатая деловые документы, очень похож на шрифт гарнитуры Times размером в 14 пунктов. Вообще, хорошо читаются шрифты с кеглем от 9 до 14 пунктов.

Обычно каждый шрифт имеет по крайней мере три модификации начертания: **полужирный**, *наклонный* (часто называемый *курсивом*) и подчеркнутый. Могут также использоваться комбинации модификаций, например одновременно полужирный, наклонный и подчеркнутый, — ***вот так***. Подчеркивание можно выполнять различными типами линий, например двойной чертой или пунктиром. Можно зачеркнуть любое ~~слово~~ или любой ~~фрагмент~~ текста. Любую комбинацию символов можно представить в виде $^{\text{верхнего}}$ или $_{\text{нижнего}}$ индекса; можно изменить расстояние между символами, например записать слово в р а з р я д к у. Интервал между символами здесь также измеряется в пунктах. Для записи текста в разрядку крайне нежелательно использовать символ «пробел» между буквами, поскольку, во-первых, при расположении такого слова на правом краю строки часть этого слова может быть автоматически перенесена на новую строку, а во-вторых, большинство текстовых редакторов, выполняя выравнивание по ширине, добивается этого растягиванием пробелов между словами. Поскольку в этом случае каждый символ будет восприниматься как отдельное слово, то пробелы между ними также будут растягиваться. Все изменения шрифтовых параметров производятся с помощью вкладки *Шрифт* в меню *Формат*.

Мы привыкли к белому фону, на котором располагается текст, как правило черного цвета. Но при желании текст можно поместить на цветной фон или сделать цветными сами символы текста.

Изменяя размер и форму шрифта, можно добиваться самых различных эффектов, но не надо впадать в крайности: большое количество шрифтов на одной странице ухудшает восприятие текста и вряд ли свидетельствует о хорошем вкусе. (Можете это проверить практически на любой газете, основу которой составляют рекламные объявления.)

Вопросы и задания

1. Что такое текстовый редактор?
2. Что такое форматирование текста?
3. Какие типы выравнивания текста применяются? Что означает каждый из них?
4. Что такое колонтитул и для чего он используется?
5. Какая строка называется красной, а какая — висячей?
6. Что такое кегль? В каких единицах он обычно измеряется? Какие значения кегля предпочтительны для обычного текста?
7. В заданном тексте требуется заменить слово «роман» на слово «детектив». Разумеется, это слово необязательно стоит в именительном падеже. Каким инструментом текстового редактора и как можно воспользоваться, чтобы выполнить эту работу? (С о в е т. Обдумайте ситуацию, когда в тексте может встретиться слово «романтический».)
8. а) Из приведенного ниже списка действий по преобразованию текста укажите те, которые, по вашему мнению, можно производить с помощью текстового редактора:
   - автоматическая вставка данного символа между двумя заданными символами во всем тексте;
   - автоматическая замена одного слова (и только его!) другим во всем тексте;
   - автоматическое заключение в кавычки заданного слова во всем тексте;
   - автоматическая замена числа, набранного цифрами, на соответствующее числительное, набранное буквами;
   - подсчет, сколько раз в тексте встречается заданное слово;
   - автоматическая ликвидация всех пробелов между словами;
   - автоматическое удвоение всех пробелов;
   - автоматическая вставка пустой строки после каждого абзаца во всем тексте;
   - автоматическая ликвидация пустых строк между абзацами во всем тексте.

   б) Из перечисленных в пункте *а* действий, которые, по вашему мнению, нельзя поручить выполнить текстовому редактору автоматически сразу во всем тексте, укажите те, которые можно выполнить пошагово, т. е. каждый раз давая компьютеру разрешение на выполнение данного действия или отказывая ему в этом.
9. Подсчитайте, какие размеры имеют листы формата A3; A2; A1; A6.

⑩* Человеческий глаз очень чувствителен к соотношениям размеров. Для человека книга, напечатанная на листах, например, формата А5, будет восприниматься как уменьшенная копия книги, напечатанной на листах формата А4. Проверьте, будет ли лист формата А4, разрезанный пополам, подобен листу формата А5 (в некотором приближении). Если да, то найдите коэффициент подобия.

⑪ Выберите какое-нибудь небольшое (от 5 до 10 страниц) литературное произведение, не содержащее большого числа диалогов. Можете взять понравившийся вам рассказ или что-нибудь из того, что вы изучаете на уроках литературы. При выполнении лабораторной работы № 7 вам будет предложено составить подробный план выбранного произведения. Продумайте этот план.

§ 27 Вставка объектов в текст документа

Текстовое сообщение — важная форма сохранения и передачи информации. Но текст — это застывшие звуки. Не случайно дети, когда учатся читать, нередко шевелят губами, как бы озвучивая то, что они читают. Но немалую роль играют другие формы представления информации — рисунки, таблицы, схемы, диаграммы и т. д. Документ, в котором, кроме текста, использованы другие информационные объекты, лучше воспринимается и нередко оказывается более продуктивным для информационной деятельности человека. Поэтому практически все современные текстовые редакторы предусматривают возможность внедрения указанных информационных объектов в текстовый документ.

Составной частью текстового процессора Word является достаточно простой, можно сказать, примитивный, табличный редактор. Он позволяет в документе создать таблицу. При вводе текста в клетку получившейся таблицы (такую клетку называют **ячейкой**) можно быть уверенным, что он будет строго ограничен левой и правой вертикальными линиями и не выйдет за их пределы. Можно изменять размеры ячеек и даже их расположение относительно друг друга, например сдвигать в горизонтальном направлении. При необходимости горизонтальные и вертикальные линии можно сделать видимыми, а некоторые ячейки выделить фоном. Все это позволяет использовать таблицы как весьма эффективное средство редактирования текста.

Если в ячейках таблицы находятся числа, то редактор может выполнить с ними различные арифметические действия. Например, найти сумму по столбцу или строке, вычислить среднее или максимальное значение и т. п.

Созданную вами таблицу вы можете поместить в рамку. Тогда она превратится в объект, который вы можете передвигать в текс-

Рис. 3.1 Наложение рисунка на текст

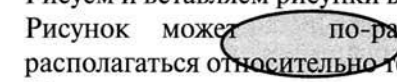


Рис. 3.2 Наложение текста на рисунок

те как единое целое. В частности, можно организовать обтекание таблицы текстом.

Если же таблица была создана в другом приложении Windows, например Microsoft Excel, то ее можно сразу вставить как объект, пользуясь меню *Вставка / Объект*. Впрочем, для таблиц, созданных в Excel, на панели инструментов имеется кнопка *Вставка таблицы Microsoft Excel*.

Вставка рисунка тоже может производиться разными способами. Во-первых, имеется встроенный векторный графический редактор, позволяющий рисовать прямо в документе. При этом может оказаться, что нарисованная фигура закроет часть текста (рис. 3.1). Если это не входит в ваши планы, легко поменять взаимное расположение фигуры и текста (рис. 3.2). Если рисунок поместить в рамку, то текст будет обтекать рисунок (рис. 3.3).

Во-вторых, рисунок может быть создан с помощью какого-либо другого приложения или взят готовым из библиотеки рисунков. В этом случае рисунок обычно вставляется как объект с эффектом обтекания. Тип рисунка — векторный он или растровый — роли не играет.

Текстовый процессор Word позволяет записывать математические формулы. Если математическое выражение не очень сложное, например многочлен от одной или нескольких переменных, то для его записи хватает обычных средств, имеющихся в Word. Но иногда в тексте требуется представить формулу, в которой фигурируют дроби, корни, переменные с верхними и нижними индексами одновременно, векторы и другие математические символы, например:

$$S = v_0 t + \frac{at^2}{2}; \quad A = \frac{mv_{\text{нач}}^2}{2} - \frac{mv_{\text{кон}}^2}{2}$$

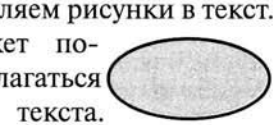


Рис. 3.3 Обтекание рисунка текстом

В этом случае приходится пользоваться специальными средствами. Одним из них является Microsoft Equation. Выбрав в меню *Вставка* этот пункт, вы в тексте документа увидите выделенную рамкой область для набора формулы и панель ин-

струментов, с помощью которых конструируется формула. После того как формула набрана, для выхода из режима создания формулы достаточно щелкнуть левой кнопкой мышки за пределами поля формулы. Вернуться в поле формулы для ее редактирования можно, установив курсор мыши на формуле и сделав двойной щелчок левой клавишей.

В текстовый документ можно встраивать и другие объекты, созданные с помощью тех или иных приложений. Это могут быть анимированные изображения, звук, видеофрагменты и т. д. Для их встраивания применяется единый механизм OLE — Object Linking and Embedding.

Вопросы и задания

1. Какие возможности в создании текстового документа предоставляет вставка объектов?
2. Какие объекты могут быть созданы и внедрены в текстовый документ собственными средствами текстового процессора Word?
3. Каков универсальный механизм внедрения объектов, созданных в других приложениях?

§ 28 Гипертекст

При всем том, что было рассказано о тексте в § 26 и 27, одно его свойство остается неизменным — линейный характер. Это означает, что для перехода от одного фрагмента текста к другому надо пролистать весь промежуточный текст. Еще сложнее такой поиск осуществить, если связанные по смыслу фрагменты текста находятся в разных документах. Чтобы автоматизировать переход с одного фрагмента на другой, их надо связать между собой. Осуществлять такую связь призваны так называемые **гиперссылки**. **Гипертекстом** называется текст, в котором организованы гиперссылки на фрагменты или объекты того же текста либо другого документа.

Можно представлять себе, что гипертекстовая страница состоит из видимого и «подвального» этажей. На видимом располагаются самый обычный текст и самые обычные картинки. Отдельные слова и картинки особо выделены, и курсор мыши, оказавшись на них, принимает другую форму. Это означает, что под такими элементами страницы находятся ссылки на другие электронные документы или даже на целые программы, которые будут выполнены в случае нажатия на кнопку мыши при видоизмененном курсоре.

Что же располагается в «подвале», какие бывают ссылки?

- Самое простое — переход к другим страницам или к другому документу.
- Тоже не очень сложно — показ картинки или фильма, прослушивание звукового фрагмента.
- Немного сложнее (с точки зрения аппаратуры, но не с точки зрения того, кто просматривает гипертекстовый документ) — переадресация пользователя к другому гипертекстовому документу на другом сервере, расположенном, быть может, за тысячи километров.
- Возможна активизация специальных программ. Чаще всего это программы пересылки файлов.

Простейший гипертекстовый документ можно создать с помощью текстового процессора Word. Для этого нужно создать два объекта: гиперссылку и закладку — текст, на который мы будем переходить при использовании гиперссылки. Чтобы создать закладку, нужно выделить фрагмент текста, затем в меню *Вставка* выбрать пункт *Закладка*, в открывшемся окне ввести имя закладки. Для создания гиперссылки нужно выделить фрагмент текста (название гиперссылки), снова открыть меню *Вставка* и выбрать теперь пункт *Гиперссылка*. На диалоговой панели указать, с чем вы намерены связать этот текст (с местом в текущем документе или каком-то другом документе), и выбрать имя закладки. При выполнении лабораторной работы вы научитесь создавать разные гиперссылки в текстовом процессоре Word.

Microsoft Word не предназначен для создания гипертекстовых страниц. Это инструмент для создания прежде всего обычных текстовых документов. А для создания гипертекстовых страниц, в которых использовались бы все возможности, перечисленные нами выше, нужен специальный гипертекстовый редактор. Таких редакторов существует довольно много, и у каждого свой стандарт форматов страниц. Понятно, что было необходимо выбрать один из них для обеспечения всем желающим беспрепятственного использования постоянно растущего множества разнообразных гипертекстовых объектов и особенно в глобальных компьютерных сетях. Таким стандартом стал HTML-стандарт. Сама аббревиатура HTML происходит от HyperText Markup Language (в переводе — язык разметки гипертекста).

В начале § 13 мы рассказали, что некоторые символы и их комбинации являются управляющими. Любой текстовый редактор, формируя текст абзац за абзацем, страница за страницей, одновременно создает описание форматирования текста и хранит его в файле вместе с самим текстом. Когда на экран компьютера вызывается текст документа, то программа-редактор, следуя этому описанию, воссоздает вид документа. Иногда это описание занимает не меньше памяти, чем сам текст. Microsoft Word в этом отношении нельзя назвать

экономным — вы, возможно, замечали, что добавление к уже имеющемуся тексту еще нескольких строк (особенно, если их вставлять в середину) может увеличить размеры файла на 5—10 килобайт. Это плата за то, что пользователь освобожден от забот по созданию описания форматирования текста. Человек не замечает этой работы, так же как богач-аристократ не видит работы своего садовника, а только наслаждается лужайками и цветниками в своем саду.

Но существует и иной тип редакторов. В них описание форматирования делает сам автор с помощью специального языка. Как правило, информационный объем таких документов всего лишь на несколько процентов превышает информационный объем собственно текста. После создания такой файл обрабатывается специальной программой, которая, следуя указаниям автора, создает документ с требуемым форматированием текста.

HTML — это язык для создания гипертекстовых документов. А программа просмотра текста, созданного с помощью HTML, называется **браузером**. В настоящее время используются браузеры Internet Explorer, Opera, Mozilla Firefox.

Вопросы и задания

1. В чем разница между обычной текстовой страницей и гипертекстовой страницей?
2. Что такое гипертекст?
3. Какой стандартный редактор используется для создания гипертекстовых страниц?
4. В чем основное отличие текстового процессора Word от HTML?
5. Как называется программа, предназначенная для просмотра файлов в HTML-формате?
6. Сравните возможности текстового процессора Word и текстового редактора Блокнот. Какие средства форматирования текста имеет Блокнот?

§ 29 Основы HTML

Вообще говоря, HTML-документ может быть создан при помощи любого текстового редактора, хотя нередко используются специализированные HTML-редакторы и так называемые конвертеры. Выбор редактора, который будет применяться для создания HTML-документов, зависит исключительно от личных пристрастий автора. Для создания своих HTML-страниц (вы займетесь этим, выполняя лабораторные работы) мы предлагаем вам использовать стандартное приложение Блокнот.

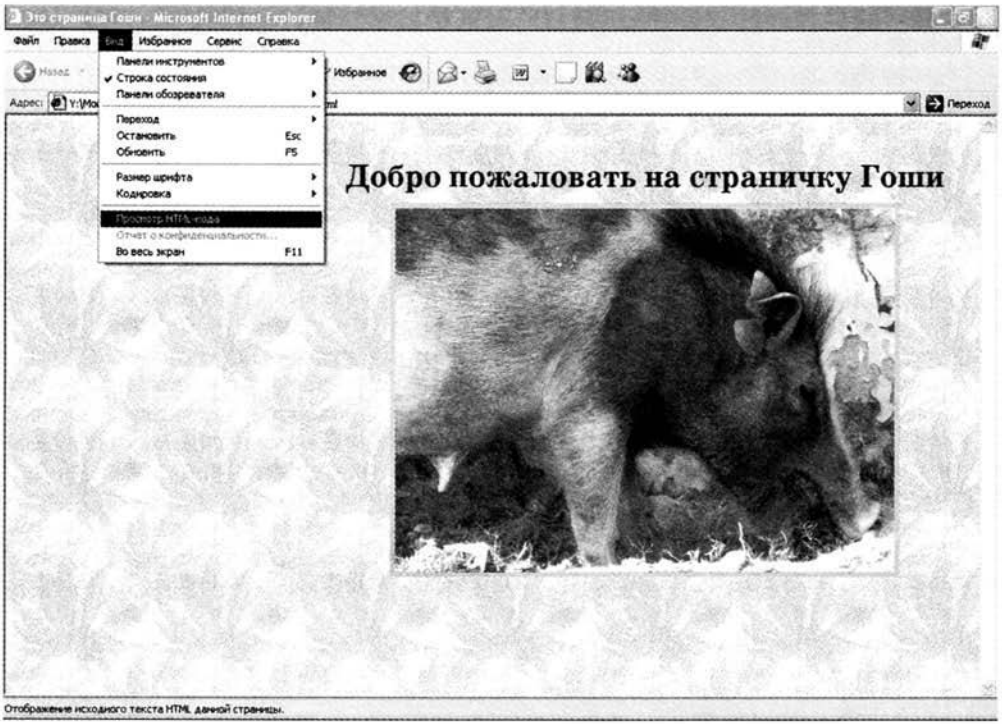


Рис. 3.4 Пример отображения HTML-страницы

А теперь давайте проанализируем самую простую страницу, описанную с помощью этого языка. В окне браузера она может выглядеть так, как показано на рисунке 3.4. В окне же текстового редактора, на который можно перейти, выбрав в меню *Вид / Просмотр HTML-кода*, вы увидите следующее:

```
<HTML>
<HEAD><TITLE> Это страница Гоши </TITLE></HEAD>
<BODY BGCOLOR="YELLOW" TEXT="BLACK" LINK="RED"
ALINK="BLUE">
<CENTER>
<FONT SIZE=5>
<B>

Добро пожаловать на страничку Гоши

</B>
<BR>
</FONT>
<IMG SRC="animal.jpg">
```

```
</CENTER>
</BODY>
</HTML>
```

Вполне понятные предложения на русском языке чередуются здесь с какими-то неясными словами, взятыми в своеобразные скобки из знаков < и >. На самом деле эти скобки сигнализируют программе просмотра, что внутри их стоят так называемые **теги** — управляющие словосочетания, указывающие браузеру на то, как надо оформлять ваш электронный документ.

Рассмотрим, к примеру, тег <CENTER>. Он означает, что все дальнейшие элементы оформления документа, а в нашем случае это текст и картинка, будут расположены строго по центру окна, выделенного программе просмотра. А отменяется это центрирование с помощью другого тега </CENTER>. Вообще косая палочка в теге означает отмену какого-либо элемента оформления. Тег <H3> заставляет программу просмотра весь дальнейший текст писать крупными буквами, так называемым заголовочным шрифтом третьего уровня. Надеемся, вы уже догадались, когда шрифт снова станет обычным. Кстати, раз уж речь пошла о заголовочных шрифтах, заметим, что всего их существует шесть уровней. Им соответствуют теги <H1>, <H2>, <H3>, <H4>, <H5> и <H6>. Первый уровень самый крупный.

Каждая пара тегов <CENTER> и </CENTER>, <H1> и </H1> и т. п. образует так называемый **контейнер**, придающий новые свойства тексту, который в него попадает.

Познакомимся с другими тегами, присутствующими на странице.

 — текст будет располагаться на новой строке (но без отступа, известного вам как «красная строка»). Такой тег не имеет отмены </BR>, а значит, и не образует контейнера.

<HTML> — указатель начала описания электронного документа на языке HTML.

<HEAD> — тег, располагающийся еще до описания самой страницы документа, в его заголовке. Текст внутри контейнера <TITLE> ... </TITLE> выводится программой просмотра страниц в верхней заголовочной части уже готового экрана с документом и помогает ориентироваться при поиске нужных документов.

<BODY> — указатель начала описания собственно странички документа. В нем присутствуют достаточно важные **атрибуты**. И один из них — BGCOLOR — определяет фон нашей странички.

Фоном может быть не просто цвет, но и любая картинка, которая сохранена в формате JPG или GIF. Для создания такого фона нужно использовать атрибут BACKGROUND (например, BACKGROUND="klen.gif"). При выборе картинки в качестве фона надо иметь в виду следующее:

Рис. 3.5 Размещение фоновой картинки на HTML-странице

1. Очень яркая картинка сильно затрудняет чтение текста и, как правило, плохо согласуется с другими картинками, которые вы помещаете на страницу.

2. Браузер «мостит» вашу страницу картинкой-фоном так, как это показано на рисунке 3.5. Для наглядности на этом рисунке слева мы оставили белые промежутки между фоновыми картинками. На самом деле картинки располагаются встык, как это показано справа. Иными словами, если фоновая картинка не очень большая, то необходимо обеспечить, чтобы происходила стыковка узора как по горизонтали, так и по вертикали. Если же картинка слишком большая, то, как правило, ее загрузка сильно тормозит получение остальной части документа. А что это за фон, когда только и приходится ждать, пока он загрузится, словно это наиважнейшая информация?

Приведем еще один любопытный атрибут тега <BODY>. Если мы запишем:

```
<BODY BACKGROUND="klen.gif" BGPROPERTIES=fixed>,
```

то в процессе просмотра странички текст будет двигаться, а фоновый узор останется на месте. В этом же теге задается и основной цвет текста на вашей страничке. Так, запись

```
<BODY BACKGROUND="klen.gif" TEXT="red">
```

обеспечит вывод красных букв. В таблице 3.1 приведены названия некоторых цветов.

У тега <BODY> есть еще несколько довольно важных атрибутов, но о них мы поговорим чуть позже.

Приведем описания других тегов, которые помогут украсить вашу страницу.

<MARQUEE> ... </MARQUEE> — вывод «бегущей строки».

<FONT COLOR="brown"> ... </FONT> — текст, расположенный внутри этого контейнера, будет выведен цветом, указанным после знака =, в частности отличающимся от цвета, заказанного в теге <BODY>.

<HR> — разделитель — горизонтальная линия, идущая через весь экран:

<BIG> ... </BIG> — текст будет отображаться шрифтом чуть большего размера, чем основной.

<SMALL> ... </SMALL> — текст будет отображаться шрифтом чуть меньшего размера, чем основной.

<B> ... </B> — **полужирный текст.**

<I> ... </I> — *текст, выделенный курсивом.*

<U> ... </U> — подчеркнутый текст.

<S> .... </S> — ~~перечеркнутый текст~~.

^{...} — текст для верхних индексов, например x^2.

_{...} — текст для нижних индексов, например A_1.

И наконец, тег

<IMG SRC="animal.jpg">

размещает на страничке картинку из графического файла, путь к которому указан с помощью атрибута SRC.

Таблица 3.1

Цвет	Название	Цвет	Название	Цвет	Название
Красный	Red	Белый	White	Темно-синий	Navy
Оранжевый	Orange	Голубой	Cyan	Коричневый	Brown
Желтый	Yellow	Синий	Blue	Фиолетовый	Magenta
Зеленый	Green	Светло-голубой	SkyBlue	Золотой	Gold
Черный	Black	Пурпурный	Purple	Серебряный	Silver

Подробное описание этих и других тегов, а также их атрибутов легко найти в любом справочнике по HTML.

Напомним, что теги <...> </...> образуют контейнер, придающий новые свойства тексту и картинкам, попавшим внутрь их.

Важно помнить, что

> контейнеры должны располагаться строго один внутри другого по принципу хорошо вам известной матрешки.

Например:

верная запись

```
<CENTER> ...
<H2>...
</H2>...
</CENTER>
```

неверная запись

```
<CENTER> ...
<H2>...
</CENTER>...
</H2>
```

Вопросы и задания

1. Почему потребовался специальный указатель для описания HTML-страниц?
2. Что означают теги </BODY> и </HTML>?
3. Каков признак тега, закрывающего контейнер?
4. Как могут располагаться контейнеры относительно друг друга?
5. Укажите, в каких примерах, приведенных ниже, контейнеры расположены правильно, а в каких неправильно.

а)
```
<MARQUEE>
<I>
</I>
</MARQUEE>
```

б)
```
<FONT COLOR="brown">
<H2>...
</FONT>
</H2>
```

в)
```
<MARQUEE>
<I>
 <CENTER>
 </I>
</CENTER>
</MARQUEE>
```

6. Какие контейнеры могут появляться в документе только один раз?
7. Приведите примеры непарных тегов (т. е. таких, для которых нельзя создать контейнер).

§ 30 Гиперссылки в HTML

Почему же HTML называется гипертекстовым языком? Приставка «гипер-» означает «над, сверх, по ту сторону». Как мы уже говорили, наша страничка состоит из двух этажей. На одном из них, так сказать, парадном, расположен ваш документ во всей его красе, а на другом — «подвальном» — его описание, с помощью которого программа просмотра и создает парадный этаж. На парадном этаже располагаются обычный текст и обычные картинки.

Возможно, вы обратили внимание на то, что отдельные слова и картинки на парадном этаже выделены особо. Это означает, что под ними находятся **ссылки** на другие электронные документы или даже на программы, которые будут выполнены в случае выбора этих элементов. Понять, как именно работают те или иные ссылки, можно, если проанализировать «подвальный» этаж странички, содержащий ее описание.

Что располагается в «подвале» и какие бывают ссылки, мы уже рассказывали (см. § 28). А как создается хотя бы самая простая ссылка — ссылка на другую страницу? Для этого существует специальный контейнер с атрибутами: <A ...> ... </A>. Как вы помните, действия контейнера уточняются с помощью атрибутов. В нашем случае необходим атрибут, указывающий, куда производится ссылка. Вот пример соответствующего контейнера:

```
<A HREF="file:///s|/html/8b2005/gosha/gosha.htm">
        Это очень интересная страничка Гоши! </A>
```

Все элементы оформления, будь то текст или картинка, расположенные внутри контейнера <A ...> ... </A>, становятся не просто текстом и не просто картинкой, а **гипертекстовой ссылкой**. При переводе на эту ссылку указателя мыши он видоизменяется, а если при этом щелкнуть по клавише мыши, то произойдет переход на страницу, которая соответствует файлу, полный путь к которому описан в атрибуте HREF.

Если файл вашей второй страницы расположен в одной папке с файлом, откуда вызывается ссылка, полный путь можно и не писать. Так, если две ваши страницы, которым соответствуют файлы 23.htm и 23a.htm, находятся в одной папке, то в файле 23.htm может быть, например, вот такой вызов второй страницы:

```
<A HREF="23a..htm"> А это мой друг Дмитрий! </A>
```

Можно сослаться не на описание странички, а прямо на фотографию. Такая ссылка будет выглядеть следующим образом:

```
<A HREF="file:///s|/html/8a2005/gosha/utki.jpg">
        Мы с братом очень любим уток! </A>
```

Или так, если фотография располагается в одной папке с описанием страницы:

<A HREF="8a34.jpg"> Это сквер возле нашей школы </A>

Если вы ссылаетесь на фотографию или картинку напрямую, то она, разумеется, будет располагаться вовсе не по центру и уж конечно без всякого фона. Если вам это не нравится, придется сделать ссылку на описание страницы на языке HTML, а в ней разместить картинку так, как вы считаете нужным. В этом случае нелишней будет и подпись под рисунком или фотографией.

Немного отвлекаясь от ссылок, опишем еще два полезных атрибута тега <IMG>.

Наверняка вам доставляло много неудобств то, что картинки и фотографии часто были не тех размеров, каких бы вам хотелось. Между тем изменить их размеры очень просто. Достаточно в тег <IMG>, отвечающий за размещение графических файлов, добавить дополнительный атрибут:

<IMG SRC="file:///s|/html/peop/besinger/dpg116.jpg" WIDTH=200>

или

<IMG SRC="file:///s|/html/peop/besinger/dpg116.jpg" HEIGHT=250>

Атрибут WIDTH определяет ширину картинки или фотографии в пикселях. Атрибут HEIGHT — ее высоту. Значит, если разрешение экрана — 800×600 пикселей, в первом случае картинка по ширине займет примерно четверть экрана. Во втором примере по высоте — примерно треть экрана.

Вместо пикселей в качестве единицы измерения можно использовать проценты. Однако если вы думаете, что атрибут WIDTH = 50% уменьшит картинку в 2 раза, то вы ошибаетесь. Этот параметр означает, что картинка займет ровно половину ширины окна, выделенного под программу просмотра. Если окно уменьшить — соответственно уменьшится и картинка. Очень может быть, что она и растянется, если ее исходная ширина меньше половины ширины экрана.

Необходимо помнить, что эти атрибуты изменяют вовсе не размеры самой картинки, а лишь ее отображения на страничке. Иными словами, они совершенно не влияют на время ее загрузки. Картинка должна быть получена целиком, а уж затем программа просмотра, работающая на вашем компьютере, будет изменять размеры ее отображения.

Но вернемся к ссылкам. Программа просмотра настроена таким образом, чтобы выделять текстовые ссылки другим цветом. А если ссылкой является фотография или картинка, то вокруг нее появляется цветная рамка.

Бывает так, что стандартный цвет, которым выделяются ссылки, вовсе не согласуется с выбранным вами фоном. Например,

в качестве такого стандартного цвета выбран синий, а цвет всего текста на экране тоже синий, и ссылки становятся просто невидимыми. Для того чтобы изменить цвет ссылки, существуют специальные атрибуты тега <BODY>:

```
<BODY BACKGROUND="kairos.jpg"  VLINK="yellow"  LINK="brown"
                    ALINK="red">
```

Атрибут LINK определяет цвет текста, за которым скрывается ссылка, или самой ссылки. Атрибут VLINK определяет цвет ссылки, которую уже «посетили». Атрибут ALINK определяет цвет ссылки, по которой щелкнули мышкой, но она еще не загрузилась в наш компьютер. Этот атрибут актуален, когда идет работа в глобальной сети и приходится долго ждать загрузки. Если же загрузка происходит быстро, то изменение цвета происходит за долю секунды и совершенно незаметно для пользователя.

Вопросы и задания

1. С помощью какого контейнера строится гипертекстовая ссылка?
2. Для чего предназначен атрибут HREF?
3. Какие атрибуты тега <BODY> позволяют управлять цветами гипертекстовой ссылки?
4. Посмотрите еще раз на записанную в объяснительном тексте команду управления цветом гиперссылок. Какой цвет в соответствии с этой командой имеет ссылка, которой еще не пользовались? А та ссылка, которой уже воспользовались?
5. Предположим, вы не хотите, чтобы гипертекстовая картинка была окаймлена цветной рамкой. Как этого можно добиться?

§ 31 Оформление HTML-страницы

Выполняя лабораторную работу № 10, вы поместили на своей странице несколько изображений. Но появиться они могли не так красиво, как от них ожидалось. Например, так, как показано на рисунке 3.6.

А нельзя ли сделать так, чтобы в окне программы просмотра появилась полоса горизонтальной прокрутки и картинки располагались не одна под другой, а рядом?

Разумеется, это можно сделать. Достаточно воспользоваться тегом <TABLE>. Этот тег служит для описания таблиц. Описание производится следующим образом:

- вначале заказывают таблицу (тег <TABLE>);
- затем описывют первую строку таблицы (в контейнере <TR> ... </TR>);
- внутри первой строки по очереди описывают содержимое каждой ячейки (в контейнерах <TD> ... </TD>);
- описывают вторую строку (в контейнере <TR> ... </TR>);

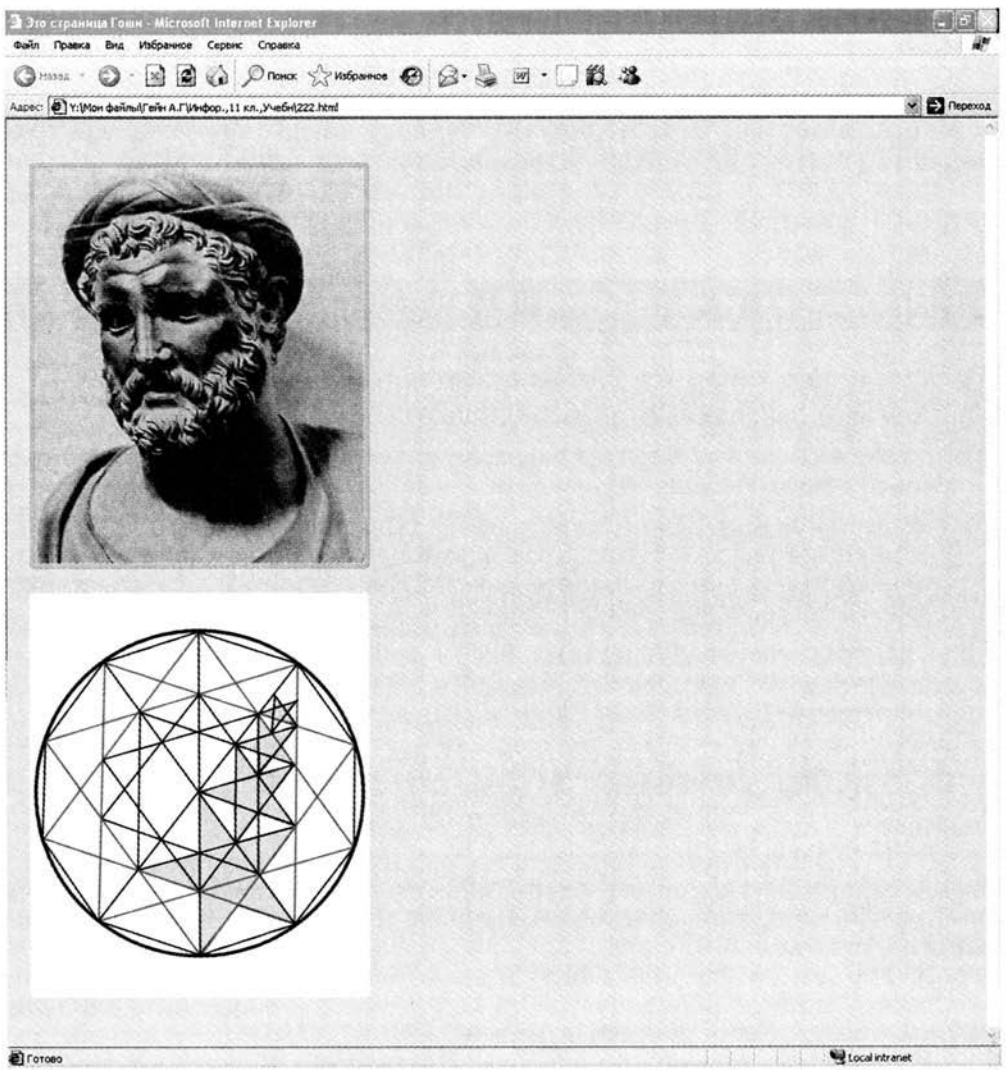


Рис. 3.6 Расположение рисунков на странице в случае простого написания тегов <IMG>

- внутри второй строки по очереди описывают содержимое каждой ячейки (в контейнерах <TD> ... </TD>);
- ...и так далее описываем все строки и все ячейки в них;
- закрываем описание таблицы (тег </TABLE>).

Давайте вспомним таблицу 3.1 и посмотрим, как выглядит ее описание с использованием тега <TABLE>. Вот это описание:

```
<HTML>
<HEAD><TITLE> Названия цветов </TITLE>
</HEAD>
<BODY background="stars.gif">
<TABLE border=1>
<! Описание первой строки
<TR>
<TD> <B><P ALIGN="CENTER">Цвет</P></B> </TD>
<TD> <B><P ALIGN="CENTER">Название</P></B></P></TD>
<TD> <B><P ALIGN="CENTER">Цвет</P></B></TD>
<TD> <B><P ALIGN="CENTER">Название</P></B></TD>
<TD> <B><P ALIGN="CENTER">Цвет</P></B></TD>
<TD> <B><P ALIGN="CENTER">Название</P></B></TD>
</TR>
<! Описание второй строки
<TR>
<TD><font color="red">Красный</font></TD>
<TD>Red</TD>
<TD><font color="white">Белый</font></TD>
<TD>White</TD>
<TD><font color="navy">Темно-синий</font></TD>
<TD>Navy</TD>
</TR>
<! Описание третьей строки
<TR>
<TD><font color="orange">Оранжевый</font></TD>
<TD>Orange</TD>
<TD><font color="cyan">Голубой</font></TD>
<TD>Cyan</TD>
<TD><font color="brown">Коричневый</font></TD>
<TD>Brown</TD>
</TR>
<! Описание четвертой строки
<TR>
<TD><font color="yellow">Желтый</font></TD>
<TD>Yellow</TD>
<TD><font color="blue">Синий</font></TD>
<TD>Blue</TD>
```

```
<TD><font color="magenta">Фиолетовый</font></TD>
<TD>Magenta</TD>
</TR>
<! Описание пятой строки
<TR>
<TD><font color="green">Зеленый</font></TD>
<TD>Green</TD>
<TD><font color="SkyBlue"> Светло-голубой</font></TD>
<TD>SkyBlue</TD>
<TD><font color="gold">Золотой</font></TD>
<TD>Gold</TD>
</TR>
<! Описание шестой строки
<TR>
<TD><font color="black">Черный</font></TD>
<TD>Black</TD>
<TD><font color="purple">Пурпурный</font></TD>
<TD>Purple</TD>
<TD><font color="silver">Серебряный</font></TD>
<TD>Silver</TD>
</TR>
```

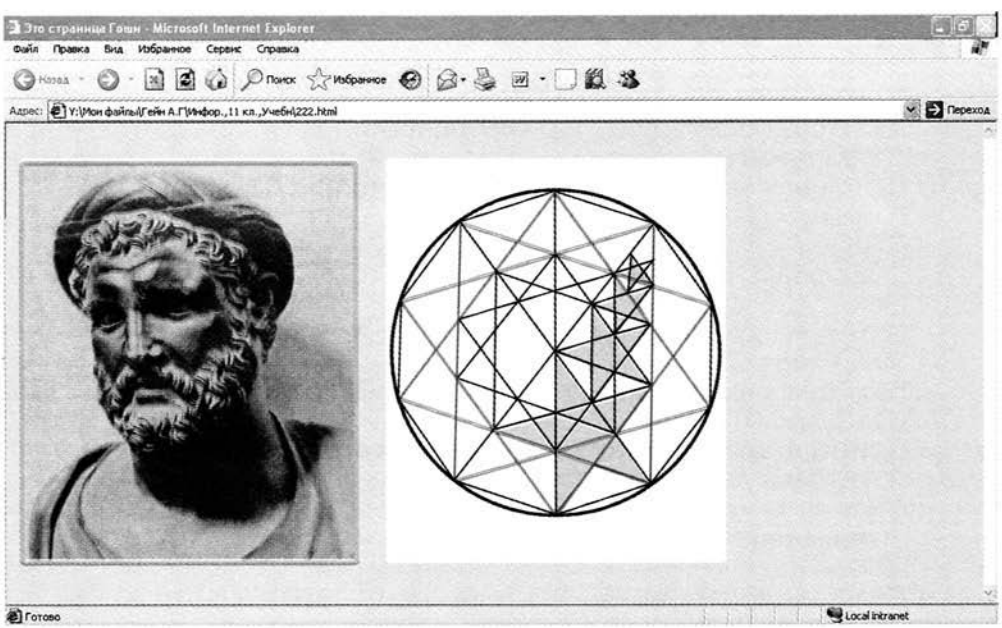


Рис. 3.7 Вид страницы с рисунками, для размещения которых использовался контейнер <TABLE>

```
</TABLE>
</BODY>
</HTML>
```

Несмотря на такое пугающе длинное описание, мы надеемся, что вам все понятно. Атрибут BORDER тега <TABLE> заказывает рисование сетки по границам ячеек таблицы, а специальные символы <! в начале строки говорят, что после них идет комментарий. На них при создании страницы по вашему описанию программа просмотра не обращает внимания.

А теперь расположим две наши картинки из рисунка 3.6 рядом. Скажем, так, как изображено на рисунке 3.7.

Еси бы вторая картинка не поместилась в окно программы просмотра, она все равно расположилась бы рядом с первой. В этом случае в нижней части окна появляется горизонтальная полоса прокрутки. С ее помощью можно просмотреть информацию, расположенную в правой части окна.

И все это достигается за счет простейшей таблицы, содержащей всего лишь одну строку с двумя ячейками. Иными словами, структура таблицы для показанной выше страницы такая:

```
<TABLE>
<TR>
<TD>...</TD>
<TD>...</TD>
</TR> </TABLE>
```

Внутри контейнеров <TD> ... </TD> расположены, сами понимаете, теги <IMG>.

Вопросы и задания

1. Какие теги используются для размещения в табличном виде информации на HTML-странице? Образуют ли эти теги контейнер?
2. Как можно зафиксировать расположение различных информационных объектов на HTML-странице?
3. Просмотрите еще раз приведенное в этом параграфе описание таблицы на языке HTML.
   а) Зачем в описании таблицы использован тег <FONT>? Чем будет отличаться изображение этой таблицы на экране компьютера от таблицы 3.1?
   б) Для каких целей там же служит атрибут ALIGN="CENTER" контейнера <P>...</P>.
   в) Узнайте, для чего служит атрибут ALIGN="CENTER" при использовании в теге <IMG>. Какие он может принимать значения?
4. На рисунке 3.7 надо сделать так, чтобы обе картинки имели одинаковую высоту, а подписи под ними расположились по центру. С помощью каких тегов и атрибутов этого можно добиться?

§ 32 Объекты других приложений в HTML

Нередко случается так, что информационные объекты, которые вы хотите разместить на своей HTML-странице, оказываются изготовленными средствами других приложений. Например, это может быть текстовый документ, подготовленный с помощью Microsoft Word.

Можно предложить два способа поместить документ в формате Microsoft Word на HTML-страницу. Но прежде давайте вспомним о контейнере.

Контейнер — это, как вы знаете, пара тегов, придающих новые свойства объектам, расположенным внутри их. Важно помнить, что контейнеры должны располагаться строго один внутри другого.

Самое интересное, что контейнером может быть не только пара HTML-тегов, но и несравненно более сложные объекты. Например, программа просмотра HTML-страниц. А содержимым такого контейнера может служить текстовый процессор Microsoft Word. Чисто внешне это будет выглядеть так, как если бы внутрь окна программы просмотра вставили окно текстового процессора Word. При этом к пунктам главного меню программы Microsoft Internet Explorer добавятся пункты главного меню Word.

Внутрь контейнера программы просмотра может попасть и окно графического редактора CorelDraw!, и вообще окно любой программы, разработчики которой придерживались оговоренного стандарта. Возможно, вам пригодится его название: ActiveX.

> Для того чтобы программы могли использовать окна друг друга, как контейнер, они должны поддерживать технологию ActiveX.

Текстовый редактор, расположенный внутри контейнера другой программы, немного отличается по своим возможностям от варианта «независимого» запуска. И в первую очередь ограничена его возможность работы с файлами.

Так, невозможно использовать текстовый редактор в многооконном режиме. В случае попытки открыть второй файл процессор Word просто стартует, как еще одна независимая программа в своем собственном окне.

Каким же образом можно воспользоваться программными контейнерами? Достаточно просто сделать ссылку не на HTML-файл, а на файл в формате Microsoft Word или, скажем, CorelDraw!.

Когда вы, выполняя лабораторную работу № 11, проделаете это в первый раз, обратите внимание на время, в течение которого откроется нужный вам документ. Оно гораздо больше открытия HTML-документа со сравнимым объемом информации, и это вполне понятно. Две довольно сложные и объемные программы, работающие одна внутри другой, как в контейнере, равносильны многозадачному запуску, когда системные ресурсы используются уже несколькими приложениями. Иными словами, это примерно то же самое, что и работа с двумя окнами, в одном из которых, скажем, программа просмотра, а в другом — графический редактор. Вы наверняка замечали, как это сказывается на быстродействии компьютера.

Тем не менее в Интернете довольно много документов именно в формате Microsoft Word и Adobe Acrobat (их расширение — pdf). Чем же они лучше привычных для нас HTML-файлов?

Во-первых, документы в формате Microsoft Word очень легко править. Для этого фактически не требуется никаких особых навыков.

Во-вторых, документы в формате текстового редактора гораздо больше приспособлены к печати на принтере, чем HTML-файлы. Поэтому многие отчеты, аналитические записки, планы работы и т. п. хранятся тоже в виде файлов Microsoft Word.

Файлы же Adobe Acrobat — это фактический стандарт электронных книг, в виде которых публикуется очень большое количество технической документации, в том числе и руководства пользователей.

Такого сорта публикацию документов часто используют во «внутреннем Интернете», или, по-другому, в **Интранете**, когда информация внутри одной организации передается по высокоскоростной локальной сети.

Скажем еще о возможности преобразования документа, созданного в текстовом процессоре Microsoft Word, непосредственно в HTML-формат. Для того чтобы ею воспользоваться, необходимо выбрать пункт главного меню *Файл*, подпункт *Сохранить в формате HTML*. Конечно, не все элементы оформления вашего документа в формате Microsoft Word будут точно так же выглядеть и в формате HTML, но в общем и целом такое преобразование обеспечивает удовлетворительное качество.

Вопросы и задания

1. Какую технологию должны поддерживать программы, чтобы они могли использовать окна друг друга?
2. Какие два пути существуют, чтобы поместить Word-документ на HTML-страницу?

3. В чем преимущества размещения на HTML-странице информационных объектов в тех форматах, в которых они изготовлены в соответствующих приложениях?

4. Назовите некоторые ограничения в работе приложений, посредством которых создаются или обрабатываются информационные объекты непосредственно на HTML-странице.

§ 33 Компьютерные словари и системы перевода текстов

Расширение возможностей применения компьютерной техники определено двумя основными факторами: ростом мощности компьютеров (памяти, быстродействия) и разработкой эффективных информационных моделей, включая алгоритмы обработки информации. Работая совместно, языковеды, специалисты в области математического моделирования и разработчики эффективных алгоритмов создали так называемые **системы компьютерного перевода текстов**. Такие системы основываются на моделях грамматики естественного языка и словарях, в которых словам и словосочетаниям одного языка сопоставляются варианты их перевода на другой язык. Обычно такие системы включают в себя те или иные модели искусственного интеллекта (о них мы рассказывали в главе 4 учебника для 10 класса). Системы компьютерного перевода могут быть адаптивными (т. е. приспосабливающимися к особенностям переводимого текста) и самообучающимися. Ясно, что если переводится текст, относящийся к проектированию и строительству зданий, то ему присущи своя терминология, свои речевые обороты и т. п. В этом случае система перевода автоматически подстраивается — как было сказано выше, адаптируется — к такому тексту, выбирая при переводе из множества существующих вариантов тот, который наиболее близок к теме сообщения.

Любая система компьютерного перевода работает тем эффективнее, чем более формализован текст, который подвергается переводу. Такими формализованными текстами в первую очередь являются технические описания, инструкции, нормативные документы, акты, предписания и другая так называемая деловая проза. Ее достаточно высокая формализованность — следствие того, что разными людьми она должна пониматься однозначно. Вспомните, о чем мы рассказывали в учебнике для 10 класса: средством для достижения однозначности понимания информации является формализация ее представления.

Но перевод художественной прозы и поэтических текстов остается искусством, т. е. деятельностью, доступной только человеку.

Без словарей не сможет работать никакая система компьютерного перевода. Впрочем, электронные словари могут успешно использоваться и независимо от программ компьютерного перевода. Нередко такие словари являются многоязычными. Среди российских словарей наиболее популярными являются словари Lingvo, Контекст и Мультилекс.

Вопросы и задания

1. Что такое система компьютерного перевода текстов? Для перевода каких текстов ее целесообразно использовать?
2. В чем вы видите причины востребованности систем компьютерного перевода текстов?
3. Приведите примеры текстов, относящихся к деловой прозе.
4. Укажите, в чем вы видите преимущества компьютерного словаря перед словарем в бумажном исполнении. Постарайтесь назвать не менее трех различных преимуществ.
5. Нередко для проверки качества работы системы машинного перевода в качестве теста используют так называемый двойной перевод текста: сначала текст переводят на иностранный язык, а затем с помощью той же системы переводят полученный текст на исходный язык. Однажды в качестве тестового примера был взят текст: «Телом слаб, но духом крепок». После двойного перевода получилась фраза: «Спирт крепок, а мясо протухло». Объясните причины такого результата.

§ 34 Компьютерная обработка графических информационных объектов

Работе с компьютерной графикой вы наверняка учились на уроках информатики еще в 8 и 9 классах. Именно тогда вы познакомились с основными видами компьютерных программ, предназначенных для создания и обработки изображений. Напомним их:

- графические редакторы; с их помощью создаются и редактируются рисунки в цифровом представлении;
- программы обработки фотоизображений; они позволяют добавить изображению яркость или контрастность, отретушировать изображение, устранив какие-либо недостатки, создать разнообразные эффекты;

- программы, позволяющие объединять текст и рисунки, формировать из них макет книги или журнала; такие программы называются программами компьютерной верстки;
- программы создания слайд-фильмов и мультипликации;
- программы создания презентаций; графическое и звуковое сопровождение, синтезируемое с их помощью, — важное подспорье при любом выступлении.

Напомним также, что в зависимости от способа формирования изображений компьютерную графику подразделяют на **векторную** и **растровую**. В случае растровой графики картинка на экране компьютера формируется заданием каждого пикселя отдельно. При векторном способе геометрический образ, создаваемый группой пикселей, описывается формулами.

С векторной графикой вы, в частности, имели дело, когда создавали рисунки средствами текстового процессора Word. Теперь вы на примере возможностей редактора Adobe Photoshop подробнее познакомитесь с растровой графикой. Adobe Photoshop — это мощный профессиональный пакет, и в наши планы не входит рассмотрение всех возможностей этого пакета.

Основные инструменты, с помощью которых создается изображение, примерно одинаковы во всех графических редакторах, и мы подробнее поговорим о них в описании лабораторной работы. А сейчас обсудим некоторые принципиальные особенности данного редактора.

Первая из них состоит в том, что, кроме простых инструментов выделения геометрической формы, в Adobe Photoshop имеются инструменты, позволяющие выделять объекты сложной формы, а также области на основе цвета. Для этого используются различные варианты инструмента **Лассо**.

Один из вариантов этого инструмента, обозначенный пиктограммой  , позволяет обрисовывать выделяемую область движением мыши по краю нужного объекта. При этом нужно держать нажатой левую клавишу мыши.

Многоугольное (полигональное) лассо — его пиктограмма выглядит так:  — предназначено для получения сложной выделенной области, ограниченной отрезками прямых. Каждый щелчок левой клавишей фиксирует точку начала или конца отрезка. Двойной щелчок левой клавишей завершает построение выделенной области соединением первой и последней точек. Другой способ завершения — щелчок в первой точке, с которой было начато построение выделения. При этом у курсора мыши, имеющего вид значка инструмента, снизу справа появляется окружность, означающая замыкание выделяемой области.

Инструмент  — **Магнитное лассо** позволяет выделить в изображении область неправильной формы перемещением мыши вдоль границы области. В местах изменения кривизны (перегиба фигуры)

щелкайте левой клавишей, чтобы кривая выделения меняла на этом участке свое направление. Линия выделения сама изгибается за курсором по разделу цветовых оттенков. Окончание выделения — двукратный щелчок левой клавишей. После этого контур автоматически замкнется.

Инструмент **Волшебная палочка** (вот ее пиктограмма: ) позволяет создавать выделенную область, исходя из цветовой близости рядом расположенных пикселей. Этот инструмент удобно применять для выделения областей с одинаковыми или близкими цветами. В панели свойств этого инструмента имеется параметр Допуск, который определяет цветовой диапазон выделения. Значение этого параметра может быть от 0 до 255 и определяется обычно методом подбора. Чем меньше значение, тем ближе будут цвета и оттенки, которые попадут в выделенную область. При допуске 0 выделятся только элементы указанного цвета. При допуске 255 в выделенную область попадут пиксели всех цветов, а значит будет выделено все изображение.

Над выделенными с помощью любого инструмента фрагментами можно выполнять разнообразные операции: копирование, перемещение в пределах данного изображения или в другие документы, удаление и многое другое.

Второй важной особенностью Adobe Photoshop является механизм слоев. Редактировать растровый графический объект сложной структуры весьма непросто. Чтобы облегчить эту работу, можно распределить фрагменты изображения по отдельным слоям — редактировать его станет намного удобнее.

Опишем типы слоев.

Когда вы даете команду создания нового документа, Adobe Photoshop создает один экранный лист указанного вами размера, называемый **Фоном**. Это особый вид слоя. Он не может иметь прозрачных участков, так как всегда является самым нижним. Но зато над ним можно надстроить еще целый ряд дополнительных слоев — прозрачных пленок, называемых **изобразительными слоями**. В каждом изображении может быть до 100 слоев. На каждом слое вы создаете фрагмент рисунка, а остальная часть изобразительного слоя остается прозрачной. Сквозь прозрачные участки слоя видны нижележащие слои. Фрагмент на данном слое вы можете редактировать независимо от фрагментов рисунка, расположенных на других слоях.

Особый вид слоев — **корректирующие слои**. Они могут работать как маски, пропуская из нижележащих слоев в общее изображение то, что попадет в «отверстие» в маске, могут применять различные эффекты к нижележащим слоям, осуществлять цветовую и тоновую коррекцию отдельных слоев или групп слоев. Меняя параметры корректирующего слоя, вы получаете разные результаты, никак не затрагивая самого изображения.

Для выполнения действий над слоями имеется специальная палитра, вызов которой осуществляется через меню *Окно/Показать слои*.

В палитре имеется список слоев с миниатюрами — уменьшенными изображениями. Справа от миниатюры — название слоя. В каждый момент времени можно редактировать только один активный слой. Его строка выделена в палитре синим цветом. Для активизации другого слоя следует щелкнуть по строке с его названием в палитре.

Когда слой активизирован, то в верхней части палитры можно изменить уровень непрозрачности для накладываемой краски: 0% означает, что слой остается прозрачным; 100% означает, что исходное изображение закрашивается полностью. Сразу оговорим, что данное действие можно выполнять, только если установлен режим *Норма*; именно он устанавливается по умолчанию.

Можно выбрать другой режим. В зависимости от выбранного режима Adobe Photoshop позволяет смешивать наносимые цвета с имеющимися на изображении самыми разнообразными способами (их около 20).

На палитре *Слои* содержатся переключатели режимов, позволяющие блокировать (запрещать) закраску прозрачных пикселей, изменение цвета непрозрачных пикселей активного слоя, рисование на нем, сдвиг слоя относительно других.

К активному слою могут быть применены **эффекты**, изменяющие отображение закрашенных пикселей выбранным способом, создающие свечение краев объектов, окаймление границ объектов и т. д. Эффект не изменяет слой и не модифицирует ни одного пикселя. Он модифицирует только отображение на экране, сам же рисунок остается неизменным.

Эффекты применимы к слоям любого типа. Они действуют лишь на видимую часть слоя, т. е. не влияют на прозрачные и замаскированные (защищенные от воздействий) области слоя. Комплект выбранных для слоя эффектов называется **стилем** слоя. Чтобы применить тот или иной эффект, нужно щелкнуть кнопку *Добавление стиля слоя* внизу палитры *Слои*, выбрать эффект из списка, а затем в диалоговом окне задать его параметры. Рядом с названием слоя, где используется какой-либо эффект, справа появляется условный значок **f**.

Пока вы работаете в Adobe Photoshop, выполняя шаг за шагом сложную послойную работу, изображение может быть сохранено только в файле с расширением psd. После того как изображение будет полностью создано, необходимо включить видимость всех слоев и выполнить их объединение командой *Слой/Объединить видимые*. После этой операции вам будут доступны команды сохранения в широко применяемых форматах gif, jpg и jpeg, где размеры файлов уменьшаются во много раз по сравнению с файлами с расширением psd.

Вопросы и задания

1. Перечислите основные виды программного обеспечения, предусматривающего работу с графическими объектами, и для каждого вида укажите его назначение.
2. В чем отличие векторной графики от растровой?
3. Назовите преимущества, которые дает механизм слоев для формирования изображения.
4. Перечислите типы слоев, существующих в Adobe Photoshop. Каково назначение слоя каждого типа?
5. Какие способы выделения обрабатываемой области рисунка предусмотрены в Adobe Photoshop?

§ 35 Компьютерная обработка цифровых фотографий

В мире современных технологий важным источником видеоинформационных объектов стал цифровой фотоаппарат. Преимущества цифровой фотографии очевидны:

- легкость создания изображения;
- возможность проверки на месте съемки качества изображения в целом;
- возможность передачи созданного видеоинформационного объекта по телекоммуникационным сетям;
- легкое встраивание в другие виды электронных информационных объектов (например, в текст документа);
- возможность обработки средствами компьютерной графики.

Вот об обработке цифровых фотографий у нас и пойдет речь в этом параграфе.

Первое, о чем мы расскажем, — это тональная и цветовая коррекция. **Коррекцией** называется изменение характеристик изображения, позволяющее добиться нужного качества.

Нередко фотографии по тем или иным причинам (обычно связанным с условиями съемки) оказываются нечеткими, блеклыми или недостаточно контрастными. **Контраст** — степень тонового различия между областями изображения. **Яркость** — характеристика цвета, определяющая его интенсивность в изображении. Фактически яркость определяется числом пикселей данного цветового оттенка.

Рис. 3.8 Гистограмма яркости

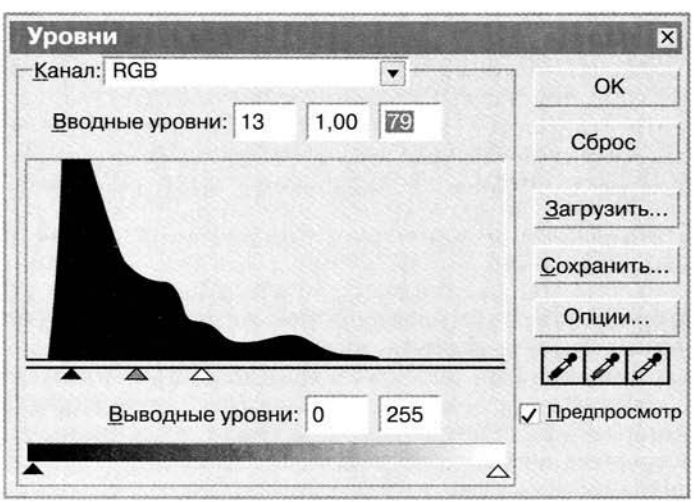


Рис. 3.9 Диалоговое окно изменения уровней яркости

Рис. 3.10 Гистограмма яркости после преобразования уровней

Adobe Photoshop позволяет для любого рисунка посмотреть распределение яркостей по тонам. Для этого надо воспользоваться диалоговым окном в меню *Изображение/Гистограмма*. Представленная в этом окне диаграмма (рис. 3.8) отображает по горизонтали значение тона, а по вертикали количество пикселей данного тона в изображении, т. е. яркость, освещенность. К примеру, из диаграммы, представленной на рисунке 3.8, видно, что яркость исходной фотографии недостаточна: отсутствуют пиксели светлых тонов, зато очень много пикселей темных тонов. Вывод: требуется коррекция. Отметим, что гистограмма используется только для получения информации об изображении и для корректировки распределения яркости в рисунке не предназначена!

Что касается настройки тонов и яркости изображения, то для этого надо воспользоваться меню *Изображение/Коррекция/Уровни*. В появившемся диалоговом окне можно передвинуть левый маркер на начало диапазона, а правый маркер на один из пиков (рис. 3.9). Тем самым расширяется область светлых тонов на выходе.

Если вы посмотрите на гистограмму после этого преобразования, то увидите ее такой, какой она представлена на рисунке 3.10.

Видно, что распределение стало более равномерным. Что при этом происходит с фотографией, вы увидите при выполнении лабораторной работы.

Самый простой способ тональной коррекции всего изображения или выделенного фрагмента осуществляется с помощью инструмен-

тов настройки в диалоговом окне, вызываемом командой *Изображение/Коррекция/Яркость-Контраст* (рис. 3.11). Перемещения маркеров вдоль шкалы изменяют яркость изображения (освещенность) и контраст (четкость граничных линий).

На гистограмме проведенное изменение яркости отразилось так, как показано на рисунке 3.12, — начало диапазона яркости сместилось вправо на те самые 44 единицы (которые видны на рисунке в окне над шкалой *Яркость*) из 256 общей длины диапазона. Есть и другие, более тонкие инструменты коррекции тонов (например, гамма-коррекция); с ними вы познакомитесь, выполняя лабораторную работу.

Аналогично можно контролировать и регулировать состояние контрастности.

После завершения тональной коррекции, если у вас цветное изображение, наступает время более сложной и более субъективной коррекции цветов.

Специфика работы с цветом заключается в том, что изменение одного цвета обязательно отразится на других. Поэтому в основе любой цветовой коррекции лежит не настройка отдельных цветов, а настройка **баланса цветов**. Для этого используется цветовая модель HSB. Цветовой круг, о котором шла речь в § 15 (см. рис. 2.2 и форзац учебника), наглядно представляет взаимозависимость цветов. Каждый цвет находится напротив дополняющего его на другом конце диагонали. Сам цвет находится между цветами (на окружности), с помощью которых получен.

Цвета, лежащие напротив друг друга, взаимно связаны. Уменьшая в рисунке содержание одного, вы увеличиваете содержание противоположного (дополняющего). Вызов окна настройки баланса цветов осуществляется командой *Изображение/Коррекция/Цветовой баланс*.

Рассказанное нами отражает лишь реализацию общих принципов кодирования цветной графики при создании графических информационных объектов. Реальные возможности Adobe Photoshop намного шире и позволяют создавать не только информационные объекты, но и поистине художественные образы.

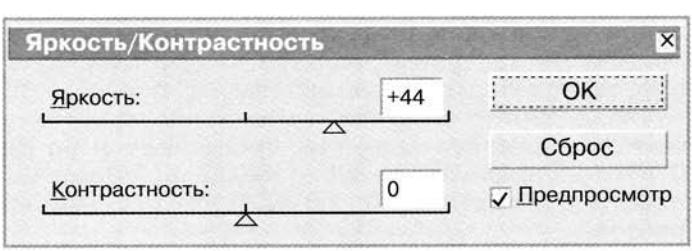


Рис. 3.11 Диалоговое окно изменения уровней яркости и контрастности

Рис. 3.12 Гистограмма после изменения уровня яркости

Вопросы и задания

1. В чем состоят преимущества цифровой фотографии перед пленочной?
2. Что такое контраст? Что такое яркость изображения?
3. Для чего используется *Гистограмма*?
4. Почему при цветовой коррекции нельзя настроить ровно один цвет?
5. Рассмотрите еще раз рисунок 3.9. В окне Канал указано RGB. Что, на ваш взгляд, будет на экране и какой смысл имеет преобразование уровня, если в окне Канал стоит только R? А если только B?

§ 36 Компьютерные презентации

Компьютерные презентации, как следует из смысла самого слова «презентация», предназначены для того, чтобы представить интересующую информацию как можно в более доступном виде. Эффективность восприятия информации возрастает, если она будет представлена в разных видах — видео, звуковой, текстовой. Такое представление информации принято называть **мультимедийным** (от лат. multi — много). Для создания мультимедийных до-

кументов имеется специальное инструментальное программное обеспечение. Мы расскажем об одном из средств создания мультимедийных продуктов — программе создания и управления электронной презентацией PowerPoint.

Нередко презентации создаются для сопровождения выступления или доклада по какому-либо вопросу. Ведь хочется, чтобы выступление было интересным, информационно насыщенным, запоминающимся. И неважно, кто будет вашим слушателем: члены строгого жюри на конференции по защите исследовательских работ, или ваши друзья по клубу, где вы рассказываете об очередном своем путешествии, или школьники более младшего класса, в котором вас попросили рассказать о чем-то, что вы хорошо знаете. Презентация, которую вы подготовите заранее, будет вашим надежным помощником.

Учитель любого школьного предмета не откажется от вашей умной презентации изучаемого материала. Это могут быть знаменитые памятники истории и некоторые доказательства теоремы Пифагора, редкие животные нашей страны или вашего родного края, история компьютерной техники, биография знаменитого писателя и т. п. Любая тема может стать основой для создания учебной презентации.

На выставке презентация может освободить вас от необходимости много раз повторять одно и то же разным посетителям. Компьютер в отличие от человека никогда не забудет сказать все нужные слова, показать все необходимые диаграммы, продемонстрировать нужные изображения. Удачная презентация способствует расширению круга партнеров. Презентацию, выполняемую компьютером автоматически, т. е. без участия человека, обычно называют слайд-фильмом.

Основным структурным элементом презентации, создаваемым в PowerPoint, является **слайд**. Во время демонстрации на смену одному слайду приходит другой. Работу по созданию презентации можно разбить на следующие этапы:

- подготовка: выбор темы, определение цели, круга слушателей, содержания и структуры презентации в целом и каждого слайда;
- создание слайдов: размещение объектов на слайде, выбор фона, применение эффектов, подключение звука;
- создание последовательности слайдов: определение переходов от слайда к слайду, создание меню и т. п.;
- репетиция выступления с презентацией;
- редактирование презентации: внесение изменений по результатам репетиции.

Первый этап полностью ложится на вас. А вот про второй мы сейчас расскажем.

Прежде всего надо выбрать макет будущего слайда. Компьютер любезно предложит вам несколько вариантов (рис. 3.13). Для начала мы советуем вам выбрать макет без разметки — ведь интересно все попробовать самому.

На поле слайда вы можете добавлять самые разнообразные объекты: это и стандартные фигуры, и надписи, и объекты из WordArt, и рисунок из ClipArt. А может быть, это будет ваш личный рисунок. Все это делается так же, как, например, в Microsoft Word.

Необходимо позаботиться и о фоне слайда. Его можно создавать самостоятельно, а можно воспользоваться *Шаблонами оформления*. В первом случае вы действуете через меню *Формат/Фон*. Во втором случае вы пользуетесь трудами художников-дизайнеров, и все слайды получаются в едином стиле. В этом случае используют меню *Формат/Оформление слайда*. Чтобы отменить используемый шаблон оформления, надо в окне *Фон* поставить галочку в поле *Исключить фон образца*.

Средствами PowerPoint можно заставить двигаться любой размещенный на слайде объект. В этом случае говорят, что к объекту добавлена анимация. Само слово «анимация» означает «оживление» (от англ. animal — животное). Настройка анимации осуществляется через меню *Показ слайдов/Настройка анимации*.

В первую очередь назначаются порядок и время анимации. Можно выбрать два режима для запуска объекта в движение: по

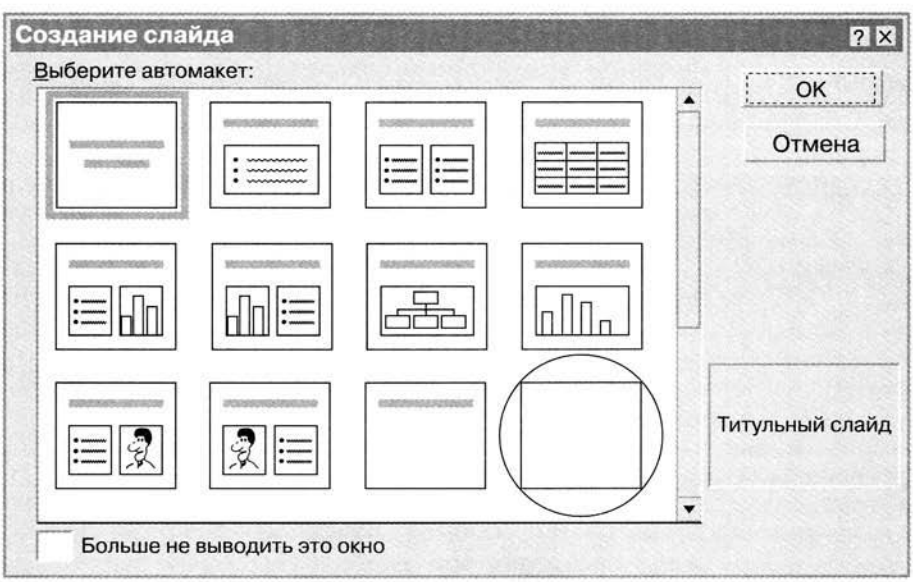


Рис. 3.13 Окно выбора макета будущего слайда

щелчку мыши и автоматически. В последнем случае надо указать время.

Среди эффектов есть визуальные — возникновение, вращение, вспышка и т. п., а есть звуковые — барабан, аплодисменты и т. п. Впрочем, звук вы можете вставить из любого звукового файла. Для этого в меню *Вставка* надо выбрать пункт *Фильмы и звук*.

Создав слайд, вы можете тут же просмотреть его, воспользовавшись кнопкой *Просмотр*.

Перейдем к третьему этапу. Прежде всего укажем, что в PowerPoint предусмотрено 6 способов отображения созданной презентации (табл. 3.2). Переходить от одного способа отображения к другому можно с помощью кнопок, расположенных в нижней части окна PowerPoint, или выбирая соответствующую команду в меню *Вид*.

Режим *Сортировщик слайдов* позволяет целиком увидеть содержание презентации в миниатюре. Каждый слайд можно снабдить дополнительными эффектами, например указать, каким образом слайд будет появляться на экране («наплыв вверх», «выцветание через черное» и т. п.). В режиме *Сортировщик слайдов* удобно перемещать, удалять, копировать и вставлять новые слайды.

Таблица 3.2

Режим отображения	Назначение режима
Обычный	Именно в этом режиме создаются файлы. Он содержит панели Структура, Слайд, Заметка.
Структура	Отображаются только заголовки и маркированные списки слайдов, что дает возможность просматривать и редактировать сценарий презентации. Здесь нельзя увидеть рисунки и другие «украшения».
Слайд	Панель слайда занимает весь экран презентации.
Заметки	Каждый слайд на экране сопровождается пустой страницей ниже основного кадра. Здесь можно разместить комментарий к данной презентации.
Сортировщик слайдов	Организовывает порядок следования слайдов.
Показ слайдов	Показ слайдов в том виде, в каком они будут демонстрироваться на презентации.

Вопросы и задания

1. В чем преимущество мультимедийных продуктов перед другими видами информации?
2. Каково назначение инструментального средства PowerPoint?
3. Для каких целей может создаваться презентация?
4. Каковы основные этапы разработки электронной презентации?
5. Выполнив перечисленные ниже задания, подготовьте создание презентации художественного произведения, которое вы недавно прочитали.

   а) Выберите произведение, презентацию которого вы намерены сделать. Продумайте, какая информация должна быть сообщена в первую очередь. Составьте план, как эта информация будет размещена на первом слайде.

   б) Продумайте, как вы можете представить главных героев произведения, в каком порядке они будут появляться в вашей презентации. Составьте план, как должен выглядеть соответствующий слайд (или последовательность слайдов), какие эффекты анимации здесь были бы полезны.

   в) Продумайте, какими событиями могут быть представлены основные сюжетные линии, как обрисовать взаимоотношения героев. Если вам нужны графические образы, продумайте, как они могут быть созданы.

   г) Продумайте, как выразить ваше отношение к произведению.
6. Подготовьте создание рекламы какого-либо любимого вами продукта, например мороженого.
7. Выполнение этого задания предполагает распределение работы между учениками класса. Вам предлагается создать электронное учебное пособие для первоклассников «Живая азбука», и каждый участник этого проекта создает несколько слайдов.

   а) Каждой букве алфавита отводится один слайд. Продумайте, какие слова и какие соответствующие им изображения животных должны иллюстрировать употребление данной буквы.

   б) Продумайте структуру каждого из слайдов — она должна быть единой. Определите, как будет осуществляться навигация. Какие требования в связи с этим должны быть объявлены разработчикам слайдов?

   в) Придумайте, как можно организовать проверку того, как первоклассник — пользователь вашего продукта — выучил употребление данной буквы. К примеру, можно предъявлять букву и набор изображений предметов, из которых он должен выбрать те, в названиях которых встречается данная буква. Более сложный вариант — предъявляется набор изображений предметов, в названиях которых встречается одна и та же буква; обучаемый должен указать эту букву.

4 ГЛАВА Телекоммуникационные сети. Интернет

Бурное развитие Интернета является самым значительным и волнующим событием в компьютерном мире после экспансии персональных компьютеров в начале 80-х гг. XX столетия. И событие это вовсе не ограничено рамками компьютерного мира — воздействие Интернета на общество намного шире. Как в Средние века изобретение Иоганном Гутенбергом печатного пресса значительно ускорило перемены в экономике, политике, культуре и общественных отношениях, так сегодня Интернет оказывает огромное воздействие на все стороны человеческого сообщества. Американский исследователь Кевин Келли в своей работе «Новые правила для новой экономики» прямо утверждает, что старая индустриально-иерархическая социальная система Адама Смита на наших глазах превращается в совершенно новую, но вполне реальную и очень интересную систему общественного устройства. Эту социальную систему будущего называют информационно-сетевой или общинно-сетевой.

§ 37 Локальная компьютерная сеть

Загляните сегодня в любой крупный магазин и посмотрите, как кассиры используют базы данных магазина и банка, считывая штрих-коды товаров и проводя расчеты с покупателями по электронным кредитным карточкам. Чтобы это было возможно, все мини-компьютеры кассовых аппаратов должны быть соединены в одну сеть. Впрочем, сегодня ни одна фирма даже средних размеров не может обойтись без сетевого объединения всех своих компьютеров.

Если компьютеры, между которыми надо организовать обмен документами, находятся недалеко друг от друга, например в пределах одного класса или здания, их можно соединить, воспользовавшись либо **коаксильным кабелем**, похожим на телевизионный, либо специальным проводом, называемым **витая пара** (похож на телефонный), либо **оптоволоконным кабелем**. Скорость передачи по такому соединению составляет обычно от 10 до 100 Мбит/с. Самый низкий показатель имеет коаксильный кабель, а самый высокий — оптоволокно. Потребуются еще и специальные **сетевые платы**

(называемые по-другому **сетевыми адаптерами**), и соответствующее программное обеспечение. Такое соединение называется **локальной компьютерной сетью.**

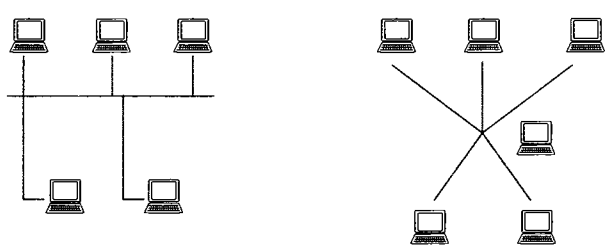
Основная цель создания локальной компьютерной сети — совместное использование информационных ресурсов и ресурсов компьютерной техники: жестких дисков, принтеров, сканеров.

С помощью локальной сети данные, помещенные в памяти одного компьютера, становятся доступными и на других компьютерах. Может ли работать, скажем, банк без компьютерной сети? Конечно, нет! Ведь сведения об операциях с клиентами банка, проводимых в разных его подразделениях, сразу же должны быть отражены в единой базе данных, иначе возникнет полная неразбериха.

В небольших по числу компьютеров локальных сетях все участники равноправны, и каждый пользователь самостоятельно решает, какие ресурсы своего компьютера сделать доступными для сетевого использования. Такие сети принято называть одноранговыми.

Однако, если компьютеров в сети уже больше 10, одноранговая сеть, как правило, оказывается неудобной и малоэффективной. Для увеличения производительности один компьютер, а иногда и больше выделяется для хранения файлов и программных приложений общего доступа всех клиентов сети. Такой компьютер называют **сервером**, и на нем размещают программное обеспечение, позволяющее работать в локальной сети. Нередко этот компьютер обладает значительно большими ресурсами памяти и большей производительностью, нежели рядовые компьютеры сети. На том же сервере могут исполняться различные приложения, требующие использования ресурсов, которых недостаточно на остальных компьютерах, подключенных к данному серверу.

Общая схема соединения компьютеров в сеть называется **топологией сети**. Топология сети может быть весьма различной. Последовательное соединение компьютеров (рис. 4.1, *а*) называется сетью типа «линейная шина». Если же к каждому компьютеру подведен



а) б)

Рис. 4.1 Топология локальной сети

отдельный кабель, соединяющий его с неким центральным узлом, то говорят, что реализована сеть типа «звезда» (рис. 4.1, *б*).

Преимущество локальной сети типа «звезда» в том, что при выходе из строя одного кабеля остальные компьютеры останутся действующими в сети.

Топология сети может быть намного сложнее, чем два представленных здесь случая. Например, она может одновременно использовать указанные варианты в разных своих частях.

Вопросы и задания

1. Для чего создаются локальные компьютерные сети?
2. Что такое топология компьютерной сети? Приведите примеры разных топологий сети.
3. Какую топологию локальной сети целесообразно, на ваш взгляд, использовать в компьютерном классе?
4. Что такое сервер? Для чего используют серверы в локальной сети?
5. Какие типы кабелей обычно используются при создании локальной компьютерной сети? У какого из них наивысшая скорость передачи информации?

§ 38 Глобальные компьютерные сети

В предыдущем параграфе мы уже упомянули про магазин, в кассах которого идет обслуживание по банковским картам. Магазинный компьютер с банковским в локальную сеть не объединишь. Выход состоит в том, что эти компьютеры подключаются к глобальным компьютерным сетям. **Глобальная компьютерная сеть** — это совокупность мощных круглосуточно работающих компьютеров, соединенных посредством той или иной системы связи, оснащенных программным обеспечением, позволяющим осуществлять обмен информацией между компьютерами, подключенными к этой сети. Современные глобальные компьютерные сети включают в себя сотни миллионов компьютеров. Только российский сегмент Интернета насчитывает более полумиллиона компьютеров. Компьютерные сети могут быть реализованы через телефонные или оптоволоконные кабели, радио- и даже спутниковые каналы связи.

Как ни прискорбно об этом сообщать, но Интернет стал таким же детищем военных технологий, как и сам компьютер. В ответ на запуск советского спутника президент США Д. Эйзенхауэр создал в рамках Министерства обороны США (Department of Defence, DOD) лабораторию DARPA. Первая буква D в этом сокращении,

как и последняя в аббревиатуре DOD, означает «оборона», т. е. лаборатория была военная.

Итак, одной из важных дат в истории Интернета можно считать 1957 г. В 60-х гг. основные работы DARPA были как раз посвящены разработке метода соединений компьютеров друг с другом. Очень важно, что возглавлял первую исследовательскую программу, посвященную системе глобальной коммуникации, начатую DARPA 4 октября 1962 г., Дж. Ликлайдер, который опубликовал работу «Galactic Network». В ней он предсказывал возможность существования в будущем глобальной компьютерной связи между людьми, имеющими мгновенный доступ к программам и базам данных из любой точки земного шара. Его предвидение отражает современное устройство международной сети Интернет. Ликлайдер сумел убедить в реальности своей концепции группу ученых, среди которых был будущий его преемник — исследователь Массачусетского технологического института (MIT) Лоренс Робертс.

К концу 1969 г. в одну компьютерную сеть были объединены четыре исследовательских центра. Эта сеть получила название ARPANET. Буква D, как видите, исчезла.

DARPA, вдохновленная успехом создания ARPANET, приступила к разработке новой программы Internetting Project с целью изучения методов соединения различных сетей между собой.

Выдвигались следующие требования:

- универсальность концепции, не зависящей от внутреннего устройства объединяемых сетей и типов аппаратного и программного обеспечения;
- максимальная надежность связи при заведомо низком качестве коммуникаций, средств связи и оборудования;
- возможность передачи больших объемов информации.

Архитектура и принципы сети ARPANET не удовлетворяли выдвинутым требованиям, поэтому была поставлена задача разработки универсального протокола информационного обмена. **Протокол информационного обмена** — это набор правил, определяющий принципы обмена данными между различными компьютерными программами.

В октябре 1972 г. была проведена большая и весьма успешная демонстрация ARPANET на Международной конференции по компьютерным коммуникациям (International Computer Communication Conference, ICCC). Это был первый показ на публике новой сетевой технологии. Также в 1972 г. появилось первое «горячее» приложение — электронная почта.

В марте 1973 г. Рэй Томлинсон, движимый необходимостью создания для разработчиков ARPANET простых средств координации, написал базовые программы пересылки и чтения электронных сообщений. В июле Робертс добавил к этим программам возмож-

ность выдачи списка сообщений, выборочного чтения, сохранения в файле, пересылки и подготовки ответа. С тех пор более чем на 10 лет электронная почта стала крупнейшим сетевым приложением.

Наконец, в 1974 г. Internet Network Working Group (INWG), созданная DARPA и руководимая Винтоном Серфом, разработала универсальный протокол передачи данных и объединения сетей **Transmission Control Protocol/Internet Protocol (TCP/IP)** — сердце Интернета.

В 1980 г. INWG под руководством Винтона Серфа объявила TCP/IP стандартом и представила план объединения существующих сетей, сформулировав основные его принципы:

- сети взаимодействуют между собой по протоколу TCP/IP;
- объединение сетей производится через специальные «шлюзы» (gateway);
- все подключаемые компьютеры используют единые методы адресации.

В 1983 г. DARPA обязала использовать на всех компьютерах ARPANET протокол TCP/IP, на базе которого Министерство обороны США разделило сеть на две части: отдельно для военных целей — MILNET и научных исследований — сеть ARPANET.

Для объединения имеющихся шести крупных компьютерных центров и поддержания глобального академического и исследовательского сообществ в 1985 г. Национальный Научный Фонд США (National Science Foundation, NSF) начал разработку программы построения межрегиональной сети NSFNET.

Но параллельно с военными и академическими исследованиями велось создание коммерческих компьютерных сетей. К разработке коммерческих стандартов локальных сетей одной из первых приступила фирма Xerox, учредив консорциум Ethernet, в который вошли также фирмы Intel и Dec. В 1980 г. консорциум выпустил документацию на сеть Ethernet. Локальные сети с успехом начали использоваться в самых различных учреждениях и компаниях.

Локальные сети буквально произвели революцию, позволив существенно увеличить не только производительность труда конторских служащих, но и качество управления в целом. Но они явно были недостаточными для крупных корпораций, в первую очередь нефтяных, имеющих отделения в разных городах и даже странах. Поэтому вполне естественным стал их интерес к разработкам DARPA и подключению своих локальных сетей фактически к общенациональной сети NSFNET. Такое подключение могло быть произведено, очевидно, только на основе TCP/IP протокола.

В сентябре 1988 г. начала работу коммерческая выставка совместимых между собой продуктов, разработанных на основе TCP/IP. Эту дату можно считать началом новой жизни глобальной сети, подключение к которой стало доступным для любого желаю-

щего и зависело только от стремительно уменьшающихся тарифов за такое подключение. Это привело ко второму взрывообразному росту сети Интернет, в результате изобретения технологии World Wide Web, речь о которой будет идти ниже. К этому моменту темпы роста сети Интернет показали, что регулирование вопросов подключения и финансирования не может находиться в руках одного NSF. В этом же году произошла передача региональным сетям права взимать оплату за подсоединение многочисленных частных сетей к национальной магистрали, ставшей к тому времени уже «наднациональной».

Революционизирующее влияние Интернета на мир компьютеров и коммуникаций не имеет исторических аналогов. Изобретение телеграфа, телефона, радио и компьютера подготовило почву для происходящей ныне беспрецедентной интеграции. Интернет одновременно является и средством общемирового вещания, и механизмом распространения информации, и средой для сотрудничества и общения людей и компьютеров, охватывающей весь земной шар.

Но Интернет так и остался бы компьютерной сетью для специалистов, если бы не два поистине революционных изобретения, сделанные в области программного обеспечения.

Первым изобретением стала **электронная почта**, вторым — **Всемирная информационная паутина**. Они, во-первых, сделали Интернет предметом первой необходимости для огромного числа людей, а во-вторых, обеспечили необыкновенную легкость работы с сетевым программным обеспечением.

История WWW (World Wide Web, WWW) началась в марте 1989 г., когда Тим Бернерс-Ли из Европейской Лаборатории физики элементарных частиц предложил новый способ обмена результатами исследований и идеями между участниками коллектива исследователей-физиков, работавших в разных странах. Для передачи документов и установления связи предлагалось использовать просто систему гипертекста (тогда никто еще не задумывался о возможности передачи фотографических изображений, звука или видео; речь шла только о распространении текстовых документов, содержащих гиперссылки на фрагменты других таких же текстовых документов, но располагающихся на удаленных компьютерах, подключенных к глобальной сети Интернет). Сегодня World Wide Web — это совокупность огромного числа информационных материалов, связанных между собой гипертекстовыми ссылками.

Вот основные понятия, относящиеся к Всемирной информационной паутине:

WWW — множество веб-страниц, размещенных на узлах Интернета и связанных между собой гиперссылками.

Web-страница — структурная единица WWW, которая включает в себя собственно информацию (текстовую и графическую) и ссылки на другие страницы или ресурсы Интернета.

Систему гипертекстовых web-страниц, контролируемых одним человеком или одной организацией, обычно называют web-**сайтом** или просто **сайтом** (от английского site — местоположение, участок под застройку).

Программа, пересылающая web-страницы на компьютер клиента с использованием протокола **http** (hyper text transfer protocol — протокол передачи гипертекста), называется **web-сервером**. На одном компьютере могут работать несколько web-серверов, и, наоборот, один web-сервер может использовать несколько компьютеров.

В настоящее время Интернет превратился в единое информационное поле планетарных масштабов. И его воздействие на развитие цивилизации грандиозно, хотя до сих пор и не осознано в полной мере. Число пользователей составляет, по различным оценкам, от 300 до 500 млн человек, из них более 5 млн в России.

Связь компьютеров с глобальной сетью Интернет, о которой говорилось выше, очень часто осуществляется с помощью телефонной сети. Но в телефонной сети передается непрерывный, или, по-другому, аналоговый, сигнал, а в компьютере вся информация кодируется дискретно, поэтому необходимо специальное устройство, преобразующее цифровой сигнал компьютера в аналоговый и обратно. Преобразование цифрового сигнала в аналоговый называется **модуляцией**, обратное преобразование — **демодуляцией**. От сокращения этих слов произошло название устройства, осуществляющего модуляцию и демодуляцию. Такое устройство называют **модемом**. Современные модемы, например, типа ADSL способны передавать данные со скоростью более 1 Мбит/с. Кроме того, они не препятствуют осуществлению параллельно обычной телефонной связи. А ведь еще 10 лет назад предельная скорость передачи по телефонным сетям составляла всего лишь 56 Кбод.

Во все современные операционные системы встроено все, что необходимо для начала работы в Интернете, включая мастера для установки и настройки программного обеспечения. Но для доступа к Интернету необходимо подключение в какой-либо форме к поставщику услуг Интернета, так называемому **провайдеру**. Это можно осуществить двумя способами:

- непосредственно обратившись к поставщику услуг Интернета;
- подключившись к локальной сети, имеющей, в свою очередь, постоянное подключение к Интернету.

В первом случае снова возможны два варианта: постоянное подключение по специально выделенному каналу связи и сеансовое подключение по обычному телефонному каналу с использованием модема. В настоящее время большинство индивидуальных пользователей и небольших фирм прибегают именно ко второму способу. Для этого достаточно зарегистрироваться у провайдера, получив учетную запись и пароль, с помощью которых будет происходить соединение с сервером. Как правило, провайдер предо-

ставляет и бесплатный электронный почтовый ящик.

Временные учетные записи и пароли содержатся и в так называемых карточках доступа к сети Интернет, продающихся в киосках. Каждая такая карточка обеспечивает строго определенное (в зависимости от тарифного плана компании) время пребывания в Интернете. Как правило, дневное время существенно дороже вечернего и ночного.

Что касается подключения к локальной сети, имеющей постоянный выход в Интернет, то в настоящее время такие локальные сети могут объединять микрорайон, отдельный дом или даже подъезд.

Вопросы и задания

1. Что такое глобальная компьютерная сеть?
2. Что такое протокол информационного обмена?
3. Что такое Интернет?
4. Что называют Всемирной паутиной?
5. Для чего предназначен протокол http?
6. Что такое модуляция сигнала и демодуляция?
7. Для чего нужен модем?
8. Какова роль провайдера?
9. Какие существуют варианты подключения индивидуального пользователя к Интернету?
10. Если ваш компьютерный класс подключен к Интернету, выясните, какой из вариантов для этого используется.

§ 39 Адресация в Интернете

Как уже говорилось, TCP/IP-протокол был создан под руководством В. Серфа группой Internet Network Working Group (INWG). Он предусматривал пакетный обмен информацией между компьютерами, входящими в сеть. И перед разработчиками встала проблема **маршрутизации** пакетов, чтобы обеспечить передачу данных на нужный компьютер. Для этого было принято решение дать каждому компьютеру уникальный четырехбайтовый номер. Его называют **IP-адресом**. Обычно он записывается в виде четырех чисел, представляющих значения каждого байта в десятичной форме и разделенных точками, например 128.10.2.30 — традиционная

точечно-десятичная форма представления адреса. Каждое из этих четырех чисел заключено в пределах от 0 до 255.

Система IP-адресации учитывает то обстоятельство, что Интернет — это совокупность сетей, у каждой из которых свой размах. Для обеспечения гибкости в распределении IP-адресов в зависимости от количества компьютеров в сети адреса разбиты на три класса — A, B и C. Первые биты адреса указывают класс, а остальные отведены под адрес сети и адрес компьютера в этой сети. В таблице 4.1 показано распределение битов в адресах разных классов.

Из таблицы видно, что для сетей класса A имеется всего лишь $2^7 = 128$ различных адресов (значит, и сетей такого класса не больше чем 128), зато компьютеров в каждой такой сети может быть $2^{24} = 16\,777\,26$. А вот сетей класса C уже может быть $2^{21} = 2\,097\,152$, но компьютеров в каждой из них не более 256. В таких сетях нередко используют динамическое присвоение IP-адресов — компьютеру присваивается такой адрес непосредственно при подключении его к сети, и он действует только в течение сеанса связи.

Числовая адресация удобна для машинной обработки таблиц маршрутов. Однако то, что удобно машинам, неудобно людям. Есть спорное мнение, что сама человеческая натура протестует против запоминания чисел типа 192.168.1.34 (что тем не менее не мешает нам запоминать телефонные номера типа 8-97021-56437). К тому же IP-адреса совсем не информативны. По IP-адресу невозможно понять, что это: сервер, ПК, маршрутизатор, сетевой принтер или, скажем, холодильник. Приятней работать с осмысленными именами, такими, как 1september.ru. И уж конечно осмысленные имена незаменимы для рекламирования компании.

Поэтому в сети сначала стали использовать таблицы соответствия числовых адресов именам машин. Эти таблицы сохранились до сих пор и используются многими прикладными программами. Чисто физически они располагаются в самых обычных текстовых файлах. Структура таблицы соответствия приведена в таблице 4.2.

В стародавние времена, когда Интернет еще назывался ARPANET и Сеть состояла лишь из небольшого количества мощных многотерминальных компьютеров, каждый компьютер имел

Таблица 4.1

Класс	Распределение битов в IP-адресах																															
A	0	Адрес сети (7 бит)							Адрес компьютера в сети (24 бита)																									
B	1	0	Адрес сети (14 бит)														Адрес компьютера в сети (16 бит)																
C	1	1	0	Адрес сети (21 бит)																							Адрес компьютера в сети (8 бит)						

такой файл (обычно /etc/hosts). При появлении в Интернете нового компьютера информация о нем заносилась в файл hosts, затем этот файл рассылался на все другие машины.

Таблица 4.2

IP-адрес	Имя машины
127.0.0.1	Localhost
192.204.461.32	Polyn
144.206.160.40	Apollo

Недостатки такой схемы начали проявляться довольно быстро: с переходом от больших машин к персональным и с ростом Интернета трафик, связанный с обновлением информации при добавлении компьютеров в Интернет, становился до неприличия большим. Кроме того, каждое имя в сети должно быть уникальным, а сделать это становилось все труднее и труднее. Поэтому к середине 80-х гг. появилась другая, более гибкая система именования — **система доменных имен (Domain NameSystem, DNS).**

DNS — это база данных, которая содержит информацию о компьютерах, включенных в сеть Интернет. Характер данных зависит от конкретной машины, но чаще всего информация включает имя машины, IP-адрес и данные для маршрутизации почты. Доменная система имен выполняет несколько задач, но основная ее работа — преобразование имен компьютеров в IP-адреса и наоборот.

В DNS вся сеть представляется в виде единого иерархического дерева. На вершине располагается корневой домен (обозначается символом «.»). Ниже находятся домены первого уровня. Поскольку Интернет развивался в первую очередь в США и за счет американских налогоплательщиков, это вызвало некоторый крен при формировании доменов первого уровня: Интернет как бы оказался поделенным между США и всем остальным миром.

Наиболее известные домены первого уровня:
com — коммерческие организации (главным образом, в США);
edu — учебные заведения США;
gov — правительственные учреждения США;
mil — военные учреждения США;
net — различные сетевые агентства и Internet-провайдеры;
int — международные организации;
org — некоммерческие учреждения;
info — средства массовой информации.

Позднее, когда сеть перешагнула национальные границы США, появились национальные домены типа uk, jp, au, ch и т. п. Для СССР был выделен домен su. После 1991 г., когда республики союза стали суверенными, многие из них получили свои собственные домены: ru, by, ua, la, li и т. п. Однако Интернет не СССР, и про-

сто так выбросить домен su из сервера имен нельзя, на основе доменных имен строятся адреса электронной почты и доступ ко многим другим информационным ресурсам Интернета. Поэтому гораздо проще оказалось ввести новый домен к существующему, чем заменить его. Таким образом, в сети существуют организации с доменными именами, оканчивающимися на su. Мало того, некоторое время назад было принято решение возобновить регистрацию в этой области.

Доменов верхнего уровня очень немного — всего около 250. Бóльшая часть из них — географические домены. Если бы вам захотелось зарегистрировать еще один домен верхнего уровня, то потребовалось бы для этого предоставить такие серьезные обоснования, что гораздо проще было бы организовать свое маленькое государство и для него уже получить географический домен.

Вслед за доменами верхнего уровня следуют домены, определяющие либо регионы (msk), либо организации (1september). Далее идут следующие уровни иерархии, которые могут быть закреплены либо за небольшими организациями, либо за подразделениями больших организаций.

Иерархия доменных имен строится следующим образом: дерево начинается с корня, обозначаемого точкой. Затем следует старшая символьная часть имени, затем опять точка и вторая по старшинству символьная часть имени и т. д. Младшая часть имени всегда соответствует конечному узлу сети. Запись доменного имени начинается с самой младшей составляющей, а заканчивается самой старшей. Составные части доменного имени всегда отделяются друг от друга точкой. Например, в имени w2000.microsoft.com составляющая w2000 является именем одного из компьютеров в домене microsoft.com.

Разделение административной ответственности позволяет решить проблему образования уникальных имен без взаимных консультаций между организациями, отвечающими за имена одного уровня иерархии. Очевидно, что должна существовать одна организация, отвечающая за назначение имен верхнего уровня иерархии. В Интернете корневой домен управляется центром ICANN, в состав которого входит по пять представителей от каждого континента.

Процедура получения имени, например, в зоне .ru или .com называется регистрацией домена. Конечно, каждая компания, подключающаяся к Интернету, стремится зарегистрировать как можно более естественное и легкое для запоминания имя. Так, для компании Microsoft логично зарезервировать домен microsoft.com.

Каждый домен администрируется отдельной организацией, которая обычно разбивает свой домен на поддомены и передает функции администрирования этих поддоменов другим организациям. Чтобы получить доменное имя, необходимо зарегистрироваться в какой-либо организации, которой ICANN делегировал свои полномочия по распределению имен доменов. В России такой органи-

зацией является NIC.ru, которая отвечает за делегирование имен поддоменов в домене ru.

Наконец, имеется еще одно важное для Интернета понятие — URL (Uniform Resource Locator) — **универсальный указатель ресурса**. Он представляет собой точное описание ресурса, включающее его местонахождение в Интернете, и имеет следующую структуру:

service://host:port/path/file.ext

(вид сервиса://имя узла:номер порта/путь/имя файла. расширение).

Например:

http://www.usu.ru/...

Сервис http означает, что мы намерены иметь дело со Всемирной паутиной; www.usu.ru — это имя того узлового компьютера, с информационными ресурсами которого мы собираемся работать; в данном случае речь идет об узловом компьютере сети WWW, обслуживающем Уральский государственный университет.

Вопросы и задания

1. Что такое IP-адрес? Для чего он предназначен?
2. Объясните, почему в IP-адресе каждое из фигурирующих в нем четырех чисел заключено в диапазоне от 0 до 255.
3. Для каждого из приведенных ниже IP-адресов определите, к какому классу относится сеть, в которой используется данный адрес:
   а) 23.123.0.78; б) 144.1.245.13; в) 195.42.12.3.
4. * Может ли IP-адрес начинаться на 249?
5. Для каждого из классов A, B и C укажите диапазон, в котором может находиться первое из четырех чисел, записанное в IP-адресе любого компьютера, входящего в сеть данного класса.
6. Сколько сетей класса B может существовать? Какое максимальное число компьютеров может быть в сети этого класса?
7. Каждый ли компьютер, который имеет выход в глобальную сеть, обладает IP-адресом?
8. Для чего служит доменное имя?
9. Как устроен универсальный указатель ресурса в Интернете? Для чего он предназначен?

§ 40 Поисковые системы Интернета

В § 38 мы уже говорили, что Интернет сегодня — это почти бездонное хранилище самых разнообразных информационных объектов: текстовых, графических, звуковых, видео и т. п. Их в тысячи, а может быть, и в миллионы раз больше, чем в любой доступной вам библиотеке. Но даже в библиотеке не всегда легко найти нужную информацию. Как же осуществить это в Интернете?

Началось все в апреле 1994 г., когда Дэвид Фило и Джерри Янг из Стэндфордского университета пришли к выводу, что у каждого из них накопилось такое большое количество ссылок на самые разные информационные источники, что пора каким-то образом их упорядочить. Так родилась идея создания специализированной базы данных, которая вскоре стала использоваться тысячами пользователей для эффективного поиска информации в сети. Называется эта поисковая система Yahoo!, что расшифровывается как Yet Another Hierarchical Officious Oracle (Еще Один Иерархический Официозный Оракул). В начале следующего, 1995 г. детище Фило и Янга поместили на более мощные компьютеры Netscape, и вскоре система Yahoo! стала самым популярным и полным иерархическим предметно-ориентированным путеводителем по Всемирной информационной паутине и Интернету в целом. В Yahoo! все рассортировано по темам и категориям, вся информация разложена по полочкам так, что даже самому ленивому пользователю не составит большого труда найти искомый ресурс в необозримом пространстве сети.

Наряду с Yahoo! существуют и другие подобные системы, в том числе и русскоязычные. Здесь мы упомянем только о четырех таких системах, наиболее популярных в нашей стране.

Одним из самых известных является поисковый сервер Яндекс. Его история восходит к 1990 г., когда в компании «Аркадия», возглавляемой Аркадием Борковским и Аркадием Воложем, начались разработки поискового программного обеспечения. Сайт Яndex появился в Интернете в 1996 г., после того как руководством фирмы CompTek и разработчиками системы было принято решение о дальнейшем развитии поисковых технологий, ориентированных на широкую аудиторию пользователей Интернета. Официально поисковая машина Yandex.Ru была анонсирована 23 сентября 1997 г. на выставке Softool.

Через два месяца, в ноябре 1997 г., был реализован естественно-языковой запрос. Отныне к Yandex.Ru можно обращаться просто «по-русски», задавать длинные запросы, например: «где купить компьютер», «генетически модифицированные продукты» или «коды международной телефонной связи», и получать ответы. Средняя длина запроса в Yandex.Ru сейчас 2,7 слова. В 1997 г. она составляла 1,2 слова — тогда пользователи поисковых машин были приучены к телеграфному стилю.

Поисковая система Апорт! (http://www.aport.ru) начала работу с лета 1997 г. Одно из главных ее преимуществ — удачные средства составления запроса. Кроме того, эта система предлагает возможность автоматического перевода запросов с русского на английский язык и наоборот. Найденные документы упорядочиваются в зависимости от частоты употребления в них искомых терминов, глубины их расположения в тексте.

Интересно, что большинство поисковых систем делают возможным разделение задач по нескольким мощным компьютерам, что и обеспечивает очень высокую скорость обработки запросов.

Сравнительно недавно появилась и стала уже одной из самых популярных поисковая система Google (http://www.google.ru и http://www.google.com). При обращении к ней появляется ее первая страница (рис. 4.2).

На первой странице любой поисковой системы имеется специальное окно, в которое вы записываете запрос на поиск. Сам запрос состоит из **ключевых слов** — они выражают то, что составляет суть разыскиваемой информации. Пусть, к примеру, вам для

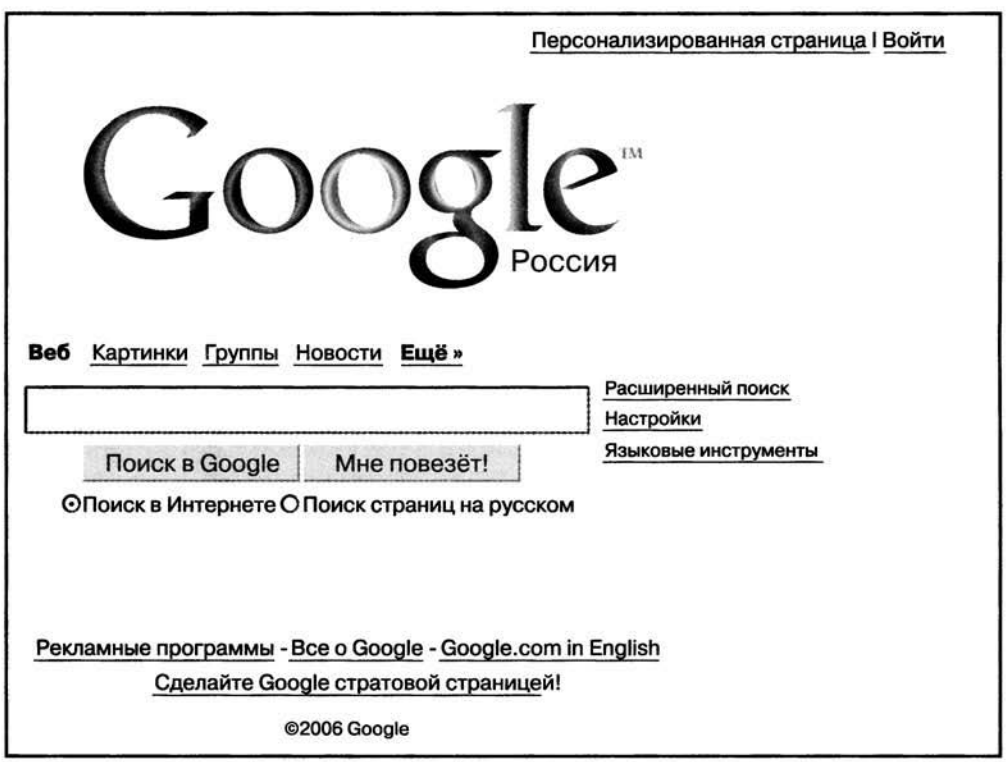


Рис. 4.2 Поисковая система Google

подготовки реферата по истории Интернета нужно найти фотографию В. Серфа. Тогда в строке поиска надо набрать слово «Серф». А еще лучше набрать два слова: «Винтон Серф». Пробел между словами большинство поисковых систем воспринимает как союз **и**. Если вы попробуете так поступить, то количество найденных документов будет составлять несколько тысяч. Среди них и статьи самого В. Серфа, и статьи о нем, и книги, в которых упоминается его имя (в частности, и этот учебник). Это вовсе не значит, что поисковая система сработала плохо, это вы неудачно составили запрос. Вы же хотите фотографию, но ничего об этом поисковой системе не сказали. Составив запрос: «Винтон Серф фото», вы резко увеличите шансы на успех. Правила составления запросов с союзом **или** весьма сильно зависят от поисковой системы; вы можете их посмотреть, обратившись к режиму *Помощь*.

Если вы захотите найти документы, содержащие слова «Люблю грозу в начале мая», и введете их в поле запроса, многие поисковые системы включат в результаты поиска и документы, содержащие, допустим, слова «люблю грозу в мае», «Маша любит грозу» и т.д. Поиск, при котором допускаются словоизменения, называется **морфологическим**. Его способны осуществлять все русскоязычные и многие зарубежные поисковые системы. Когда в поле запроса введены слова «Люблю грозу в начале мая», мы, скорее всего, хотим найти документы, содержащие все четыре слова. Однако поисковая система, представив вначале документы со всеми словами, начнет затем давать ссылки на документы, в которых есть хотя бы одно из указанных слов. Чтобы указать поисковой системе, как должны быть связаны между собой введенные пользователем ключевые слова, используются специальные операторы.

Выше мы уже упоминали о логических операторах. Они позволяют указать поисковой системе, как она должна отбирать документы, если в запросе перечислено несколько ключевых слов. Логическое И между ключевыми словами показывает, что необходимо выдать документы, содержащие все искомые слова. Логическое ИЛИ используется при необходимости поиска документов, содержащих хотя бы одно из перечисленных ключевых слов. Логическое НЕ позволяет исключить из множества документов те, в которых присутствует слово, следующее за этим оператором.

Другая часто используемая группа операторов — операторы расстояния. Они позволяют задать ограничения на удаленность вхождений ключевых слов друг от друга в тексте документа. Выделяют три оператора этой группы: оператор поиска фразы, позволяющий находить точные вхождения указанной последовательности слов; оператор, позволяющий задать расстояние в словах; оператор, позволяющий задать расстояние в предложениях. Как именно записываются операторы в запросах, определяется правилами той системы, которой вы намерены воспользоваться. Поэтому мы настоятельно советуем: прежде чем начать работать в поисковой

системе, надо познакомиться с правилами составления запросов в этой системе — это сэкономит ваши силы и время.

Каждая поисковая система представляет собой комплект программ, в основе которого лежат следующие пять программ:

• **Spider** («паук») — программа, которая загружает в поисковую машину Web-страницы. Она работает аналогично браузеру, установленному на компьютере пользователя, но ничего не отображает ни на каком экране.

• **Crawler** («червяк», или «путешествующий паук») — программа, способная найти на Web-странице все ссылки на другие страницы. Задача этой программы — определить, куда дальше должен идти «паук», руководствуясь ссылками или заранее заданным списком адресов.

• **Indexer** (индексатор) — программа, которая «разбирает» страницу на составные части и анализирует их. При этом вычленяются и анализируются заголовки Web-страниц, заголовки документов, ссылки, текст документов, отдельно анализируется текст, выделенный полужирным шрифтом, курсивом и т.п. Результаты анализа поступают на хранение в специальную базу данных (см. следующий пункт). Глубина индексации может быть разной. В частности, полные тексты документов, размещенных на Web-странице, в базу данных копируют далеко не все поисковые системы — некоторые ограничиваются лишь заголовками.

• **Database** (база данных) — хранилище всех данных, которые поисковая система загружает и анализирует. Она требует огромных ресурсов как для хранения, так и для последующей обработки.

• **Search Engine Results Engine** (система выдачи результатов поиска) решает, какие страницы удовлетворяют запросу пользователя и в какой степени. Именно с этой частью поисковой системы общается пользователь.

Первые две программы, работающие в связке, часто называют **поисковым роботом**.

На самом деле поисковая система, получив запрос на поиск, не отправляется в длительное путешествие по Всемирной паутине, а анализирует лишь ту информацию, которая была ею проиндексирована. С одной стороны, это позволяет резко повысить скорость обработки запроса на поиск. С другой — область поиска оказывается ограниченной внутренними ресурсами поисковой системы. Надо понимать, что эти ресурсы, во-первых, не полны – ведь ни одна поисковая машина не в состоянии загрузить в свою базу данных информацию со всех узлов Сети, а во-вторых, уже в какой-то степени устарели. Ситуация в Интернете меняется очень быстро. Если «паук» с целью обновления информации об уже когда-то проиндексированных Web-страницах «заползает» на них раз в два месяца, пользователь рискует получить в результатах запроса ложную ссылку.

Большинство поисковых систем осуществляет разделение задач по нескольким мощным компьютерам, что также повышает скорость обработки запросов.

Вопросы и задания

1. Какие средства поиска информации имеются в WWW?
2. Как воспользоваться поисковой системой?
3. Какой поиск называется морфологическим?
4. Какими средствами обладают поисковые системы для более точной формулировки поискового запроса?
5. Каковы основные программные компоненты поисковой системы? Каково назначение каждого из них?
6. Может ли случиться так, что поисковая система найдет документ, который не существует? Если да, то в чем, на ваш взгляд, причина такого результата?

§ 41 Интернет как источник информации

Поиск нужной информации — это не только умение составить запрос по правилам используемой вами поисковой системы. К поиску информации нужно подходить как к решению любой информационной задачи. Об этапах решения информационной задачи мы рассказывали в § 11 учебника для 10 класса.

Напомним их:

- на первом этапе определяется, какую исходную информацию необходимо иметь для решения задачи и как ее получить;
- на втором этапе привлекаются знания, позволяющие связать исходную информацию с результатами и определяющие, как именно получить требуемый результат;
- на третьем этапе исходная информация извлекается и преобразуется в результирующую информацию.

Мы проиллюстрируем прохождение этих этапов на примере решения одной из возможных поисковых информационных задач. Пусть, к примеру, вам потребовалось найти информацию для того, чтобы составить сравнительную хронологическую таблицу жизни А. С. Пушкина и событий русской истории. Какую исходную информацию надо иметь? Во-первых, надо определить основные события в жизни поэта и сопоставить им временные рамки. Во-вторых, определить, что будем понимать под событиями русской истории.

Можно, например, разбить их на внешнеполитические, внутриполитические и культурные. Каждый класс событий определяет свою стратегию поиска, которая подлежит дальнейшему уточнению. Впрочем, вполне возможно, что вы иначе произведете уточнение характера разыскиваемой информации, мы же только иллюстрируем, как это может происходить. Но обратите внимание: это еще не поиск, а только подготовка к нему.

После выполнения первого этапа начинаем выполнять второй. Как выявить основные события в жизни поэта и даты этих событий? Возможно, вы знаете, что есть соответствующий биографический справочник. Более того, вы помните его название и кем он был издан. Это параметры вашего поиска. Поиск по таким точно определенным параметрам называется **адресным**. Впрочем, и в этом случае лучше подстраховаться, задавая себе вопросы типа «Точно ли я знаю название?», «Как еще мог бы называться справочник?», «Правильно ли я указал издательство?» и т. п.

Но обычно провести адресный поиск бывает трудно. Что же тогда выступает в роли параметров поиска? Это уже упоминавшиеся выше **ключевые слова**, которые содержательно характеризуют тему поиска. Применительно к рассматриваемой задаче лучше сформировать несколько наборов ключевых слов, относящихся к биографии А. С. Пушкина, например «детство Пушкина», «лицейские годы Пушкина», «первые стихи Пушкина», «женитьба Пушкина» и т. п. Как вы видите, здесь нам пришлось все-таки воспользоваться определенными знаниями о Пушкине. Если же вы совсем ничего не знаете, то и ключевые слова у вас получатся, скорее всего, только такие: «биография Пушкина».

Поиск по ключевым словам называется **тематическим** — ведь в основу его положено ваше понимание темы поиска. При организации тематического поиска надо иметь в виду, что, чем более распространено данное слово в жизни, тем меньше оно подходит на роль ключевого. Если, к примеру, вы захотите узнать, что такое русская баня, то на запрос по ключевому слову «баня» среди первых двадцати ссылок, выданных поисковой системой, вы обнаружите не менее 8 предложений, где и как купить баню, рекомендации, как правильно париться и какие веники использовать, 6 реклам различных банных комплексов, 2 стихотворения о бане, статью «Как самому построить русскую баню», ссылку на словарь В. Даля (на пословицу о бане) и еще на статью с названием «Кто про что, а вшивый про баню», которая, собственно, к бане никакого отношения не имеет. А вот если вы захотите узнать, что такое «Улигеры ононских хамниган», смело пишите эти слова в запрос и через 14 секунд будете знать, что хамнигане — это одна из бурятских народностей, живущих на реке Онон, а улигеры — это сказания о мифических героях этого народа.

Подбирая ключевые слова, надо заботиться о том, чтобы они максимально подробно отражали смысл вашего поискового запро-

са. Не нужно забывать и о подборе синонимического ряда к ключевым словам. Ведь может оказаться, что поисковая система «не знает» ни одного из указанных вами слов, но «знает» синонимы каких-то из них. Полезно при организации поиска продумать и то, в какой форме вы хотите получить информацию — текстовой, графической, видео- и т. п.

Возможно, что к выполнению второго этапа вам придется возвращаться несколько раз, уточняя поисковый запрос.

Может случиться и так, что вы совсем не знаете, как охарактеризовать тему поиска с помощью ключевых слов. Тогда вам на помощь придут электронные словари и энциклопедии. Создав запрос «Что такое ... ?», вы получите ссылки на статьи в справочных изданиях. Одним из них является **википедия** — открытая электронная энциклопедия, построенная по принципу гипертекста. Из ее статей вы узнаете, с какими еще понятиями связано то понятие, смысл которого вы хотите для себя уяснить, и с помощью гиперссылок сможете познакомиться с этими родственными понятиями. Принцип открытости википедии состоит в том, что, во-первых, она постоянно пополняется новыми сведениями, а во-вторых, постоянно уточняется информация о том, что в ней уже содержится.

Как бы точно вы ни пытались составить запрос, почти наверняка вам будут представлены документы, не удовлетворяющие вашу информационную потребность. Информация, соответствующая информационной потребности, называется **релевантной**. Эффективность поиска обычно оценивают по двум параметрам — **полноте** поиска и **точности** поиска.

Полнота поиска — это отношение числа выданных релевантных документов к общему числу релевантных документов, имеющихся в распоряжении поисковой системы. В идеале это число должно было бы равняться 1. На практике такое почти никогда не достигается (как и положено любому идеалу). В реальных поисковых системах коэффициент полноты поиска может достигать значений 0,7—0,9.

Точность поиска — это отношение числа выданных релевантных документов к общему числу документов, выданных системой по данному запросу. Значение этого параметра колеблется от 0,1 до 1. Как вы понимаете, здесь многое зависит от того, насколько хорошо был составлен запрос пользователем.

Вопросы и задания

1. Чем адресный поиск отличается от тематического?
2. Составьте запросы, с помощью которых, по вашему мнению, можно получить следующую информацию:
   а) Когда появилась IBM PC?

б) Кто является создателем системы Google?
в) Что означает слово «блоггер» и откуда оно, вообще, взялось?
г) Кто и когда изобрел шариковую авторучку?
д) Был ли известный ученый-физиолог И.П. Павлов атеистом?
е) Что изображено на гербе МГУ им. М.В. Ломоносова?

3. Что такое википедия? Почему ее называют открытой энциклопедией?

4. Какая информация называется релевантной?

5. Сформулируйте, какое качество выполненного поиска характеризуется параметром «точность», а какое параметром «полнота».

§ 42 Сервисы интернета

Мы уже довольно подробно обсудили преимущества Интернета как бездонного хранилища информации. Но исторически первым сервисом, предоставлявшимся еще сетью ARPANET, была **электронная почта**. Она позволяет обмениваться текстовыми сообщениями с любым пользователем, имеющим выход в глобальную компьютерную сеть и зарегистрированным на каком-либо почтовом сервере. При регистрации абонент получает **электронный почтовый адрес, имя пользователя для входа в сеть** — так называемый **login** — и **пароль**, позволяющий избежать несанкционированного доступа посторонних лиц к вашей почте.

Адрес в электронной почте образован из доменных имен, которые стоят после символа @. Например: Billt@tenet.edu, или Shaynes@monroe.lib.mi.us, или 2170@dialup.mplik.ru. Слева от значка @, как правило, указывается **идентификатор конечного пользователя**, т. е. присвоенный вам позывной на том сервере, где вы имеете электронный почтовый ящик.

Преимущества электронной почты по сравнению с обычной весьма значительны. Она позволяет:

- посылать сообщения сразу нескольким абонентам;
- содержать не только собственно текст письма, но и файловые вложения с графикой, звуком, программным обеспечением и т. д.;
- пересылать полученные сообщения другим адресатам;
- использовать автоответчик, который будет уведомлять отправившего вам письмо, что оно получено.

Это далеко не все, что умеют делать современные программы почтового сервиса. Одной из таких программ является, например, Outlook Express, которая входит в состав Microsoft Internet Explorer. Впрочем, в последнее время все большее распространение в работе с электронной почтой получают Web-технологии. Появились Web-сайты, которые предлагают всем желающим зарегистрировать бесплатный почтовый ящик.

Надеемся, что вы адекватно оценили Интернет как информационное хранилище и средство коммуникации с любыми уголками нашей планеты. Но информация информации рознь, и ваш интерес к ней тоже может быть весьма различным. Одно дело — просмотреть новости в той или иной газете, и совсем другое — изучить большую научную статью. Для такого изучения необходимо время, и возвращаться к ее тексту, возможно, потребуется не один раз. Гораздо эффективнее получить нужную информацию к себе на компьютер в виде соответствующего файла. Кроме того, Интернет наполнен не только текстовой, звуковой или видеоинформацией, но в нем имеется немало самого разнообразного программного обеспечения, распространяемого как солидными программистскими фирмами, так и просто программистами-любителями. Получить файлы по сети позволяет **ftp-сервис** (от **F**ile **T**ransfer **P**rotocol — протокол передачи файлов). Именно он предоставляет доступ к файловым системам и выполняет передачу файлов в сеть (например, при публикации собственного Web-сайта). Одним из распространенных видов ftp-серверов является анонимный ftp-сервер. На таком сервере вы используете в качестве своего имени слово anonymous, а в качестве пароля свой e-mail. Если же вы хотите установить соединение с компьютером, не предоставляющим анонимного сервиса, вы должны иметь право доступа к системе, т. е. иметь собственное имя пользователя и пароль.

Для работы с ftp-серверами можно, как обычно, воспользоваться браузером. Только в начале адреса надо написать ftp://, а затем URL нужного вам ресурса. Удобнее, однако, для работы с большими по объему файлами использовать специализированные программы, которые называются менеджерами загрузки файлов (например, FlashGet, Go!Zilla и др.). Они разбивают файловый архив на части и загружают эти части одновременно, что уменьшает время перекачивания файла на ваш компьютер, позволяют продолжить загрузку файла после обрыва соединения с сервером и выполняют ряд других работ, облегчающих жизнь пользователю.

Но польза от ftp-сервиса была бы не так значительна, если бы для получения информации пришлось бы просматривать все имеющиеся в сети анонимные серверы. Интернет и здесь идет навстречу пользователю. Для этого существуют ИПС, с помощью которых можно проводить поиск информации, расположенной на анонимных ftp-серверах. Системы, поддерживающие этот вид услуг, регулярно собирают с таких серверов информацию о содержащихся на них файлах. К таким системам относится, например, ИПС Archie.

Перестроив на электронный манер почтовую связь, разработчики глобальных сетей не успокоились. Получать письма приятно, но еще приятнее диалог, когда на каждый твой вопрос или суждение тут же реагирует собеседник. Простейший вариант такого общения — письменное сообщение: ты строчку, тебе в ответ строчку, снова ты и т. д. Есть сервис Интернета, который позволяет таким

образом общаться на различные темы. Сокращенно его называют **IRC-сервис** (от Internet Relay Chat; напомним, что chat в переводе означает «разговор», «беседа», «болтовня»). Он используется для общения. Если к компьютеру подключен микрофон и наушники, то общение станет не только письменным, но и звуковым. А если еще подключена и веб-камера, то ваш собеседник увидит вас на экране своего компьютера.

Для организации такого общения необходима специальная программа. Ею может быть программа NetMeeting, входящая в состав Internet Explorer. После загрузки этой программы в поле Сервер вы можете выбрать один из серверов, на котором в данный момент происходит общение. После того как сервер вами выбран, на экране появится список участников общения и сведения о них: имя, фамилия, адрес электронной почты и др. В строчках некоторых участников имеются значки микрофона и видеокамеры. В этом случае у вас есть возможность установить с этим собеседником голосовую и видеосвязь.

В последнее время большой популярностью пользуется интерактивное общение с помощью сервиса ICQ. Каждый участник такого общения имеет уникальный идентифицирующий его номер, который он получает при регистрации. Чтобы стать участником ICQ, достаточно скачать эту программу, например, с сервера www.freeware.ru и во время установки зарегистрироваться. После запуска программы пользователь может начать общение с любым подключенным в этот момент зарегистрированным членом этого сообщества, которое сейчас насчитывает более 150 млн человек. По меткому выражению одного блоггера, Интернет заполнен болтающими подростками. Мог ли все это представить себе президент Эйзенхауэр, когда отдавал распоряжение об организации DARPA?

Появившись в сравнительно недавнее время, система ICQ содержит в себе многие другие сервисы Интернета: электронную почту, чаты, интернет-телефонию (о которой будет рассказано в следующем параграфе), передачу файлов и т.д.

Личный дневник. В самом названии заложено общение с самим собой, изложение собственного восприятия происходящих вокруг событий и тех эмоциональных переживаний, которые они вызвали. Раньше достоянием общества такие дневники становились как мемуарная литература знаменитых людей. Интернет изменил и эту сторону жизни. Теперь каждый желающий может вести такой дневник в Интернете, причем не только на домашних страницах собственного сайта, который известен только узкому кругу. Живой Журнал (Live Journal; адрес русскоязычного сайта — www.livejournal.ru) — сайт, предназначенный для ведения частных дневников. У этого информационного ресурса 6 млн пользователей; Россия стоит на 3-м месте после США и Австралии. В Живом Журнале можно делать свои записи, комментировать дневники других, вставлять фотографии и иллюстрации, делать гиперссылки, зада-

вать настроение (словесно или смайликом). В Живом Журнале можно искать друзей по интересам. Особенно много объединений, посвященных книгам или образованных «вокруг» книг. Размышления о книгах есть во многих интернет-дневниках. Особую атмосферу интернет-разговоров о книгах создает отсутствие какой бы то ни было цензуры и усиленное сдерживание рекламных атак со стороны книжного бизнеса. Вот несколько адресов таких книжных сообществ:

http://community.livejournal.com/knigi/
http://community.livejournal.com/4itaem_za_vas/
http://community.livejournal.com/fantasy_ru/
http://community.livejournal.com/lovebooks/
http://community.livejournal.com/booktalk_spb/

Еще более широкое общение предоставляют так называемые социальные сети. В нашей стране наиболее популярными являются vkontakte.ru и odnoklassniki.ru. В этих сетях вы можете найти и тех, с кем давно утеряна связь, и тех, кто сейчас находится в тесном с вами общении, но сегодня вас разделяет расстояние.

Для серьезного общения в Интернете существует система **телеконференций** UseNet, которая позволяет тем, кто к ней обращается, участвовать в различных дискуссионных группах. Телеконференции тоже родились не на пустом месте — они представляют собой сетевой вариант так называемых **досок объявлений** (BBS — от **B**ulletin **B**oard **S**ystem), изначально располагавшихся не на серверах, а на компьютерах с модемным доступом.

Тому, кто еще не участвовал в телеконференции, но желает этого, лучше всего начать с просмотра тех телеконференций, которые имеют в своем имени слово newuser (новый пользователь). Например, с телеконференции news.announce.newusers.

Прежде чем задавать вопросы, просмотрите **FAQ-файлы** (от **F**requently **A**sk **Q**uestions — часто задаваемые вопросы) или посмотрите телеконференцию news.answers.

Система имен телеконференций, как обычно, построена по иерархическому принципу. Первый домен указывает общую тематику конференции.

Наиболее распространены следующие основные категории:

biz	Бизнес
comp	Компьютеры
news	Новости общего характера
rec	Развлечения (хобби и искусство)
sci	Наука
soc	Социальные темы
talk	С ориентацией на дискуссию
misk	Темы, не подходящие под вышеуказанные категории
alt	Альтернативные

Телеконференции расширяют возможности человека удовлетворить свои познавательные и культурные потребности. Но Интернет все активнее вторгается в коммерческую сферу. С помощью Интернета проходят деловые встречи и совещания, заключаются сделки, проводятся аукционы. В течение одного дня, не выходя из офиса (а то и просто из домашнего кабинета), руководитель фирмы может провести переговоры с фирмами-компаньонами, находящимися за десятки, сотни и даже тысячи километров. Это в несколько раз повышает возможности осуществления деловой активности, снижает непроизводственные расходы, экономит силы людей.

Но и для рядового пользователя Интернет сегодня предоставляет значительное количество коммерческих услуг. Все большее распространение получают интернет-магазины, для которых процесс общения с покупателем происходит не в специально оборудованном помещении с витринами и стеллажами, на которых располагается товар, со значительным количеством персонала, продавцов и кассиров. В интернет-магазине товар представлен виртуально. На сайте вы найдете фотографии, видеоролик и описание интересующей вас вещи, продукта или услуги (например, туристической поездки). Вы можете сделать заказ, и выбранный вами товар будет вам доставлен. Нередко цены в интернет-магазине оказываются ниже, чем в магазине на соседней улице, ведь торгующей организации не нужно арендовать помещение и содержать столь значительный персонал. В нашей стране, весьма недавно вступившей в стихию свободных рыночных отношений, эта форма торговли пока не завоевала такого доверия, как в странах с устоявшейся рыночной экономикой.

Сегодня многие имеют мобильные телефоны. Оперативная, чаще говорят мобильная, телефонная связь не только удобна, но часто в нашей насыщенной событиями жизни просто необходима. Но можно установить мобильную связь не только между двумя телефонами, но и между телефоном и компьютером. Прежде всего речь идет о возможности отправлять с телефона на компьютер и обратно так называемые SMS (сокращение от Short Message Service — доставка коротких сообщений). Для беспроводного доступа с мобильных телефонов к ресурсам Интернета используется специальный протокол WAP (от Wireless Application Protocol — протокол беспроволочного применения). Отметим, что этот протокол поддерживает связь не с каким-то конкретным компьютером, а с сетью Интернет и его сервисами в целом. Более точно связь поддерживается с так называемыми WAP-сайтами, которые располагаются на Web-серверах и представлены в специальном формате WML (от Wireless Markup Language). Это специальный язык разметки, ориентированный на возможности мобильного телефона — его графику, маленький экран и небольшую память.

На WAP-сайтах вы найдете новости политики, экономики, спорта, прогноз погоды и многое другое. Вы можете отправить электронную почту и поучаствовать в WAP-чате. Тем самым вы оказываетесь в мире Интернета, не имея под рукой компьютер.

Мобильный телефон может быть подключен к компьютеру напрямую. В этом случае телефон превращается в одно из внешних устройств, осуществляющих, в частности, полномасштабную связь компьютера с Интернетом (без модема или других устройств выхода в глобальную сеть). Для такой работы используется технология GPRS — General Packet Radio Service (мы думаем, что смысл этих слов вам ясен и без нашего перевода). Скорость передачи данных в технологии GPRS составляет 171,2 Кбит/с, что почти в 12 раз больше скорости передачи данных в мобильных телефонных сетях стандарта GSM — 9,6 Кбит/с.

Жизнь человека в обществе невозможна без информации о происходящих в нем экономических и политических процессах, без приобщения к культурным событиям своей страны. Еще полвека назад потребность в этой информации удовлетворяли газеты, журналы, радио, телевидение. Все эти источники существуют и сейчас. Но теперь Интернет предоставляет возможность обратиться практически к любому из них в виде интернет-версии. Нередко в них имеются гиперссылки на другие источники, так что, пользуясь такой версией, каждый из вас получает возможность самостоятельно выстроить маршрут для получения разносторонней информации по заинтересовавшему вас вопросу. Ведь глубина освещения вопроса и ракурс его рассмотрения могут весьма отличаться в разных изданиях. Отметим также, что все современные поисковые системы включают в набор своих основных сервисов группу Новости. К примеру, в 2002 г. разработчиками поисковой системы Google была создана специальная служба Google News, которая регулярно и оперативно собирает информацию с разнообразных 4500 сайтов.

Сегодня широкое распространение получила так называемая технология передачи потокового звука и видео. Наиболее используемыми являются технологии RealAudio и RealVideo компании Progressive Networks, а также Windows Media Technology 7 (WMT7) компании Microsoft. С помощью этой технологии в буфер компьютера передаются по частям звуковые и видеофайлы, что позволяет использовать даже модемную связь. Правда, снижение скорости передачи по каналу может приводить к временному пропаданию звука или пропуску кадров. Прослушивание и просмотр таких файлов осуществляются с помощью специальных программ, например RealPlayer или Windows Media Player. Просмотр и прослушивание передач можно осуществлять в фоновом режиме, работая в то же время с другими приложениями. Одним из распространенных вариантов применения этой технологии стала непрерывная трансляция с видеокамер LiveCam, установленных на улицах городов, в музеях, заповедниках и т. д.

В настоящее время достаточно много радио- и телевизионных станций осуществляют вещание через Интернет, используя потоковую технологию. Более того, некоторые компании предлагают самому пользователю сформировать из предлагаемого меню индивидуальную программу передач на его персональный компьютер, так что слушать новости и смотреть любимые фильмы вы сможете именно тогда, когда для вас это будет наиболее удобно.

Возможности Интернета уже сегодня превышают возможность рассказать обо всех них в этой книге. В следующем параграфе мы несколько подробнее расскажем только об одной из них — IP-телефонии.

Вопросы и задания

1. Почему Интернет можно назвать бездонным хранилищем информации?
2. Как устроен адрес электронной почты?
3. В чем суть ftp-сервиса?
4. Как узнать, имеется ли на каком-нибудь ftp-сервере нужная вам информация?
5. Какие возможности предоставляет ICQ-сервис?
6. Для чего предназначены телеконференции? Как стать их участником?
7. Для чего используется протокол WAP?
8. Чем применение технологии GPRS отличается от WAP?
9. Какие преимущества имеют интернет-издания по сравнению с обычными?
10. Какие выгоды дает применение интернет-технологий в бизнесе?
11. Подготовьте сообщение (или компьютерную презентацию) об использовании интернет-технологий в коммерческой деятельности.

§ 43 Интернет-телефония

Идея использования телефонной линии для связи двух компьютеров чрезвычайно стара. С помощью модемов она приспособила изобретение столетней давности к современному компьютеризированному миру. Международный звонок для соединения двух компьютеров и передачи сообщения весьма разорительная операция. Именно поэтому и были придуманы серверы, соединенные друг с другом высокоскоростными линиями связи.

Связь с сервером, находящимся в вашем городе, — вещь не очень дорогостоящая. Это только плата за минуты пребывания в Интернете.

Казалось бы, использовать телефонные линии для того, чтобы соединить компьютеры для использования их как телефоны, немного нелогично. Но это кажется таковым только до тех пор, пока вы не сравните тарифы на Интернет с тарифами на междугородные, а тем более международные переговоры. Можете проверить, что разница весьма значительна.

Разумеется, для того чтобы поговорить с вашим другом в Австралии, необходимо войти в Интернет со скоростью не менее 12 Кбод и одновременно запустить на вашем компьютере специальную программу. Если скорость будет недостаточной, то вряд ли разговор доставит вам удовольствие.

В марте 1996 г. было объявлено о совместной программе VocalTec и крупнейшего производителя оборудования для компьютерной телефонии — корпорации Dialogic. Цель программы — создание программно-аппаратного комплекса для интернет-телефонии с возможностью использования обычного телефона. Проект получил название Internet Telephony Gateway.

В результате был выпущен в коммерческую продажу программно-аппаратный комплекс VocalTec Telephony Gateway (VTG), который в своей работе использует голосовые платы Dialogic в качестве интерфейса с обычными телефонными линиями. При этом, учитывая, что сами голосовые платы многоканальные, одна система VTG может поддерживать несколько телефонных разговоров через Интернет, проходящих одновременно и независимо друг от друга.

VTG также решает еще одну проблему интернет-телефонии — адресацию. Раньше, чтобы позвонить удаленному пользователю, нужно было знать его IP-адрес. Теперь же, чтобы вызвать удаленного пользователя, нужно знать только номер телефона пользователя, а все остальное VTG возьмет на себя.

Как работает VTG?

- VTG, находящийся на городском сервере вызывающего абонента, соединен с телефонной сетью.
- VTG как компонент серверного оборудования соединен с Интернетом и пользуется высокоскоростным каналом. Это дает возможность быстро связаться с любым сервером сети.
- VTG принимает стандартный телефонный вызов, оцифровывает и существенно сжимает речь абонента (используя при этом возможности голосовой платы Dialogic), разбивает полученный файл на пакеты для Интернета и отправляет их, используя TCP/IP-протокол, к адресату.
- VTG, находящийся в районе вызываемого абонента, проделывает обратную операцию для пакетов, приходящих из сети, и выдает их в местную телефонную линию.

- Операции передачи голоса (исходящий голосовой поток и поток, идущий к телефонной сети) происходят одновременно, что позволяет вести нормальный двусторонний диалог.

Фактически VTG многократно увеличивает эффективность использования линии связи, избегая монопольного резервирования канала при прямом телефонном разговоре. Вспомните, во сколько раз увеличилась пропускная способность ARPANET при переходе на пакетную обработку. Примерно то же самое происходит и при использовании интернет-телефонии, что и объясняет ее относительную дешевизну. Мы говорим «относительную» потому, что стоимость минуты разговора уменьшается отнюдь не на порядок, как это можно было бы ожидать при десятикратном увеличении пропускной способности линии за счет пакетной передачи предварительно сжатых данных.

Мы не предлагаем к этому параграфу ни вопросов, ни заданий. Надеемся, что прочитать о рассказанных возможностях Интернета вам было просто интересно.

§ 44 Этика Интернета. Безопасность в Интернете

Живя в любом обществе, человек подчиняется писаным и неписаным законам этого общества. Пока круг общения невелик, такие законы легко познаются и к ним легко привыкнуть. Но вот вы вошли в Интернет — и ваше общение распространилось на весь мир.

Поскольку Интернет — это объединение самых разных глобальных сетей, то каждая из этих сетей имеет свои собственные правила поведения и обычаи, а незнание закона, как известно, не освобождает от ответственности.

Вопрос, что дозволено в Интернете, весьма непростой. На Интернет распространяется действие международных законов, но в то же время это и поле действия национальных законов, которые непрерывно изменяются. Например, посредством Интернета можно передавать информацию через любые государственные границы. Но при передаче чего бы то ни было через государственные границы начинают действовать экспортные законы, которые на разных территориях могут сильно различаться. В частности, эти законы, как правило, требуют лицензии на любой экспорт.

Что касается Интернета, то экспортную лицензию можно отнести к категории «общая лицензия», которая разрешает вывозить все, что не запрещено явно. Так что все, что мы узнаем на телеконференции или передаем по Всемирной сети и на что не наложены ограничения из соображений безопасности, подпадает под об-

щую лицензию. Правда, разные государства по-разному трактуют понятие «национальная информационная безопасность».

При пересылке программного обеспечения необходимо считаться с интеллектуальной собственностью и лицензионными ограничениями. Перед тем как получить что-либо по сети, необходимо знать, кто имеет права на это нечто. Внимательно смотрите, к какой категории относится программный продукт, свободного ли он распространения (free). А перед тем как высылать не свой продукт, убедитесь, что вы имеете на то разрешение.

Немало правовых аспектов, связанных с Интернетом, еще предстоит решить обществу. Законы об электронных коммуникациях, публикациях, платежах, защите данных просто не успевают за прогрессом информационных технологий, и перед государствами стоит непростая задача создания грамотных законов, не нарушающих прав человека.

Что касается этики, то Интернет — общество теоретически очень этичное. Даже имеется соответствующая декларация сетевого сообщества (Материалы интернет-сообщества RFC 1087).

Вот только два его принципа:

- проявление индивидуальности уважается и поощряется;
- сеть следует защищать.

В обществе каждый может претендовать на личную свободу, но в то же время должен идти на компромисс с интересами общины. Аналогична ситуация и в сетевом сообществе, где возможно создание самых необычных сообществ и групп. И люди свободны действовать так, как им нравится. Гонения за взгляды и вкусы строго воспрещены.

Но Интернет, предоставляя почти неограниченные возможности для общения людей, весьма слабо защищен от неэтичного поведения. К себе в дом вы вряд ли пустите любого незнакомца с улицы. А размещая свои ресурсы в Интернете, вы открываете дверь для каждого. Надо помнить об этом, когда на своих страницах вы указываете свой электронный адрес. Будьте тогда готовы, что на вас может хлынуть потоком реклама назойливых продавцов, неприятные вам шутки анонимных шутников и т. п. Все это вы должны учитывать, когда включаете свой компьютер в глобальную сеть. Уметь защищаться — это ваша забота. Например, можно шифровать свои сообщения — такой сервис имеется сейчас для связи по электронной почте.

Общение в Интернете может привести и к весьма неприятным последствиям для вашего компьютера. Угроза безопасности компьютерной системы — это потенциально возможное происшествие, которое может оказать нежелательное воздействие на саму систему, а также на информацию, хранящуюся в ней.

Уязвимость компьютерной системы — это некоторая ее характеристика, которая делает возможным возникновение угрозы безопасности.

Атака на компьютерную систему — это действие, предпринятое злоумышленником, которое заключается в поиске и использовании той или иной уязвимости компьютерной системы.

В последние несколько лет в связи с бурным развитием WWW значительно увеличилось и число атак через Web. В целом все типы атак через Web можно разделить на две большие группы: атака на клиента и атака на сервер.

Перечислим виды атак, под которые может попасть компьютерная система пользователя:

- удаленное проникновение в компьютер: программы, которые получают неавторизованный доступ к другому компьютеру через Интернет (или локальную сеть);
- локальное проникновение в компьютер: программы, которые получают неавторизованный доступ к компьютеру, на котором они работают;
- удаленное блокирование компьютера: программы, которые через Интернет (или сеть) блокируют работу всего удаленного компьютера или отдельной программы на нем;
- локальное блокирование компьютера: программы, блокирующие работу компьютера, на котором они работают;
- вскрыватели паролей: программы, которые обнаруживают легко угадываемые пароли в зашифрованных файлах паролей;
- сетевые анализаторы (sniffers): программы, которые слушают сетевой трафик; часто в них имеются возможности автоматического выделения имен пользователей, паролей и номеров кредитных карт из трафика.

В своем развитии браузеры ушли очень далеко от первоначальных версий, предназначенных лишь для просмотра гипертекста. Функциональность браузеров постоянно увеличивается, сейчас это уже полноценный компонент операционной системы. Параллельно с этим возникают и многочисленные проблемы с безопасностью используемых технологий, таких, как подключаемые модули (plug-ins), элементы ActiveX, приложения Java, средства подготовки сценариев JavaScript, VBScript, PerlScript, Dynamic HTML.

Из-за поддержки этих технологий не только браузерами, но и почтовыми клиентами, а также из-за наличия в них ошибок в последние годы появилось большое количество почтовых вирусов, а также вирусов, заражающих html-файлы (реализованные на VBScript с использованием ActiveX-объектов). Весьма распространены программы, внедряющиеся в систему и сообщающие злоумышленнику конфиденциальную информацию (троянцы). Чаще всего это сетевые реквизиты (учетная запись и пароль) для удаленного соединения. Существуют программы, позволяющие несанкционированно проникнуть в компьютер, подключенный к Интернету. Одна из таких — BackOrifice 2000.

У обычного пользователя не так уж и много средств для защиты в Интернете, но они весьма действенные.

1. Никогда не следует вскрывать какие бы то ни было присоединенные файлы в почтовом отправлении, если адресат вам неизвестен или внушает сомнение.
2. Периодически следует проверять систему антивирусными программами, а также обновлять саму антивирусную систему.
3. Полезно использовать брандмауэр, который служит «экраном» между глобальной сетью и подключенными к ней компьютерами пользователя. Брандмауэр позволяет по заданным признакам фильтровать информацию, поступающую из сети. Отметим, что операционная система Windows XP содержит встроенный брандмауэр.
4. При навигации по Интернету ни в коем случае не следует игнорировать предупреждения браузера о том, что вы переходите в опасную зону. В этом случае он требует от вас подтверждения дальнейшего продвижения (в частности, такое подтверждение требуется при просмотре на сервере провайдера статистики собственных соединений и состояния счета). Если такое предупреждение получено в незнакомой области Интернета, стоит отказаться от дальнейшего передвижения в этом направлении.
5. Не игнорируйте предупреждения о наличии в электронном документе макроопределений. Лучше отказаться от их загрузки, а если вдруг выяснится, что в обычном документе Microsoft Word их и в принципе быть не может, — это явный признак заражения документа компьютерным вирусом.

Подробнее о компьютерных вирусах и защите информации в целом мы поговорим в § 46.

Вопросы и задания

1. В чем вы видите правовые проблемы в использовании Интернета?
2. В чем может состоять угроза безопасности компьютерной системы при работе в глобальных телекоммуникационных сетях?
3. Как понимается уязвимость компьютерной системы?
4. Что такое атака на компьютерную систему?
5. Каким видам атак может подвергнуться компьютер пользователя, подключенный к глобальной сети?
6. Верно ли, что пользователю Интернета угрожают только вирусы?

§ 45 Информационная безопасность и защита интересов субъектов информационных отношений

Любая информационная система нужна для того, чтобы предоставлять пользователям тот или иной набор услуг. Оценкой эффективности работы информационной системы может служить уровень доступности ее сервисов, а для отдельного пользователя мерой эффективности служит время, за которое будет получен ответ на его запрос. Наиболее распространенной угрозой информационной безопасности информационных систем является блокирование их узлов доступа.

В историю функционирования популярных интернет-серверов Yahoo, Buy, Amazon черными днями войдут 7—9 февраля 2000 года, когда они были выведены из строя хакерской атакой в виде огромного потока запросов к этим серверам. Например, поток запросов на сервер Buy в 24 раза превысил средние показатели и в 8 раз максимально допустимую нагрузку. По разным оценкам, ущерб, нанесенный за три часа простоя сервера, составил 6 млрд долл. США.

Под **информационной безопасностью** понимают защищенность информации и поддерживающей инфраструктуры информационной системы от случайных или преднамеренных воздействий естественного или искусственного характера, которые могут нанести ущерб субъектам информационных отношений, имеющих место в рамках данной информационной системы. Субъектами информационных отношений являются владельцы и пользователи информации, включенные в единую систему информационного взаимодействия. К примеру, субъектами информационных отношений являются банковские учреждения и их клиенты — жители города.

Спектр интересов субъектов информационного взаимодействия включает следующие категории:

- доступность информации;
- целостность информации;
- конфиденциальность информации.

Доступность — это возможность получения необходимой информационной услуги в течение приемлемого времени.

Целостность — это однозначность и полнота информации, ее защищенность от разрушения и несанкционированного изменения.

Конфиденциальность — это защита от несанкционированного доступа к информации.

Окинавская Хартия глобального информационного общества провозглашает доступность приоритетной категорией, поскольку

«суть стимулируемой информационно-коммуникационными технологиями экономической и социальной трансформации заключается в их способности содействовать людям, обществу в использовании знаний и идей. Они также дают возможность частным лицам, фирмам и сообществам, занимающимся предпринимательской деятельностью, более эффективно и творчески решать экономические и социальные проблемы». Это, однако, не означает, что можно ослабить внимание к целостности и конфиденциальности.

Вопросы и задания

1. Что понимают под информационной безопасностью информационной системы?
2. Какая категория интересов субъектов информационных отношений была нарушена в результате хакерской атаки на серверы Buy 7 мая 2000 года?
3. а) Почему доступность информации для субъектов информационных отношений способствует предпринимательской деятельности?
   б) Какую роль, по вашему мнению, играет целостность информации в успешном развитии предпринимательской деятельности?
   в) Какую роль играет конфиденциальность информации в успешном развитии предпринимательской деятельности?
4. В чем, на ваш взгляд, проявляются доступность, целостность и конфиденциальность информационного взаимодействия между банком и его клиентами?
5. В чем, на ваш взгляд, может состоять неприемлемый ущерб при нарушении: а) доступности; б) целостности; в) конфиденциальности?

§ 46 Защита информации

Мы начнем с того, что процитируем Федеральный закон «Об информации, информатизации и защите информации», принятый Государственной Думой РФ 26 февраля 1995 года. В статье 20 этого Закона сказано, что «целями защиты являются:

- предотвращение утечки, хищения, утраты, искажения, подделки информации;
- предотвращение несанкционированных действий по уничтожению, модификации, искажению, копированию, блокированию информации; предотвращение других форм незаконного вмешательства в информационные ресурсы и информационные системы...».

Данная цитата показывает, насколько широко понимается защита информации. Но чтобы говорить о защите, надо прежде всего сказать, каким атакам может подвергнуться информация, хранящаяся в компьютере или в компьютерной сети.

Об одном средстве атаки мы упоминали в § 44 — компьютерный вирус. В 2002 году было зарегистрировано 65 000 вирусных атак. Основная их доля пришлась на узлы компьютерных телекоммуникаций, принадлежащих США, Великобритании и Германии (табл. 4.3).

Таблица 4.3

№ п/п	Государство	Количество компьютерных атак
1	США	32 434
2	Бразилия	7294
3	Великобритания	5589
4	Германия	5301
5	Италия	3682
6	Франция	2693
7	Канада	2642
8	Дания	2061
9	Австралия	1506
10	Южная Корея	1320

Компьютерным вирусом называют программу, способную к самовоспроизведению (возможно, в измененном виде) и производящую несанкционированные пользователем изменения в информации, хранящейся в компьютере. Компьютерный вирус, каким бы он ни был, фактически ведет к уничтожению информации, имеющейся в компьютере. Даже самый простенький вирус, всего лишь нагло размножающийся при каждом обращении процессора к винчестеру, способен парализовать работу, заполнив всю память, а избавиться от него без специальных программ можно, только переформатировав винчестер.

Проявлять себя компьютерный вирус может по-разному. Например, это может выглядеть как нарушение работы операционной системы: вы хотите скопировать файл, а вместо этого копируется

какой-нибудь его небольшой фрагмент, на экране вдруг начинают сыпаться буквы, информация в том или ином файле становится недоступной и т. п. Обычно так ведут себя так называемые **перезаписывающиеся вирусы.** Вирусы этого типа записывают себя вместо кода программы, не изменяя имени исполняемого файла. Поэтому при запуске программы выполняется код вируса, а не исходная программа.

Другая разновидность вируса — так называемые **компаньон-вирусы.** Они тоже создают свою копию на месте заражаемой программы, но не уничтожают оригинальный файл, а переименовывают его или перемещают. При запуске программы сначала выполняется код вируса, а затем управление передается исходной программе.

Вирусы, которые называются **файловыми червями,** нередко создают собственные копии с привлекательными для пользователя именами: Game.exe, install.exe и т. п. — в расчете, что пользователь не устоит от искушения запустить такой файл.

Первый **сетевой червь** был создан в 1988 году. Первоначально он разрабатывался как «безвредный» и имел целью скрытно проникать в системы, связанные с сетью ARPANET, и оставаться там необнаруженным. Программа раскрывала пароли, существующие в сети, что позволяло ей маскироваться под легальных пользователей системы и под чужими именами рассылать по сети свои копии. Однако ввиду ошибок программирования вирус вышел из-под контроля автора и начал стремительное саморазмножение. По самым скромным оценкам, инцидент с этим вирусом стоил свыше 8 миллионов часов потери доступа к системе и около миллиона часов прямых потерь, потраченных на восстановление работоспособности сетевых систем. Червь поразил более 6200 компьютеров, а большинство сетей вышло из строя более чем на 5 суток. Общая стоимость потерь оценивается в 96 миллионов долларов. Ущерб был бы гораздо больше, если бы он изначально создавался с разрушительными целями. Тем не менее судом присяжных автор был признан виновным и приговорен к двум годам заключения условно, 400 часам общественных работ и штрафу в размере 10 000 долларов США. К сожалению, история сетевых червей этим не закончилась.

Троянская программа — это компьютерный вирус, который якобы выполняет некоторую полезную функцию, а на самом деле делает совсем другую работу. Например, программа имитирует текстовый редактор или какую-нибудь игру и при этом выполняет не санкционированные пользователем действия: удаление или изменение данных, сбор и пересылку информации другим лицам, передачу управления удаленному пользователю и т. д. В качестве троянца может использоваться программа, предлагаемая для скачивания под предлогом обновления или улучшения уже существующих версий, но в которую автор заранее поместил фрагмент, выполняющий вредоносные функции. Свое название этот класс прог-

рамм получил в честь известного из гомеровской «Илиады» троянского коня.

Наконец, **шпионские программы** (Spyware: от английского spy — шпион и software — программное обеспечение). Они обычно распространяются вместе с другим полезным программным обеспечением и занимаются сбором информации на компьютере пользователя и отсылкой ее создателю. Одним из типичных признаков наличия шпионской программы может служить появление в окне программного приложения дополнительных панелей инструментов.

Как видно из уже перечисленного, основным каналом распространения вирусов стали глобальные сети. Зараженные компьютеры клиентов сети нередко используются для рассылки другим пользователям «мусорной» информации (например, рекламы, не запрашиваемой данным пользователем), что мешает эффективному функционированию сети. Для такой информации появился даже специальный термин — **спам**, а ее производители и рассыльщики называются спамерами. Можно сказать, что создание вирусов сегодня превратилось из одиночного, подчас хулиганского занятия в бизнес, имеющий связи с бизнесом спама и другой противозаконной деятельностью. Поэтому во многих государствах применение вирусных программ рассматривается как уголовное преступление.

Каковы же средства борьбы с компьютерными вирусами? Это антивирусные программы и аппаратные средства антивирусной защиты. **Антивирусная программа** — это программа для обнаружения вируса и, возможно, восстановления зараженных им программ. Антивирусная программа также служит для предотвращения заражения вирусом. В настоящее время активно используются антивирусные программы Касперский, KIS, NOD32, Panda и др.

Современная классификация антивирусных программ выглядит так:

- **детекторы**, позволяющие обнаруживать файлы, зараженные каким-либо известным вирусом (обычно их снабжают еще и функцией доктора);
- **доктора (фаги)**, не только обнаруживающие зараженные файлы, но и пытающиеся вернуть их в исходное состояние;
- **ревизоры**, которые контролируют изменения в местах вероятных компьютерных атак; с этой целью они запоминают сведения о начальных, предположительно незараженных состояниях программ и системных областях дисков и затем сверяют их с текущим состоянием в указанный пользователем момент работы компьютера, например при очередной загрузке операционной системы;
- **доктора-ревизоры**, сочетающие в себе свойства двух указанных выше видов программ;

- **фильтры**, перехватывающие те обращения к операционной системе, которые используются вирусами для размножения и нанесения вреда;
- **вакцины**, или **иммунизаторы**, модифицирующие программы таким образом, чтобы, не теряя своей работоспособности, они казались вирусу, против которого проводится иммунизация, зараженными, и он к ним уже «не приставал».

Обычно программы-фаги обезвреживают не один какой-то вирус, а работают сразу против всех известных вирусов, такие многонаправленные программы называют **полифагами**. Более того, современные полифаги снабжены эвристическим анализатором, позволяющим предупреждать о вероятной атаке ранее неизвестного вируса.

Детекторы устроены иначе — они представляют собой базу данных известных к текущему моменту вирусов и осуществляют их розыск в памяти компьютера по специфическим для каждого вируса признакам. Работают они значительно быстрее. Еще быстрее работают ревизоры. Что касается иммунизаторов, то их действие обычно малоэффективно.

В соответствии с этими особенностями и выстраивается эшелонированная антивирусная защита. При каждом запуске операционной системы работает ревизор, при поступлении на компьютер информации из «внешнего мира» в действие приводятся детекторы, ну и конечно необходимо регулярное обследование компьютера полифагами.

А что делать, если вирус на вашем компьютере обнаружен? Вот несколько рекомендаций:

- не торопитесь и не принимайте опрометчивых решений — непродуманные действия могут привести не только к потере части файлов, но и к повторному заражению компьютера;
- выключите компьютер, чтобы вирус не продолжал своих разрушительных действий;
- лечение компьютера с помощью антивирусных программ следует выполнять только при загрузке компьютера с лицензионного диска. Если вы производите загрузку с дискеты, то она должна быть специально подготовлена: ее надо заблаговременно отформатировать как системную, записать на нее антивирусную программу и обязательно защитить от записи. Дело в том, что существуют вирусы, нападающие на антивирусные программы. Например, пока антивирусная программа проверяет жесткий диск, вирус заражает файлы этой программы;
- если вы не обладаете достаточными знаниями или опытом для лечения зараженных файлов, попросите помочь вам учителя или других более опытных специалистов.

Аппаратные антивирусные средства представляют собой дополнительные устройства, практически полностью предотвращающие вирусную атаку.

Главное в антивирусной профилактике — не разглашать своего пароля, активно и грамотно пользоваться предоставляемыми средствами защиты информации, не пользоваться на своем компьютере чужими файлами, для которых нет гарантий вирусной незараженности. Особую опасность представляет нелицензированное программное обеспечение. Надо помнить хорошую пословицу — бесплатный сыр бывает только в мышеловке. Заметим, что первая вирусная эпидемия, поразившая в 1968 году только в США 18 000 компьютеров, была вызвана вирусом, созданным братьями Амджатом и Базитом Алви как наказание местным пиратам за то, что они воруют программное обеспечение, разрабатываемое их фирмой. Вирусный джинн, однако, вырвался из бутылки.

Впрочем, воровать и пользоваться ворованным программным продуктом нельзя не только потому, что за этим может последовать наказание — по закону или внезаконным вирусом. Коммерческая честность, соблюдение авторских и имущественных прав — основа современного экономического и информационного общества. Иными словами, надо каждому вести себя этично и способствовать в этом своим партнерам.

Во всем сказанном выше есть некая однобокость — речь идет только о защите от *преднамеренных* действий. Но информация может быть утеряна, искажена или блокирована *непреднамеренно*, например, из-за ошибочного действия пользователя или сбоя оборудования. Для предотвращения этого создаются резервные копии программ и документов. А большинство современных средств информационных технологий предусматривают автоматическое сохранение информационного продукта в ходе его разработки.

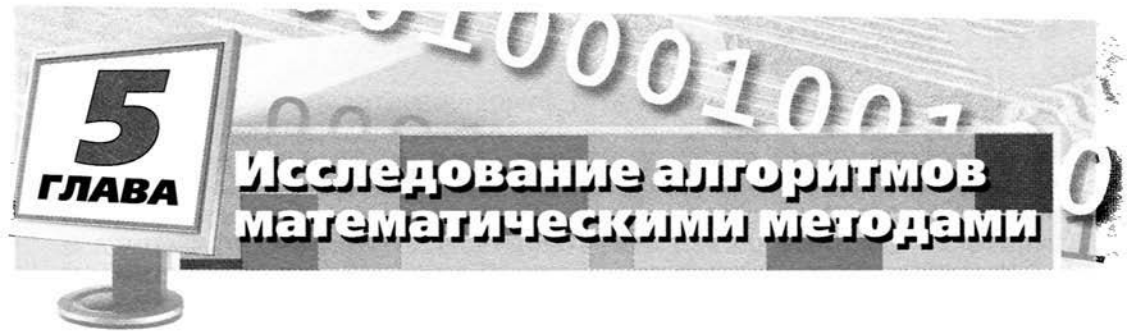
Защита от случайной потери или изменения информации осуществляется в основном следующими материалами:

- обязательным запросом на подтверждение команды, приводящей к изменению содержания какого-либо файла или группы файлов;
- установкой атрибутов, ограничивающий возможность изменения файла (например, с помощью атрибута «*только для чтения*»);
- возможностью отменить последнее действие;
- разграничением доступа пользователей к ресурсам, в частности, с помощью системы паролей.

Но и на самом пользователе лежит немаля ответственность за то, чтобы он не нарушал режим информационной защиты.

Вопросы и задания

1. Что такое компьютерный вирус?
2. Каковы типичные пути проникновения вируса в компьютер?
3. Какими мерами можно предотвратить проникновение вируса в компьютер?
4. В Законе «О правовой охране программ для электронных вычислительных машин и баз данных», принятом 23 сентября 1992 года, в статье 15 сказано: «Лицо, правомерно владеющее экземпляром программы для ЭВМ или базой данных, вправе без согласия правообладателя и без выплаты ему дополнительного вознаграждения... изготовлять или поручать изготовление копии программы для ЭВМ или базы данных при условии, что эта копия предназначена только для архивных целей и при необходимости (в случае, когда оригинал программы для ЭВМ или базы данных утерян, уничтожен или стал непригодным для использования) для замены правомерно приобретенного экземпляра».
   Каким целям защиты информации отвечает эта статья Закона, призванная защищать авторские права разработчиков программного и информационного обеспечения компьютеров?
5. В Уголовном кодексе Российской Федерации, принятом 13 июня 1996 года, в статье 273 сказано: «Создание программ для ЭВМ или внесение изменений в существующие программы, заведомо приводящих к несанкционированному уничтожению, блокированию, модификации либо копированию информации, нарушению работы ЭВМ, системы ЭВМ или их сети, а равно использование либо распространение таких программ или машинных носителей с такими программами наказывается лишением свободы на срок до трех лет со штрафом в размере от двухсот до пятисот минимальных размеров оплаты труда...»
   Как принято называть программы, за изготовление и распространение которых предусмотрено наказание в цитированной выше статье УК?
6. Как вы понимаете этику Интернета?



В учебнике информатики для 10 класса мы неоднократно с разных точек зрения обсуждали понятие алгоритма. Вы осваивали приемы построения алгоритмов, познакомились с машиной Тьюринга как одним из вариантов универсального исполнителя алгоритмов обработки символьной информации, узнали о существовании алгоритмически неразрешимых задач. Вот формулировка одной из таких задач: по тексту алгоритма и исходным данным определить, закончит ли этот алгоритм свою работу за конечное число шагов. В указанном учебнике приведено доказательство утверждения, что невозможно составить алгоритм, который бы решал данную задачу (это и означает ее алгоритмическую неразрешимость). С одной стороны, на это утверждение можно смотреть с грустью — из него, в частности, следует, что нельзя создать такой транслятор с языка программирования, который бы находил не только синтаксические ошибки, но и проверял, не зациклится ли транслируемая программа. С другой стороны, этот факт означает, что создание алгоритмов всегда останется творческим актом человеческого разума, и каждый раз, создав тот или иной алгоритм, человек должен сам убеждаться в правильности его работы (в частности, в том, что через конечное число шагов алгоритм завершит работу). В этой главе мы рассказываем о математических методах, которые применяются для решения указанной проблемы.

§ 47 Еще раз о понятии «алгоритм»

Понятие алгоритма в его общем виде принадлежит к числу основных понятий, не допускающих определение в терминах более простых понятий. Чтобы тем не менее дать представление об алгоритме, говорят, что алгоритм — это точное предписание, которое задает процесс, начинающийся с некоторого исходного данного (из некоторой совокупности данных, допустимых для данного алгоритма) и направленный на получение результата, полностью определяемого этим исходным данным. Вообще говоря, при этом

не предполагается, что результат будет обязательно получен: вполне возможно, что при применении алгоритма к некоторому конкретному исходному данному процесс его преобразования может закончиться безрезультатно (так называемая **безрезультатная остановка**) или не закончиться вовсе. В случае, если процесс заканчивается получением результата, говорят, что **алгоритм применим** к рассматриваемому допустимому исходному данному. В противном случае говорят, что алгоритм неприменим к этому исходному данному.

Чтобы лучше уяснить обсуждаемые понятия, давайте рассмотрим пример.

Пусть допустимыми исходными данными и возможными результатами являются всевозможные слова, составленные с использованием всего двух символов *a* и *b*. Составим следующий алгоритм:

```
Алгоритм Преобразование_слов
сим: A, B;
{  Запросить A;
   Делать пока (Часть (A, 1, 2) ≠ "aa")
   {  Если (Часть (A, 1, 1) = "a") то
         {  A : = Часть (A, 2, LEN (A) − 1) + "b";
         }
      иначе
      {  Если (Часть (A, 1, 2) = "ba") то
         {  A : = Часть (A, 3, LEN (A) − 2) + "aba";
         }
      }
   }
   Сообщить Часть (A, 3, LEN (A) − 2);
}
```

Напомним, что LEN(*A*) обозначает оператор, вычисляющий количество символов в значении символьной переменной *A*, Часть (*A*, *b*, *c*) — оператор, который в значении символьной переменной *A* выделяет часть, начинающуюся с символа, занимающего *b*-е место, и содержащую *c* символов.

Возьмем слово *babaa* в качестве исходного данного, т. е. именно оно будет присвоено переменной *A* при выполнения команды **Запросить**. После первого исполнения тела цикла переменная *A* будет равна слову *baaaba*; после второго — слову *aabaaba*. На этом исполнение тела цикла закончится, и будет выдан результат — слово *baaba*. Следовательно, наш алгоритм применим к слову *babaa*.

Возьмем теперь в качестве исходного слово *babaa*. После первого исполнения тела цикла получится слово *abaaba*. Затем получатся слова *baabab*, *abababa*, *bababab*, *babababa*, ... Нетрудно увидеть, что процесс получения слов никогда не закончится. Следовательно, данный алгоритм неприменим к слову *babaa*.

Можно, конечно, испытывать и другие слова на применимость, но мы думаем, каждому ясно, что нужны какие-то универсальные методы для определения множества тех исходных данных, к которым применим тот или иной алгоритм. Такое множество называется **областью применимости данного алгоритма**. О методах, позволяющих обосновать, какое множество исходных данных является областью применимости, будет рассказано в последующих параграфах этой главы. В основном будут рассматриваться алгоритмы, предназначенные для обработки числовых данных. Однако можно говорить, например, об алгоритме перевода с одного языка на другой, об алгоритме работы диспетчера железнодорожной станции, который информацию о ситуации на подъездных путях перерабатывает в соответствующие указания, и т. д. Важно только каждый раз, говоря о построении алгоритма, точно указывать исполнителя и набор его допустимых действий, а также описывать среду (т. е. набор объектов и связей между ними), в которой действует данный исполнитель.

Вопросы и задания

1. Почему, на ваш взгляд, нельзя считать определением приведенное в начале параграфа объяснение того, что такое алгоритм?

2. Что значит: алгоритм применим к заданному исходному данному?

3. а) Из символов *a* и *b* можно составить 8 трехбуквенных слов: *aaa, aab, aba, abb, baa, bab, bba, bbb*. К каким из этих слов применим алгоритм Преобразование_слов, приведенный в объяснительном тексте параграфа?
   б) Составьте всевозможные четырехбуквенные слова из символов *a* и *b* и выясните, к каким из них применим алгоритм Преобразование_слов.

4. Рассмотрите приведенный ниже алгоритм.
   Среда: чудо-дерево, на котором растут апельсины и бананы, обладающее следующим свойством: если с него срывают два одинаковых плода, то тут же вырастает один апельсин, а если два разных, то один банан.
   Исполнитель: Садовник.
   Допустимые действия: сорвать два плода с чудо-дерева; проверить, что количество плодов на чудо-дереве больше заданного натурального числа.

   Алгоритм Сбор урожая
   { **Делать пока** (на чудо-дереве более одного плода)
   { Сорвать два плода;

```
    }
}
```

Зависит ли результат от того, какой плод будет срывать Садовник при каждом исполнении тела цикла, или он определяется начальной ситуацией на чудо-дереве?

5 Исполнитель имеет следующие допустимые действия:

- стереть с доски два числа и вместо них написать одно, равное их сумме, увеличенной на 1;
- сосчитать количество чисел на доске;
- сравнить два натуральных числа.

Рассмотрите следующий алгоритм:

```
Алгоритм Суммирование
{  Делать пока (на доске более одного числа)
   { Стереть какие-нибудь два числа и вместо них записать
     их сумму, увеличенную на 1;
   }
}
```

Зависит ли результат от того, какие именно два числа будут выбираться на каждом шаге исполнения алгоритма?

6 Библиотекарь обнаружил, что тома Полного собрания сочинений В. Скотта (а это больше 20 томов!) стоят на полке в полнейшем беспорядке. Чтобы расположить их на полке по порядку, библиотекарь решил поступать так: обнаружив два рядом стоящих тома, расположенные в неправильном порядке (т. е. том с меньшим номером стоит правее тома с бóльшим номером), он переставляет их местами. Иными словами, он пользуется следующим алгоритмом:

```
Алгоритм Упорядочение
{  Делать пока (есть два соседних тома, стоящие в неправильном
   порядке)
   {  Поменять эти тома местами;
   }
}
```

Библиотекарь при этом не придерживается никакого правила выбора пары неправильно стоящих томов. Зависит ли результат его исполнения от начальной ситуации?

7 Для каждого из алгоритмов, приведенных ниже в пунктах *а*—*д*, постарайтесь описать множество тех исходных данных, к которым он применим.

а)
```
Алгоритм Суммирование_1
цел: K, N;
вещ: S;
{  Запросить N;
   S := 0;
   Делать от K := 1 до N
   { S := S + K;
   }
   Сообщить S;
}
```

б)
```
Алгоритм Преобразование_в_вопрос
сим: A, B, C;
{  Запросить C;
   A := "goes to school.";
   B := "Does" + " " + C + " " + Часть(A, 1, 2) + Часть(A, 5, 10)
   + "?";
   Сообщить B;
}
```

в)
```
Алгоритм Суммирование_2
цел: K, N;
вещ: a, b, d, S;
{  Запросить a;
   Запросить b;
   Запросить N;
   d := (b - a)/N;
   S := 0;
   Делать от K := 1 до N
   { S := S + 1/(a + d*K);
   }
   Сообщить S;
}
```

г)
```
Алгоритм Значение функции_1
вещ: a, x;
{  Запросить x;
   Запросить a;
   Если (a > 0) то { Сообщить √(2 - a^x - a^(-x)); }
}
```

д)
```
Алгоритм Значение функции_2
вещ: x, y;
{  Запросить x;
   Если (cos x - 1 ⩾ 0) то
   {  y := √(cos x - 1);
      Сообщить y;
   }
}
```

§ 48 Как доказывают применимость алгоритма

Рассмотрим задачу.

Задача 1. Последовательность X_n строится следующим образом:

$X_1 = 1$; $X_2 = 3$; ...; $X_n = X_{n-2} - 2X_{n-1}$ для каждого $n > 2$.

Составьте алгоритм нахождения первого числа в этой последовательности, большего 100 000.

Переменные здесь можно считать имеющими целочисленный тип. Нужный алгоритм может быть записан так:

```
Алгоритм Задача_1
цел: X, Y, Z;
{  X := 1;
   Y := 3;
   Z := Y;
   Делать пока (Z ⩽ 100 000)
   {  Z := X - 2*Y;
      X := Y;
      Y := Z;
   }
   Сообщить Z;
}
```

Изменим условие задачи.

Задача 2. Последовательность X_n строится следующим образом:

$X_1 = 1$; $X_2 = 3$; ...; $X_n = X_{n-2} - 2X_{n-1}$ для каждого $n > 2$.

Найдите первое в этой последовательности число, большее 100 000.

На первый взгляд кажется, что это та же задача. Но давайте задумаемся, что является ответом в каждой из них. В первой это алгоритм, и он уже приведен выше; во второй ответом служит число. Значит, это все-таки не одна и та же задача, хотя ясно, что для решения задачи 2 надо просто запустить алгоритм, служащий ответом в задаче 1.

И тут возникает вопрос: получим ли мы ответ, запустив созданный нами алгоритм? Легко понять, что помешать этому может только одно — отсутствие в данной последовательности хотя бы одного числа, большего 100 000. Алгоритм же в этом случае будет

исполняться бесконечно, вырабатывая все новые и новые члены последовательности.

Что же мы видим? Вопрос о применимости данного алгоритма к исходным данным $X_1 = 1$, $X_2 = 3$ оказался равносилен вопросу о существовании объекта, который этим алгоритмом строится.

Итак, надо убедиться, что в данной последовательности встретится хотя бы один раз число, большее 100 000. Для этого полезно взглянуть на десяток первых членов последовательности:

$$1;\ 3;\ -5;\ 13;\ -31;\ 78;\ -181;\ 437;\ -1055;\ 2547.$$

Глядя на них, можно высказать гипотезу, что члены с четными номерами положительны и неограниченно возрастают. Докажем эту гипотезу. Для этого выразим X_{2k} через предшествующие члены последовательности с четными номерами:

$$\begin{aligned} X_{2k} &= X_{2k-2} - 2X_{2k-1} = X_{2k-2} - 2(X_{2k-3} - 2X_{2k-2}) = 5X_{2k-2} - 2X_{2k-3} = \\ &= 5X_{2k-2} - 2(X_{2k-5} - 2X_{2k-4}) = 5X_{2k-2} + 4X_{2k-4} - 2X_{2k-5} = \\ &= 5X_{2k-2} + 4X_{2k-4} - 2(X_{2k-7} - 2X_{2k-6}) = \\ &= 5X_{2k-2} + 4X_{2k-4} + 4X_{2k-6} - 2X_{2k-7} = \ldots = \\ &= 5X_{2k-2} + 4X_{2k-4} + 4X_{2k-6} + \ldots + 4X_2 - 2X_1. \end{aligned}$$

Поскольку $4X_2 - 2X_1 = 10$, индуктивное рассуждение показывает, что все члены последовательности с четными номерами положительны. Более того, теперь ясно, что $X_{2k} > 5X_{2k-2}$. Применяя это неравенство k раз, получаем $X_{2k} > 5^{k-1}X_2$. Поэтому в данной последовательности имеются члены, большие не только 100 000, но и вообще любого положительного числа. Можно сказать, что мы доказали следующую теорему:

Теорема. *Пусть последовательность X_n определена следующим рекуррентным соотношением:*

$$X_1 = 1; \quad X_2 = 3; \quad \ldots; \quad X_n = X_{n-2} - 2X_{n-1}$$

для каждого $n > 2$. Тогда для любого положительного числа M в этой последовательности существует член, больший, чем M.

Заметьте, что ни в формулировке теоремы, ни в ее доказательстве не указывается, каким по номеру будет искомый член последовательности. Более того, не намечается даже способ нахождения нужного члена последовательности. Теоремы, в которых доказывается существование какого-либо объекта, обладающего требуемыми свойствами, без указания способа построения этого объекта, называют **теоремами чистого существования**. А теоремы, в которых существование нужного объекта доказывается указанием способа его построения, называют **конструктивными теоремами существования.**

Сейчас мы сформулируем задачу, в которой напрямую требуется доказать, что область применимости представленного в ней алгоритма — множество всех положительных чисел.

Задача 3. Дан алгоритм:

```
Алгоритм Задача_3
вещ: A, E, X;
{  Запросить A;
   Запросить E;
   X := 1;
   Делать пока (ABS(X*X – A) ⩾ E)
   {  X := (X + A/X)/2;
   }
   Сообщить X;
}
```

Докажите, что при любых положительных A и E алгоритм завершит свою работу за конечное число шагов.

Не будем скрывать, что данный алгоритм представляет собой способ быстрого вычисления приближенного значения квадратного корня из числа A (параметр E задает, как мы позже увидим, уровень точности). Чтобы продемонстрировать это более отчетливо, преобразуем данный алгоритм таким образом, чтобы после каждого выполнения цикла сообщалось, сколько всего значений X было вычислено:

```
Алгоритм Задача_3м
вещ: A, E, X; цел: N;
{  Запросить A;
   Запросить E;
   X := 1;
   N := 1;
   Делать пока (ABS(X*X – A) ⩾ E)
   {  X := (X + A/X)/2;
      N := N + 1;
   }
   Сообщить X;
   Сообщить N;
}
```

Тогда можно сказать, что каждому вычисляемому значению X присвоен порядковый номер; иными словами, мы определили следующую рекуррентно заданную последовательность:

$$X_1 = 1;\quad X_n = (X_{n-1} + A/X_{n-1})/2 \text{ при } n > 1 \text{ и } A > 0.$$

После того как стало ясно, откуда взялся данный алгоритм и для чего он предназначен, приступим к доказательству утверждения Задачи 3.

Прежде всего отметим, что из формул, определяющих X_n, сразу следует, что $X_n > 0$ при любом n. Из хорошо известного неравенства между средним арифметическим и средним геометрическим вытекает, что при $n > 1$

$$X_n = (X_{n-1} + A/X_{n-1})/2 \geqslant \sqrt{X_{n-1} \cdot A/X_{n-1}} = \sqrt{A}.$$

Отсюда, в частности, следует, что $(X_n)^2 \geqslant A$ при всех $n > 1$.

Далее рассмотрим разность двух соседних членов последовательности:

$$X_{n-1} - X_n = X_{n-1} - (X_{n-1} - A/X_{n-1})/2 = (X_{n-1} - A/X_{n-1})/2 = \\ = ((X_{n-1})^2 - A)/2X_{n-1}.$$

Из вышесказанного следует, что $X_{n-1} - X_n \geqslant 0$ при всех $n > 2$ и

$$X_n - \sqrt{A} = (X_{n-1} - 2\sqrt{A} + A/X_{n-1})/2 = (X_{n-1} - \sqrt{A})^2/(2X_{n-1});$$

$$X_n + \sqrt{A} = (X_{n-1} + 2\sqrt{A} + A/X_{n-1})/2 = (X_{n-1} + \sqrt{A})^2/(2X_{n-1}),$$

откуда, перемножая левые и правые части равенства, получаем

$$X_n^2 - A = (X_{n-1}^2 - A)^2/(4X_{n-1}^2) = (1 - A/X_{n-1}^2)\,(X_{n-1}^2 - A)/4.$$

Поскольку при $n > 2$ справедливо неравенство $X_{n-1}^2 > A$, имеем

$$X_n^2 - A \leqslant (X_{n-1}^2 - A)/4.$$

Переходя в этом неравенстве от n к $n-1$, от $n-1$ к $n-2$ и т. д., получаем следующую цепочку неравенств:

$$X_n^2 - A \leqslant (X_{n-1}^2 - A)/4 \leqslant (X_{n-2}^2 - A)/16 \leqslant \ldots \leqslant (X_2^2 - A)/4^{n-2}.$$

Вспомнив, что $X_2 = (1 + A)/2$, получаем неравенство

$$0 \leqslant X_n^2 - A \leqslant (1 - A)^2/4^{n-1}.$$

Поскольку знаменатель с ростом n неограниченно возрастает, а числитель при этом остается одним и тем же, это неравенство означает, что для любого положительного числа E найдется такой номер, начиная с которого $|X_n^2 - A| < E$, что и требовалось доказать.

Впрочем, из полученного неравенства нетрудно найти номер, начиная с которого $|X_n^2 - A|$ заведомо меньше, чем E. Для этого достаточно, чтобы меньше, чем E, было число $(1 - A)^2/4^{n-1}$. Отсюда $n > 1 + \log_4 (1 - A)^2/E$, т. е. в роли нужного n может выступать число $2 + [\log_4 (1 - A)^2/E]$, и, значит, цикл будет выполняться не более чем $2 + [\log_4 (1 - A)^2/E]$ раз. (Через $[a]$ традиционно обозна-

чается целая часть числа a, т. е. наибольшее целое число, не превосходящее a.)

Интересно оценить, насколько X_n отличается от $\sqrt{A}$, когда $|X_n^2 - A| < E$. Помня, что $X_n \geqslant \sqrt{A}$, легко получить неравенство

$$X_n - \sqrt{A} < E/(X_n + \sqrt{A}) \leqslant E/2\sqrt{A}.$$

Поэтому при $A \geqslant 1$ выполнено неравенство $X_n - \sqrt{A} < E/2$, которое означает, что $E/2$ дает нам оценку точности приближения X_n к $\sqrt{A}$. Именно этот факт мы имели в виду, когда выше говорили, что E задает уровень точности вычисления квадратного корня. А как быть, если $A < 1$? Один из подходов может быть таким: заменим A на $B = 1/A$, вычислим $\sqrt{B}$, а затем воспользуемся тем, что $\sqrt{A} = 1/\sqrt{B}$.

Отличие ситуации этой задачи от ситуации, которая рассматривалась в Задаче 2, состоит в том, что здесь нам удалось не только доказать конечность алгоритма, но и оценить, сколько шагов потребуется, чтобы достичь нужного результата.

Вопросы и задания

1. В чем различие между конструктивными теоремами и теоремами чистого существования?

2. Для решения задачи «В какую наименьшую натуральную степень надо возвести число 3, чтобы получилось число, оканчивающееся на 00001?» был составлен следующий алгоритм:

```
Алгоритм Показатель степени
цел: X, Y, N;
{ X := 1;
  Y := 2;
  N := 0;
  Делать пока (Y > 1)
  { X := 3*X;
    Y := mod(X, 100000);
    N := N + 1;
  }
  Сообщить N;
}
```

а) Объясните, почему данный алгоритм решает поставленную задачу.
б) Докажите, что данный алгоритм конечен.

❸ Дан алгоритм:

```
Алгоритм Последовательность
цел: X, Y, Z, M;
{  Запросить M;
   X := 1;
   Y := 3;
   Z := Y;
   Делать пока (Z < M)
   {  Z := Y - 2*X;
      X := Y;
      Y := Z;
   }
   Сообщить Z;
}
```

Верно ли, что при любом целом M этот алгоритм конечен?

❹ Рассмотрите следующий алгоритм:

```
Алгоритм Суммирование
вещ: S; цел: N;
{  S := 1;
   N := 1;
   Делать пока (S < 20)
         {  N := N + 1;
            S := S + 1/N^2;
         }
   Сообщить N;
}
```

а) Для решения какой задачи предназначен этот алгоритм?
б)* Всегда ли исполнение этого алгоритма заканчивается за конечное число шагов?

❺ Рассмотрите следующий алгоритм:

```
Алгоритм Сумма
вещ: S; цел: N, M;
{  S := 1;
   N := 1;
   Сообщить "Введите натуральное число M";
   Запросить M;
   Делать пока (S < M)
   {  N := N + 1;
      S := S + 1/N;
```

```
    }
    Сообщить N;
}
```

а) Для решения какой задачи предназначен этот алгоритм?
б) При $M = 1$ алгоритм конечен, так как тело цикла не выполнится ни разу. При любом ли значении M данный алгоритм конечен?

6 Рассмотрите следующий алгоритм, преобразующий натуральные числа:

```
Алгоритм Преобразование
цел: a, b, n, m;                    (*n — целое положительное число*)
{ Запросить n;
  Запросить a;
  Запросить b;
  m := n;
  Делать пока (m ≠ a) и (m ≠ b)
  { m := СКВ(m);
  }                                 (*конец цикла*)
  Сообщить m;
}
Функция СКВ (цел: n) : цел
{ Если n < 10 то { знач := n * n; }
  иначе { знач := СКВ (n div 10) + (n mod 10)*(n mod 10); }
}
```

При каких a и b этот алгоритм конечен? Перечислите все возможные здесь варианты.

7 Дан алгоритм:

```
Алгоритм
цел: K, M;
{ Запросить K;
  Запросить M;
  Делать пока (K mod 2 = 0 или M mod 2 = 0)
  { Если (K mod 2 = 0) то
      { M := M + K/2;
      }
    Если (M mod 2 = 0) то
      { K := K + M/2;
      }
  }
  Сообщить M + K;
}
```

Определите, для каких пар натуральных чисел K и M этот алгоритм завершает работу за конечное число шагов.

§ 49 Лимитирующая функция

Рассмотрите еще раз те алгоритмы из § 48, для которых потребовалось доказывать конечность. Отличительной их чертой является наличие конструкции цикла в форме **Делать пока**. Это и понятно: ведь если алгоритм содержит только линейные фрагменты, конструкции ветвления и цикла со счетчиком, то исполнение алгоритма заведомо заканчивается за конечное число шагов. Конечность числа шагов нужно обосновывать лишь в том случае, если в нем используется цикл, для которого заранее неизвестно количество исполнений тела цикла (т. е. цикл с предусловием или цикл с постусловием), или рекурсия. Основным инструментом обоснования конечности алгоритма в этом случае служит **лимитирующая функция**. Так называют переменную величину, которая каждый раз меняет свое значение при очередном исполнении тела цикла или рекурсивном обращении к вспомогательному алгоритму и обладает следующими двумя свойствами:

1. Лимитирующая функция принимает лишь конечное число значений (быть может, зависящее от исходных данных).
2. В ходе исполнения алгоритма лимитирующая функция принимает каждое свое допустимое значение не более одного раза.

Часто лимитирующую функцию выбирают так, чтобы ее значениями были только натуральные числа и при каждом исполнении тела цикла ее значение уменьшалось.

Рассмотрим пример. В учебнике для 10 класса мы упоминали знаменитый алгоритм Евклида, предназначенный для нахождения наибольшего общего делителя двух натуральных чисел. Вот этот алгоритм:

```
Алгоритм НОД
цел: m, n;
{ Запросить m;
  Запросить n;
  Делать пока (не (m = n))
  { Если (m > n) то { m := m − n; }
    иначе { n := n − m; }
  }
  Сообщить m;
}
```

Как доказать, что этот алгоритм действительно вычисляет НОД двух чисел, мы обсудим в следующем параграфе. А вот почему он конечен?

Рассмотрим величину $k = \max\{m; n\}$. До начала исполнения цикла она имеет некоторое значение k_0. Очевидно, что переменная k

принимает только натуральные значения. Заметим также, что при каждом исполнении тела цикла значение величины k обязательно уменьшается. Но натуральных чисел, меньших числа k_0, лишь конечное число. Значит, исполнение тела цикла не может осуществляться более чем k_0 раз. Следовательно, и весь алгоритм заканчивает свою работу за конечное число шагов, какими бы ни были начальные натуральные значения m и n.

Впрочем, далеко не всегда бывает так легко указать лимитирующую функцию. Вспомним, к примеру, задачу 6 из § 47.

Библиотекарь обнаружил, что тома Полного собрания сочинений В. Скотта (а это больше 20 томов!) стоят на полке в полнейшем беспорядке. Чтобы расположить их на полке по порядку, библиотекарь решил поступать так: обнаружив два рядом стоящих тома, расположенные в неправильном порядке (т. е. том с меньшим номером стоит правее тома с бóльшим номером), он переставляет их местами. Иными словами, он пользуется следующим алгоритмом:

Алгоритм
{ **Делать пока** (есть два соседних тома, стоящие в неправильном порядке)
{ Поменять эти тома местами;
}
}

Библиотекарь при этом не придерживается никакого правила выбора пары неправильно стоящих томов и меняет местами первые попавшиеся на глаза тома.

Едва ли очевидно, что этот алгоритм конечен. Не будет ли библиотекарь обречен на сизифов труд? (Миф о Сизифе, по-видимому, первое упоминание о никогда не кончающихся циклических алгоритмах.)

Приведем доказательство, которое предложил Эдсгер Дийкстра. Имя этого голландского программиста — классика науки о программировании (Computer Science) — должно быть известно каждому профессиональному программисту.

Заменим каждый том Вальтера Скотта гирей, вес которой равен номеру заменяемого тома. Воспринимая такое «собрание сочинений» как единое целое, можно говорить о центре его тяжести. Легко понять, что, меняя местами более тяжелую гирю с более легкой, мы смещаем центр тяжести совокупности всех гирь вправо.

Однако ясно, что всего возможных положений центра тяжести — конечное число, не больше, чем различных перестановок этих гирь. Значит, неограниченно долго смещаться вправо центр тяжести не может, и потому наступит момент, когда перестановка соседних гирь (томов) закончится. А следовательно, закончится и исполнение алгоритма. Это, в частности, означает, что нарушится условие продолжения цикла, т. е. все тома окажутся расположенными

в правильном порядке. Таким образом, попутно мы доказали, что является результатом исполнения алгоритма.

Если внимательно присмотреться, то и здесь видна та же идея лимитирующей функции — каждому расположению томов мы ставим в соответствие положение центра тяжести. Однако математическую запись этой функции получить непросто.

Таким образом, для решения задачи надо:

- ввести координатную ось;
- положение каждого тома описать координатой центра тяжести;
- вычислить координаты центра тяжести всего «собрания сочинений» (это и будет лимитирующая функция);
- проверить, что функция возрастает (центр тяжести смещается вправо) при каждом исполнении тела цикла.

Желающие могут это проделать.

Известны весьма простые алгоритмы, для которых решение вопроса об области применимости весьма проблематично. Вот алгоритм, для которого на сегодняшний день не удалось ни доказать, ни опровергнуть гипотезу, что при указанных начальных данных его работа завершается за конечное число шагов:

```
Алгоритм Проблемный
цел: n, K;
{ K := 1;
  Запросить n;
  Делать пока (n > 1)
  { Если (n mod 2 = 1) то { n := 3*n+1; }
    иначе { n := n/2; }
    K := K+1;
  }
  Сообщить K;
}
```

Совершенно ясно, что делает этот алгоритм: если на каком-то шаге получилось четное число, то оно делится пополам, если нечетное, то умножается на 3 с последующим прибавлением 1 (т. е. число становится четным). Тем самым после такого увеличения обязательно идет деление пополам и, быть может, не один раз.

В предлагаемых ниже заданиях вам предстоит обосновывать конечность тех или иных алгоритмов. Теперь вы знаете, что для этого нужно подобрать подходящую лимитирующую функцию.

Вопросы и задания

1. Какие алгоритмические конструкции могут быть причиной бесконечного числа шагов при исполнении алгоритма?
2. Какую функцию называют лимитирующей?

❸ Как с помощью лимитирующей функции доказывают, что исполнение алгоритма заканчивается за конечное число шагов?

❹ а) Рассмотрите алгоритм, обрабатывающий натуральное число n.

```
Алгоритм Превращение_1
цел: n, I;
{  Запросить n;
   I := 1;
   Делать пока (n > 1)
   {  Если (n mod 2 = 1) то { n := n − 1; }
      иначе { n := n/2; }
      I := I + 1;
   }                                    (*конец цикла*)
   Сообщить I;
}
```

Закончится ли исполнение данного алгоритма за конечное число шагов? Ответ обоснуйте.

б) Рассмотрите еще один алгоритм, обрабатывающий натуральное число n.

```
Алгоритм Превращение_2
цел: n, I;
{  Запросить n;
   I := 1;
   Делать пока (n > 1)
   {  Если (n mod 2 = 1) то { n := n + 5; }
      иначе { n := n/2; }
      I := I + 1;
   }                                    (*конец цикла*)
   Сообщить I;
}
```

Закончится ли исполнение этого алгоритма за конечное число шагов? Ответ обоснуйте.

❺ Массив $A[1{:}20;\ 1{:}30]$ заполнен двумя числами: 0 и 1. Рассмотрите следующий алгоритм:

```
Алгоритм Мимикрия
цел: k, m, n, I, J, A[1:20; 1:30];
{  k := 1;
   Делать пока (k = 1)
   {  k := 0;
      Делать от I := 1 до 20
      {     n := 0;
            m := 0;
            Делать от J := 1 до 30
            {     Если (A[I, J] = 0) то { n := n + 1; }
```

```
                Если A[I, J] = 1 то { m := m + 1; }
            }                                    (*конец цикла*)
            Если (m > n) то
            {   Делать от J := 1 до 30
                {   Если A[I, J] = 0 то { A[I, J] = 1; }
                    иначе { A[I, J] = 0; }
                }
                k := 1;
            }
        }                                        (*конец цикла*)
        Делать от J := 1 до 30
        {  n := 0;
           m := 0;
           Делать от I := 1 до 20
           {    Если A[I, J] = 0 то { n := n + 1; }
                Если A[I, J] = 1 то { m := m + 1; }
           }                                     (*конец цикла*)
           Если (m > n) то
           {    Делать от I := 0 до 20
                {    Если A[I, J] = 0 то { A[I, J] = 1; }
                     иначе { A[I, J] = 0; }
                }
                k := 1;
           }
        }                                        (*конец цикла*)
    }
```

Верно ли, что этот алгоритм применим к любому массиву указанного типа? Ответ обоснуйте.

6 Предположим, что исполнитель обладает способностью за одно действие написать весь натуральный ряд, а также вычеркнуть из него все числа, удовлетворяющие какому-либо условию, даже если этих чисел бесконечно много. Рассмотрите следующий алгоритм для этого исполнителя:

```
Алгоритм Решето
{ Написать весь натуральный ряд;
  Вычеркнуть из него число 1;
  Делать пока (есть необведенные числа среди невычеркнутых)
    { Среди невычеркнутых чисел обвести самое маленькое
      из необведенных;
      Из необведенных чисел вычеркнуть те, которые кратны
      последнему обведенному числу;
    }                                        (*конец цикла*)
  Сообщить Обведенные числа;
}
```

а) Конечен ли этот алгоритм? Ответ обоснуйте.

б) Попытайтесь определить, для чего предназначен этот алгоритм, придуманный древнегреческим математиком Эратосфеном. Докажите свою гипотезу.

7 Рассмотрите следующий алгоритм:

```
Алгоритм
цел: K, M;
{ Запросить K;
  Запросить M;
  Делать пока (K mod 2 = 0 или M mod 2 = 0)
  { Если (K mod 2 = 0) то
    { K := K/2;
      M := M + K;
    }
    Если (M mod 2 = 0) то
    { M := M/2;
      K := M + K;
    }
  }
  Сообщить M * K;
}
```

Определите, для каких пар натуральных чисел *K* и *M* этот алгоритм завершает работу за конечное число шагов.

8 Дан массив *M*[1:30], состоящий из положительных чисел. Рассмотрите следующий алгоритм (напомним, что через rand обозначен датчик случайных чисел, генерирующий вещественное число из промежутка [0; 1)):

```
Алгоритм
вещ: x, y, M[1:30];
цел: I, J;
{ y := 1;
  Делать пока (y > 0)
  { I := INT (29,5*rand) + 1;
    J := INT (29,5*rand) + 1;
    x := (M[I] + M[J])/2;
    y := ABS(M[I] - M[J])/2;
    M[I] := x;
    M[J] := y;
  }
  Сообщить "Случайность не помеха";
}
```

а) Верно ли, что этот алгоритм применим к любому массиву *M*, содержащему только натуральные числа? Ответ обоснуйте.

б) Будет ли применим этот алгоритм, если массив *M* состоит из произвольных положительных чисел? Ответ обоснуйте.

§ 50 Инвариант цикла

Алгоритм написан. А как доказать, что результатом его работы является именно то, что требовалось? Рассмотрим, к примеру, следующий алгоритм:

```
Алгоритм
цел: k, n; вещ: a, b, c;
{ Запросить k;
  Запросить a;
  b := 1;
  n := k;
  c := a;
  Делать пока (n > 0)
  { Если (n mod 2 = 0) то
    { c := c*c;
      n:= n/2;
    }
    иначе
    { b := b*c;
      n := n - 1;
    }
  }
  Сообщить b;
}
```

Как вы думаете, для чего предназначен этот алгоритм? Гипотез может быть много. И в любую из них можно поверить. Можно взять конкретные два числа и исполнить для них алгоритм. Вера в высказанную гипотезу от этого испытания может возрасти, либо мы убедимся, что гипотеза неверна.

Можно взять много разных пар чисел и на них проверить высказанную гипотезу. Но все равно это не будет доказательством, что алгоритм предназначен именно для того, что было заявлено.

Как же изучать алгоритмы доказательно?

Исследовать алгоритмы, в которых нет циклов и рекурсии, в общем-то легко: в них каждое действие исполняется не более одного раза. Для такого алгоритма можно просто составить протокол исполнения, записывая результаты последовательного исполнения всех действий.

С циклами и рекурсией все обстоит сложнее: далеко не всегда можно указать конкретное число исполнений тела цикла или обращений к рекурсивной подпрограмме — оно само может оказаться переменной величиной.

Поэтому поступают так. Отыскивают какое-либо свойство объекта, обрабатываемого алгоритмом, такое, которое не меняется при

исполнении тела цикла. Такое свойство называют **инвариантом цикла**. Если таким объектом является некоторый набор величин, то инвариант цикла нередко разыскивают в виде соотношения между этими величинами.

Для цикла, записанного в нашем алгоритме, рассмотрим величину bc^n. До начала цикла она равна a^k. А теперь составим протокол исполнения тела цикла (см. табл. 5.1).

Таблица 5.1 **Протокол исполнения тела цикла**

Действие и комментарий к нему	**Изменение величины bc^n**
Проверка условия (n mod 2 = 0).	Величина bc^n изменяться не может, поскольку не меняются величины b, c и n.
Предположим, что условие выполняется. Выполняются действия, записанные в операторных скобках после слова **то**.	
$c := c*c;$	Величина c меняется на $c' = c^2$.
$n := n/2;$	Величина n меняется на $n' = n/2$. Величина $(c')^{n'}$ не изменяется: $(c')^{n'} = (c^2)^{n/2} = c^n$. Поскольку величина b не менялась, то не изменилась и величина bc^n.
Исполнение первой ветви ветвления закончено.	
Предположим теперь, что условие не выполнено. Исполняются действия, записанные в операторных скобках после слова **иначе**.	
$b := b*c;$	Величина b меняется на $b' = bc$.
$n := n - 1;$	Величина n меняется на $n' = n - 1$. Величина $b'c^{n'}$ не изменяется: $b'c^{n'} = (bc)c^{n-1} = bc^n$. Поскольку величина c не менялась, то не изменилась и величина bc^n.
Исполнение ветвления закончено.	
Исполнение тела цикла закончено.	

Итак, мы проверили, что величина bc^n — инвариант цикла. Это значит, что все время $bc^n = a^k$, в том числе и после выхода из цикла.

Что же мы будем иметь по окончании работы цикла? Чтобы понять это, нужно посмотреть условие окончания цикла (а вовсе не проделывать все операции в цикле, как могут подумать некоторые).

Это условие: $n > 0$. Значит, по окончании исполнения цикла $n = 0$. Но тогда справедливо равенство $bc^0 = a^k$. Следовательно, после исполнения цикла $b = a^k$, и именно это значение b сообщит алгоритм накануне своего завершения.

Вот мы и выяснили (более того, доказали), что является результатом алгоритма. Чтобы убедиться в наличии у этого алгоритма свойства результативности, надо еще доказать его конечность. Мы надеемся, что вы легко сделаете это самостоятельно, указав подходящую лимитирующую функцию.

Вопросы и задания

1. Что такое инвариант цикла?

2. Рассмотрите следующий алгоритм, обрабатывающий натуральное число k:

```
Алгоритм
цел: k, m, n;
{ Запросить k;
  m := 0;
  Делать от n := 0 до k
  { m := m + 2*n + 1;
  }
  Сообщить m;
}
```

Для чего предназначен данный алгоритм? Выдвинутую гипотезу докажите.

3. Рассмотрите следующий алгоритм, обрабатывающий натуральные числа m и n:

```
Алгоритм Загадка
цел: m, n, x, y, u, v;
{ x := m;
  y := n;
  u := m;
  v := n;
```

```
Делать пока (x < y) или (x > y)
{  Если (x < y) то
   { y := y − x;
     v := v + u;
   }
   иначе
   { x := x − y;
     u := u + v;
   }
}
y := (u + v)/2;
Сообщить x, y;
}
```

а) Конечен ли этот алгоритм? Ответ обоснуйте.

б) Для чего предназначен данный алгоритм? Выдвинутую гипотезу докажите.

4 Даны два массива K и M с целочисленными элементами, расположенными в порядке возрастания. Рассмотрите следующий алгоритм:

```
Алгоритм Количество
цел: a, b, c, K[1:20], M[1:30];
{  a := 1;
   b := 1;
   c := 1;
   Делать пока (a ⩽ 20) и (b ⩽ 30)
   {  Выбор
         при K[a] < M[b] { a := a + 1; }
         при K[a] > M[b] { b := b + 1; }
         при K[a] = M[b]
         {   a := a + 1;
             b := b + 1;
             c := c + 1;
         }
   }                                    (*конец цикла*)
   Сообщить c;
 }
```

а) Для решения какой задачи предназначен этот алгоритм?

б) Найдите подходящий инвариант и докажите гипотезу, выдвинутую вами при выполнении пункта *а*.

6 ГЛАВА
Графы и алгоритмы на графах

Графы — мощное средство моделирования. Оглянитесь на пройденное вами в 10 классе, и вы убедитесь, что не раз графы возникали при обсуждении весьма, казалось бы, далеких друг от друга вопросов. Транспортная схема (скажем, схема линий метрополитена) и схема предложения, генеалогическое дерево и дерево папок (или каталогов) в памяти компьютера, схема переходов для состояний конечного автомата и схема переходов в экспертной системе — вот далеко не полный перечень тех примеров, когда при изучении информатики в 10 классе возникало понятие графа. В 11 классе вы имели дело с графами, когда рассматривали расстояние Хэмминга между словами и строили алгоритм Хаффмана. Короче говоря, графы заслуживают того, чтобы уделить им особое внимание.

§ 51 Простейшие свойства графов

Напомним, что **граф** — это конечная совокупность **вершин**, некоторые из которых соединены **ребрами**. Мы будем рассматривать только такие графы, у которых две вершины могут быть соединены только одним ребром. Иногда возникает необходимость рассматривать конфигурации, когда пара вершин соединена несколькими ребрами, — в этом случае говорят, что задан **мультиграф**, а ребра, соединяющие одну и ту же пару вершин, называют **кратными**. Ребро не обязано соединять разные вершины. Если ребро соединяет вершину саму с собой, то такое ребро называют **петлей**.

Две различные вершины графа, соединенные ребром, называют **смежными**. Количество ребер, выходящих из одной вершины, называют **степенью** этой вершины. Для петли будем считать, что это ребро выходит из вершины дважды. Степень вершины a будем обозначать $\gamma(a)$.

Первое свойство, которое мы сформулируем, таково:

> Сумма степеней всех вершин графа равна удвоенному числу его ребер.

Это свойство обосновывается просто: если подсчитывается сумма степеней всех вершин, то каждое ребро в этой сумме фигурирует ровно два раза.

Из этого свойства есть следствие, которое принято называть **леммой о рукопожатиях.** Вот формулировка этой леммы:

> Для любого графа количество вершин нечетной степени всегда четно.

Леммой о рукопожатиях это утверждение называют из-за следующей интерпретации:

> В любой момент времени количество людей, сделавших нечетное число рукопожатий, четно.

Действительно, если вершины графа — это люди, а ребра — это рукопожатия, то видно, что это одно и то же утверждение о получившемся графе.

Вот еще одно свойство:

> В любом графе есть по крайней мере две вершины, имеющие одинаковую степень.

Докажем это свойство. Пусть в графе n вершин. Степень каждой вершины может иметь значение от 0 до $n-1$. Если степени всех вершин различны, то каждое из указанных значений должно реализоваться ровно для одной вершины. Рассмотрим вершины степени 0 и степени $n-1$. Степень 0 означает, что эта вершина не соединена ни с какой другой; степень $n-1$ означает, что эта вершина соединена со всеми другими вершинами. Но одновременно так быть не может. Свойство доказано.

Маршрутом на графе называется последовательность ребер e_1, e_2, ..., e_k, в которой конец одного ребра служит началом следующего. Если при этом конец последнего ребра последовательности совпал с началом первого ребра, то маршрут называется **циклическим**. Для графа, изображенного на рисунке 6.1, последовательности e_1, e_2, e_4, e_5, e_2, e_3 и e_2, e_4, e_5 являются маршрутами, причем второй из них циклический. А последовательность e_1, e_2, e_5 маршрутом не является.

Если вершина является концом какого-либо ребра, принадлежащего маршруту, то будем говорить, что данный маршрут проходит через эту вершину.

Маршрут называется **цепью**, если каждое ребро содержится в нем не более одного раза. Цепь, являющаяся циклическим маршрутом, называется **циклом**. Цепь, проходящая через каждую свою вершину ровно один раз, называется **простой**. Если цикл является простой цепью, то его тоже называют **простым**.

Поскольку в рассматриваемых нами графах нет кратных ребер, то маршрут можно задавать и последовательным перечислением вершин, через которые он проходит. Мы будем иногда этим пользоваться.

Вершины a и b называют **связанными**, если существует цепь, начинающаяся в вершине a и заканчивающаяся в вершине b. Договариваются также считать, что каждая вершина связана сама с собой.

Граф называют **связным**, если любые две его вершины связаны. Если же граф несвязен, то в нем можно выделить так называемые **связные компоненты**, т. е. такие множества вершин, соединенных ребрами исходного графа, каждое из которых является связным графом. Но вершины из разных множеств уже не связаны.

Один и тот же граф может быть изображен по-разному. Например, граф на рисунке 6.2 на самом деле тот же, что и на рисунке 6.1.

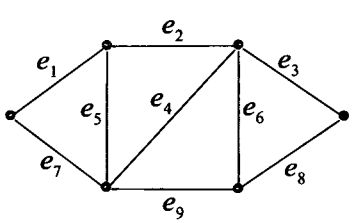


Рис. 6.1

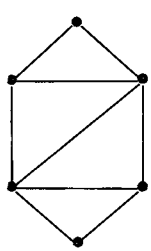
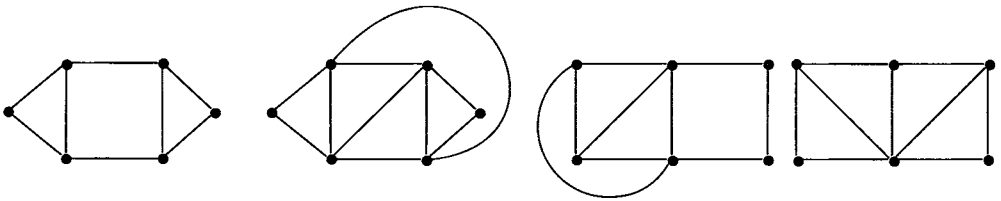


Рис. 6.2

Вопросы и задания

1. Что такое граф?
2. Что называют степенью вершины?
3. Какое наибольшее число ребер может содержать граф, имеющий n вершин?
4. Пусть $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ — множество вершин графа. Для каждого из перечисленных ниже случаев изобразите соответствующий граф:
   а) вершины x и y соединены ребром тогда и только тогда, когда $(x - y)/3$ — целое число;
   б) вершины x и y соединены ребром тогда и только тогда, когда $x + y = 9$;
   в) вершины x и y соединены ребром тогда и только тогда, когда $x + y$ содержится в множестве $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$;
   г) вершины x и y соединены ребром тогда и только тогда, когда $|x - y| < 3$;
   д) вершины x и y соединены ребром тогда и только тогда, когда x и y не взаимно просты.
5. Составьте список степеней вершин для каждого из графов, построенных вами при выполнении задания 4.
6. Существует ли граф с пятью вершинами и следующим набором степеней вершин:
   а) 0, 1, 2, 3, 4; б) 1, 1, 2, 3, 4;
   в) 1, 1, 2, 2, 4; г) 1, 1, 2, 3, 3?
7. Может ли в государстве, в котором из каждого города выходят ровно три дороги, быть ровно сто дорог?
8. — Наша шпионская сеть была хорошо законспирирована, — признался на допросе агент 007. — В ней было 77 агентов, но каждый знал только семерых.
   Почему наверняка можно утверждать, что агент врет?
9. Какие вершины графа называются смежными?
10. Что называется маршрутом на графе?
11. Что такое цепь? Какая цепь называется простой?
12. Что такое цикл? Какой цикл называется простым?
13. Найдите число простых циклов длины 3, 4, 5 и 6 для графов, изображенных на рисунке 6.3.



а) б) в) г)

Рис. 6.3

⓮ Рассмотрите граф на рисунке 6.2. Расставьте на нем обозначения ребер так, чтобы стало ясно, что это тот же граф, что и на рисунке 6.1.

⓯ Выясните, одинаковы ли графы, изображенные на рисунке 6.4; на рисунке 6.5; на рисунке 6.6.

⓰* На конгрессе собрались ученые, среди которых есть друзья. Оказалось, что никакие двое ученых, имеющие равное число друзей, не имеют общих друзей. Докажите, что найдется ученый, который имеет ровно одного друга.

⓱ Какой граф называется связным?

⓲ Укажите, какие из графов, построенных вами при выполнении задания 4, связны. Для несвязных графов найдите число компонент связности.

⓳ Докажите, что если в связном графе удалить ребро, принадлежащее циклу, то граф останется связным.

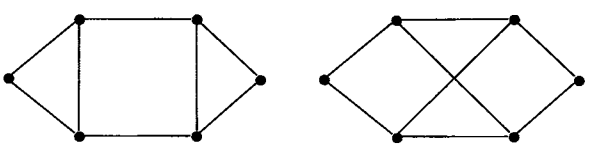


Рис. 6.4

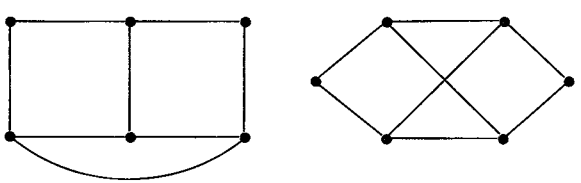


Рис. 6.5

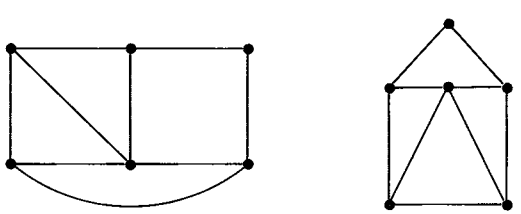


Рис. 6.6

§ 52 Способы представления графов

Изображение графа рисунком удобно для восприятия человеком. Однако, если для решения задачи, связанной с графом, надо применить компьютер, такой способ представления уже малопригоден. Поэтому используют другие способы представления графов.

Граф называется **нагруженным**, если каждому ребру сопоставлено некоторое число. В зависимости от рассматриваемой задачи это число может обозначать расстояние между вершинами, или время перехода от одной вершины к другой (если, например, графом изображена какая-либо транспортная схема), или пропускную способность канала, соединяющего две данные вершины (если в виде графа изображена какая-либо коммуникационная сеть), или еще что-либо. Иногда удобно рассматривать ненагруженный граф как нагруженный, у которого каждому ребру поставлено в соответствие число 1. Поэтому мы обсудим способы представления нагруженных графов.

Обычно граф задают одним из двух способов: перечислением всех его ребер или таблицей, где в клетке на пересечении строки и столбца, соответствующих данным вершинам, указано, соединены эти вершины ребром или нет. Такая таблица называется **таблицей смежности**. Если граф нагруженный, то для каждого ребра в соответствующей клетке указывается нагрузка. Приведем список ребер для нагруженного графа, изображенного на рисунке 6.7: (*AA*; 2), (*AB*; 3), (*AC*; 6), (*BC*; 2), (*AD*; 4), (*BD*; 3), (*CD*; 5). Таблица 6.1 является таблицей смежности для этого графа.

В таблице смежности ненагруженного графа везде вместо чисел, указывающих нагрузку (т. е. отличных от 0), стояло бы число 1. А в списке ребер ненагруженного графа просто не нужна числовая характеристика.

Надо уметь переходить от одного способа описания графа к другому. Но эта работа совершенно формальна и, следовательно, может быть поручена компьютеру, только нужно составить соответствующий алгоритм.

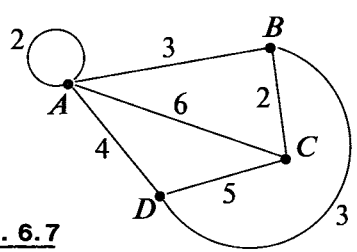


Рис. 6.7

Таблица 6.1

Таблица смежности

Вершина	*A*	*B*	*C*	*D*
A	2	3	6	4
B	3	0	2	3
C	6	2	0	5
D	4	3	5	0

Таблица 6.2

1	1	1	1	2	2	3
1	2	3	4	3	4	4
2	3	6	4	2	3	5

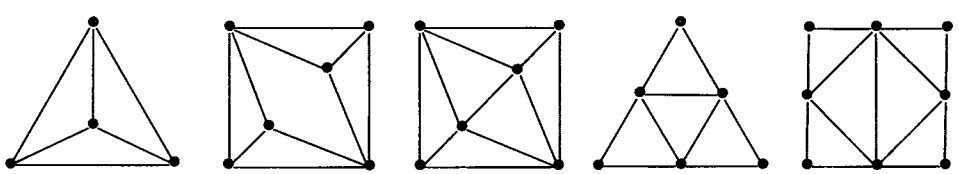
Таблица 6.3

Вершина	1	2	3	4
1	2	3	6	4
2	3	0	2	3
3	6	2	0	5
4	4	3	5	0

В дальнейшем в заданиях на составление алгоритма, тем или иным образом обрабатывающего граф, мы для простоты будем считать вершины графа перенумерованными натуральными числами от 1 до n (без пропусков и повторений). Список ребер для нагруженного графа будем задавать как двумерный массив $A[1 : 3; 1 : n]$, где в первой строке соответствующей этому массиву таблицы указывается один конец ребра, во второй — другой его конец, а в третьей — величина нагрузки (здесь n — число ребер в графе). Для ненагруженного графа соответствующий массив содержит только первые две строки. Если граф задается таблицей смежности, то договоримся считать значение первого индекса номером первой вершины, а второго индекса — номером второй вершины; сами номера вершин в массиве не присутствуют. В частности, для графа на рисунке 6.7 при естественной нумерации вершин A — 1, B — 2, C — 3 и D — 4 список ребер в силу нашей договоренности задается массивом, который можно изобразить таблицей 6.2, а таблица смежности имеет вид таблицы 6.3.

Вопросы и задания

1. Что такое таблица смежности?
2. Для каждого из графов, изображенных на рисунке 6.8, запишите его представление списком ребер и таблицей смежности.



а) б) в) г) д)

Рис. 6.8

3 Изобразите графы, для которых в таблицах 6.4, 6.5, 6.6 заданы таблицы смежности.

Таблица 6.4

	A	*B*	*C*	*D*	*E*	*F*	*G*
A	0	0	0	0	1	1	1
B	0	0	0	0	1	1	0
C	0	0	0	0	1	0	1
D	0	0	0	0	0	1	1
E	1	1	1	0	0	1	1
F	1	1	0	1	1	0	1
G	1	0	1	1	1	1	0

Таблица 6.5

	A	*B*	*C*	*D*	*E*	*F*	*G*
A	0	0	1	0	1	1	0
B	0	0	1	0	0	0	1
C	1	1	0	1	0	1	0
D	0	0	1	0	0	0	0
E	1	0	0	0	0	0	1
F	1	0	1	0	0	0	1
G	0	1	0	0	1	1	0

Таблица 6.6

	A	*B*	*C*	*D*	*E*	*F*	*G*
A	0	0	0	1	1	1	0
B	0	0	1	1	1	0	1
C	0	1	0	1	0	0	1
D	1	1	1	0	0	0	0
E	1	1	0	0	0	1	1
F	1	0	0	0	1	0	0
G	0	1	1	0	1	0	0

4 а) Граф, имеющий *n* вершин, задан списком ребер. Составьте алгоритм, создающий по этому списку таблицу смежности.

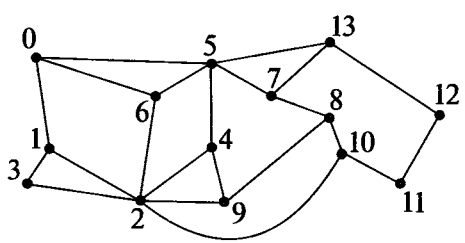
б) Граф, имеющий *n* вершин, задан таблицей смежности. Составьте алгоритм, создающий по этой таблице список ребер.

§ 53 Алгоритмы обхода связного графа

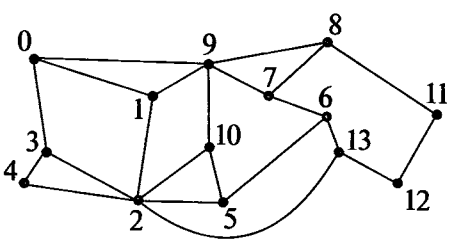
Пусть имеется связный граф. Это означает, что из любой вершины можно, двигаясь по ребрам, добраться до любой другой его вершины. Скажем сразу: алгоритма, позволяющего по двум заданным вершинам построить путь из одной вершины в другую, нет. Но можно указать алгоритм, позволяющий из заданной вершины совершить обход всех остальных вершин и, значит, заведомо добраться до нужной вершины. Таких алгоритмов существует несколько, мы рассмотрим два из них — наиболее популярных.

Первый из них называется **поиском в глубину**. Идея алгоритма такова. Пусть зафиксирована начальная вершина v_0. Выберем смежную с ней вершину v_1. Затем для вершины v_1 выбираем смежную с ней вершину из числа еще не выбранных вершин и т. д.: если мы уже выбрали вершины v_0, v_1, ... , v_k, то следующая вершина выбирается смежной с вершиной v_k из числа невыбранных. Если для вершины v_k такой вершины не нашлось, то возвращаемся к вершине v_{k-1} и для нее ищем смежную среди невыбранных. При необходимости возвращаемся еще на шаг назад и т. д. Ясно, что так будут перебраны все вершины графа и поиск закончится. На рисунке 6.9 показаны две реализации поиска в глубину для одного и того же графа (при одинаковом выборе начальной вершины): около каждой вершины написан присвоенный ей порядковый номер при исполнении поиска в глубину. Свое название этот метод получил за то, что при его реализации мы стремимся как можно дальше уйти от исходной вершины, а когда идти уже некуда, возвращаемся в ту вершину, откуда идет хотя бы одно ребро в не пройденные еще вершины.

Однако от идеи до алгоритма путь неблизкий: надо договориться, как задан граф, как помечать вершины, которые уже про-



а)



б)

Рис. 6.9 Два варианта применения поиска в глубину

смотрены, и как из нескольких вершин, смежных с данной, выбрать следующую.

Будем считать, что граф задан таблицей смежности. Кроме того, организуем одномерный массив B, число элементов в котором совпадает с числом вершин в графе. На k-м месте этого массива будем писать номер вершины, в которую мы попали на k-м шаге. Из всех смежных вершин будем выбирать вершину с наименьшим номером.

Вот как может выглядеть алгоритм, реализующий поиск в глубину:

```
Алгоритм Поиск в глубину
цел: k, m, s, t, u, n, a, v, i, G[1:n; 1:n], B[1:n];
{ Запросить n;                    (*запрашивается количество вершин*)
  Запросить G[1:n; 1:n]; (* реально это действие — ввод таблицы смежности — оформляется двойным циклом *)
  B[1] :=1;    (* в качестве исходной взята вершина с номером 1 *)
  Делать от k := 2 до n
  {  B[k] := 0;
  }
  k := 1;                  (*счетчик количества пройденных вершин*)
  m := 0;                        (*k – m — порядковый номер вершины
                                  для очередного шага поиска*)
  Делать пока (k < n)
  {  s := 1;
    Делать от t := 1 до n
    {  v := 1;
      Делать от i := 1 до k
      {  Если (B[i] = t) то { v := 0; } }
      Если (v = 1 и G(B[k – m], t) = 1) то
      { a := t;
        s := 0;
      }
    }
(*если существует смежная вершина, которая еще не просматривалась, то s = 0*)
    Если (s = 0) то
    {   k := k + 1;
        B[k] := a;
        m := 0;
    }
    иначе  { m := m + 1; }
  }
  Делать от k := 1 до n
  { Сообщить B[k], k;
  }
}
```

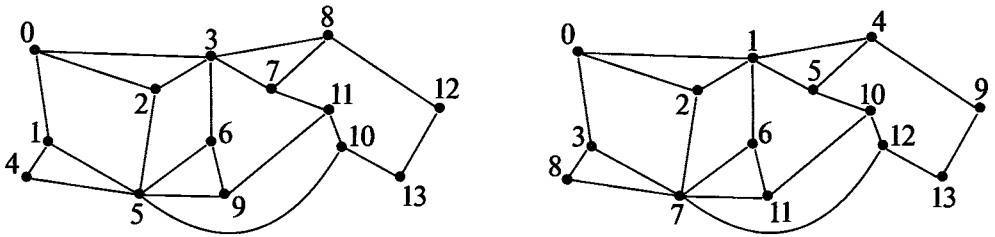


Рис. 6.10 Два варианта применения поиска в ширину

Второй алгоритм называется **поиском в ширину**. Суть этого подхода состоит в том, чтобы рассматривать все вершины, смежные с уже рассмотренными. На рисунке 6.10 показаны две реализации поиска в ширину для одного и того же графа (при одинаковом выборе начальной вершины).

Как и для поиска в глубину, прежде чем написать алгоритм, надо ответить на те же три вопроса: каким способом представлен граф? Как помечать просмотренные вершины? Как выбирать очередную вершину из нескольких еще не просмотренных, но смежных с уже просмотренными? Ответы на эти вопросы дадим такие же, что и при составлении алгоритма поиска в глубину.

А теперь сам алгоритм:

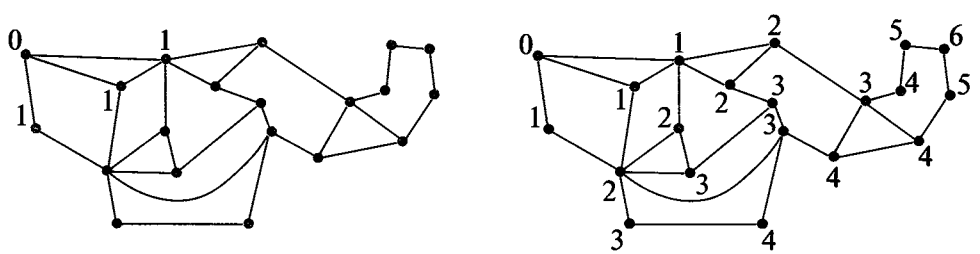
```
Алгоритм Поиск в ширину
цел: k, m, s, t, n, G[1:n; 1:n], B[1:n];
{ Запросить n;
  Запросить G[1:n; 1:n];   (*реально это действие — ввод таблицы смежности — оформляется двойным циклом*)
  B(1) :=1;   (* в качестве исходной взята вершина с номером 1 *)
  Делать от k := 2 до n
  { B[k] := 0;
  }
  s := 1;
  Делать от k := 2 до n
  { t := 1;
    m := −1;
    Делать пока (G[s, t] = 0 или t = s или B[t] ≠ 0)
    { t := t + 1;
    }
    Если (t < n + 1) то
    { B[t] := k;
      m := m + 1;
    }
```

```
        иначе
        {  s := k − m;
        }
    }
    Делать от k := 1 до n
    {  Сообщить B[k], k;
    }
}
```

А сейчас рассмотрим одну из часто встречающихся задач: найти длину кратчайшей цепи от заданной вершины до любой другой. **Длиной цепи** при этом называют количество содержащихся в ней ребер.

Идея решения этой задачи состоит в следующем. Исходной вершине припишем число 0. Каждой смежной с ней вершине припишем число 1. Каждой вершине, смежной с той, которая уже помечена числом 1 и не была помечена раньше, приписываем число 2. Каждой вершине, смежной с той, которая уже помечена числом 2 и не была помечена раньше, приписываем число 3. И так далее до тех пор, пока такое действие можно будет производить. При этом могут оказаться вершины, добраться до которых так и не удастся. На рисунке 6.11 приведен результат обработки указанным образом конкретного графа.

Легко понять, что число, помечающее вершину, показывает длину кратчайшей цепи, ведущей к этой вершине от заданной. Описанный метод называется **волновым алгоритмом**. Запишем волновой алгоритм, считая, что граф задан таблицей смежности. Предположим для простоты, что в графе 20 вершин, а таблица смежности представлена целочисленным массивом $G[1:20;\ 1:20]$. Результатом работы алгоритма является одномерный целочисленный массив $P[1:20]$, для которого каждое значение элемента $P[k]$



а) б)

Рис. 6.11 Результат обработки графа волновым алгоритмом:
а) после первого шага; б) полностью обработанный

равно длине пути от заданной вершины до вершины с номером k. Договоримся элементу $P[k]$ присваивать значение -1, если вершина с номером k недостижима.

```
Алгоритм Кратчайший_путь
цел: I, J, A, K, G[1:20; 1:20], P[1:20];
{ Запросить G[1:20; 1:20];
  Запросить A;      (* запрашивается номер начальной вершины *)
  Делать от I := 1 до 20
  {  P[I[ := -1;
  }          (* сначала все элементы массива-результатов равны -1*)
                    (*до исходной вершины добираемся за 0 шагов*)
  P[A] := 0;
  Делать от I := 0 до 19
  { Делать от K := 1 до 20
    { Если P[K] = I то
      { Делать от J := 1 до 20
        { Если (P[J] = -1 и G[J, K] = 1) то
          { P[J] := I + 1;
          }
        }
      }
    }
  }
  Делать от I := 1 до 20
  { Сообщить "Кратчайший путь от вершины", A;
    Сообщить "до вершины", I;
    Сообщить "равен", P[I];
  }
}
```

Вопросы и задания

1. Исполните алгоритм Поиск в глубину для графов, заданных таблицами 6.4—6.6.

2. Модифицируйте алгоритм Поиск в глубину так, чтобы номер вершины, с которой начинается поиск, запрашивался у пользователя.

3. а) Объясните, почему приведенный алгоритм Поиск в глубину не может применяться к несвязным графам. Какое из свойств алгоритмов будет нарушаться при таком применении?

   б) Модифицируйте алгоритм Поиск в глубину так, чтобы он стал применимым и к несвязному графу.

4. Составьте алгоритм, реализующий Поиск в глубину, для графов, заданных списком ребер.

5. Исполните алгоритм Поиск в ширину для графов, заданных таблицами 6.4—6.6.

6. Каков смысл переменной *m* в алгоритме Поиск в ширину?

7. Модифицируйте алгоритм Поиск в ширину так, чтобы номер вершины, с которой начинается поиск, запрашивался у пользователя.

8. Составьте алгоритм, реализующий Поиск в ширину, для графов, заданных списком ребер.

9. Исполните волновой алгоритм для графов, заданных таблицами 6.4—6.6.

10. Граф, изображенный на рисунке 6.11, имеет в точности 20 вершин. Из представленного на рисунке 6.11, *б* результата работы алгоритма, приведенного в объяснительном тексте, видно, что при значениях параметра *l*, больших чем 6 (а таких значений 14), цикл будет работать вхолостую — ведь новых чисел при вершинах появиться не может. Модифицируйте алгоритм так, чтобы цикл не исполнялся сверх нужного числа раз.

11. Граф задан списком ребер. Составьте алгоритм поиска кратчайших цепей от заданной вершины, аналогичный разобранному в объяснительном тексте.

12. Составьте алгоритм, с помощью которого для заданной вершины можно определить компоненту связности, которой эта вершина принадлежит. Рассмотрите два варианта задания графа — списком ребер и таблицей смежности. Компоненту связности в первом случае опишите списком входящих в нее ребер, во втором — списком входящих в нее вершин.

13. Составьте алгоритм, с помощью которого можно найти все компоненты связности для заданного графа. Каждую компоненту связности опишите списком входящих в нее вершин. Рассмотрите два варианта задания графа — списком ребер и таблицей смежности.

14. Если граф имеет *k* компонент связности, то достаточно добавить *k*—1 ребер, чтобы превратить его в связный граф (меньшего числа ребер недостаточно). Составьте алгоритм, который позволяет превратить граф, имеющий *k* компонент связности, в связный граф добавлением наименьшего числа ребер. Ясно, что выбор вершин из разных компонент связности для задания ребра между ними можно осуществить неоднозначно. Предусмотрите в своем алгоритме использование датчика случайных чисел для выбора таких вершин.
Рассмотрите два варианта задания исходного графа — списком ребер и таблицей смежности. Новый граф должен быть задан тем же способом, что и исходный.

15* У путешественника есть карта, на которой отмечены города и дороги между некоторыми из них. Все дороги допускают двустороннее движение, и для каждой дороги, соединяющей два города, указана ее длина. Путешественник хочет из одного города добраться в другой кратчайшим путем. Предложите алгоритм, позволяющий найти нужный путешественнику маршрут или сообщить, что такого маршрута нет.

16* Квадратная вселенная. В некоторой вселенной, имеющей форму квадрата со стороной 100, расположено несколько точечных планет. Каждая планета отстоит от сторон вселенной и от других планет на целочисленные расстояния. Требуется организовать межпланетную экспедицию, которая побывает на всех планетах. Между планетами космолет движется по прямой. Составьте алгоритм, позволяющий найти кратчайший маршрут для такой экспедиции. Начальный и конечный пункты экспедиции совпадать не обязаны.

§ 54 Мосты и точки сочленения

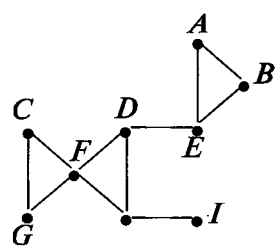
Пусть имеется какая-нибудь система связи, например компьютерная сеть. Такую сеть естественно представлять графом, в котором узлы связи — это вершины графа, а линии связи — его ребра. Такой граф должен быть связным, чтобы информация из любого узла связи могла быть передана в любой другой. И весьма желательно, чтобы при выходе из строя какого-либо узла связи или линии сохранилась бы возможность передачи информации из одного узла связи в любой другой. Иными словами, граф остался бы связным, если из него удалить некоторую вершину вместе с входящими в него ребрами или если удалить из него ребро.

Вершина связного графа называется **точкой сочленения**, если после ее удаления из графа (вместе с входящими в нее ребрами) граф перестает быть связным.

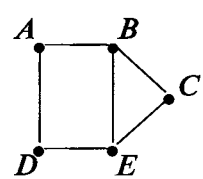
Ребро связного графа называется **мостом**, если после его удаления граф перестает быть связным.

Связный граф называется **двусвязным**, если он не имеет точек сочленения.

Для графа, изображенного на рисунке 6.12, *а*, вершины D, E, F и H — точки сочленения, а ребра DE и HI — мосты. Граф, изображенный на рисунке 6.12, *б*, двусвязен.



а)



б)

Рис. 6.12 Примеры графов

Вопросы и задания

1. Что такое точка сочленения?

2. Какое ребро связного графа называют мостом?

3. Какой граф называют двусвязным?

4. а) Проверьте, что граф, изображенный на рисунке 6.12, *а*, действительно имеет точками сочленения те и только те вершины, которые перечислены в объяснительном тексте параграфа.

   б) Проверьте, что граф, изображенный на рисунке 6.12, *а*, действительно имеет мостами те и только те ребра, которые перечислены в объяснительном тексте параграфа.

   в) Проверьте, что граф, изображенный на рисунке 6.12, *б*, действительно двусвязен.

5. Укажите мосты и точки сочленения для графов, представленных на рисунке 6.13.

Рис. 6.13

6. а) Дан связный граф, содержащий не менее трех вершин. Известно, что в нем имеется мост. Можно ли утверждать, что этот граф содержит точку сочленения?

   б) Дан связный граф, содержащий не менее трех вершин. Известно, что в нем имеется точка сочленения. Можно ли утверждать, что этот граф содержит мост?

7. Составьте алгоритм, с помощью которого можно найти все точки сочленения заданного графа. Рассмотрите два варианта задания графа — списком ребер и таблицей смежности.

8. Составьте алгоритм, с помощью которого можно найти все мосты заданного графа. Рассмотрите два варианта задания графа — списком ребер и таблицей смежности.

9. а) Докажите, что вершина v является точкой сочленения в связном графе тогда и только тогда, когда найдутся такие две вершины a и b, для которых любая цепь, соединяющая эти вершины, проходит через вершину v.

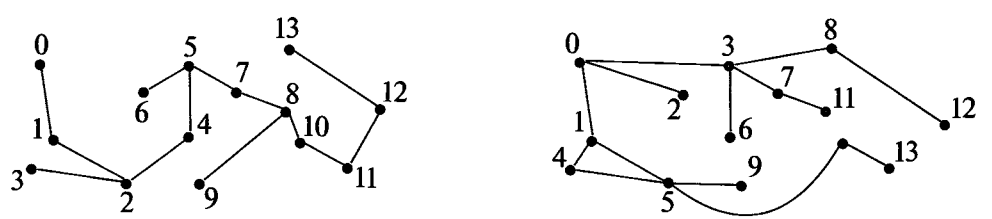
   б) Докажите, что ребро связного графа является мостом тогда и только тогда, когда оно не принадлежит никакому циклу этого графа.

§ 55 Деревья

В теории графов **деревом** называется связный граф без циклов. Но взгляните на любое дерево за окном: если точки, где ветви соединяются, принять за вершины графа, то получится именно граф без циклов.

Деревья — это те графы, с которыми вы в курсе информатики, да и других предметов, чаще всего имеете дело. Дерево каталогов и генеалогическое дерево, да и вообще любая иерархическая система с точки зрения своей структуры представляют собой именно дерево. Применяя алгоритмы поиска в глубину или в ширину, вы из исходного графа извлекаете дерево — вершинами в нем являются вершины исходного графа, и эти вершины соединяются ребром, если при исполнении алгоритма переход от одной вершины к следующей осуществлялся по этому ребру. На рисунке 6.14, *а* представлено дерево, полученное применением поиска в глубину в соответствии с рисунком 6.7, *а*; на рисунке 6.14, *б* представлено дерево, полученное применением поиска в ширину в соответствии с рисунком 6.10, *а*. Дерево, которое содержит все вершины некоторого заданного графа G и ребра которого являются ребрами этого графа, называется **каркасом** графа G. Так что можно сказать, что на рисунке 6.14 представлены два каркаса одного и того же графа. По-другому каркас называют **остовом** или **стягивающим деревом**.

Дерево, как правило, изображают некоторым стандартным образом. Для этого фиксируют одну из вершин, ее называют **корнем**. Корень обычно изображают внизу, а все остальные вершины распределяют по уровням. На первом уровне размещаются вершины, смежные с корнем, на втором — смежные с вершинами первого уровня, отличные от корня, на третьем — смежные с вершинами второго уровня, отличные от вершин первого уровня, и т. д.



а) б)

Рис. 6.14 Деревья, получаемые при обходе графа

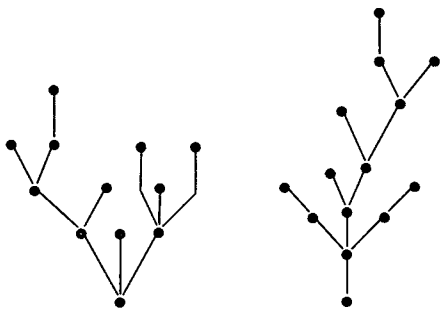


Рис. 6.15 Одно и то же дерево при разном выборе корня

На рисунке 6.15 приведены два изображения одного и того же дерева, взятого с рисунка 6.14, *б*, при разном выборе корневой вершины: в первом случае корнем служит вершина, обозначенная числом 0, во втором — обозначенная числом 6. Впрочем, нередко бывает удобнее рисовать дерево сверху вниз или слева направо.

Выбор корня фактически превращает дерево в ориентированный граф: на каждом ребре направление выбирается от меньшего уровня к большему. Именно такой граф фигурировал у нас при рассмотрении алгоритма Хаффмана в § 18. Там же вершины степени 1, отличные от корня, были названы листьями. Легко видеть, что, удаляя лист вместе с ведущим в него ребром, мы одновременно уменьшаем на 1 количество вершин и количество листьев. Проделав это столько раз, сколько ребер в дереве, мы останемся один на один с корнем. Следовательно, в любом дереве вершин всегда на 1 больше, чем ребер. На самом деле справедливо и обратное утверждение: если в связном графе количество вершин на 1 больше числа ребер, то это — дерево. В задании 7 мы предлагаем вам доказать это утверждение самостоятельно.

Вопросы и задания

1. Какой граф называют деревом?
2. Как связаны количества ребер и вершин в дереве?
3. Для графа, изображенного на рисунке 6.14, *а*, нарисуйте дерево, взяв в качестве корня вершину, обозначенную: а) числом 5; б) числом 10.
4. Изобразите все деревья с четырьмя и пятью вершинами.
5. а) Постройте дерево, полученное применением поиска в глубину для графа, изображенного на рисунке 6.9, *б*.
   б) Постройте дерево, полученное применением поиска в ширину для графа, изображенного на рисунке 6.10, *б*.
6. * Докажите, что связный граф является деревом тогда и только тогда, когда любое его ребро является мостом. (Совет. Воспользуйтесь утверждением, сформулированным в задании 9б к § 54.)

7* Докажите, что связный граф, в котором количество ребер на 1 меньше числа вершин, является деревом. (Совет. Воспользуйтесь утверждением, сформулированным в задании 19 к § 51.)

8 Составьте алгоритм, позволяющий построить каркас с использованием поиска в глубину. Рассмотрите два варианта задания графа — списком ребер и таблицей смежности.

9 Составьте алгоритм, позволяющий построить каркас с использованием поиска в ширину. Рассмотрите два варианта задания графа — списком ребер и таблицей смежности.

§ 56 Каркасы минимального веса

Представьте себе, что надо соединить несколько пунктов линиями связи. Известна стоимость возможного строительства линии между парами таких пунктов. Какие именно пункты надо соединить, чтобы получившаяся сеть обслуживала все пункты и при этом имела минимальную стоимость?

Переводя эту задачу на язык графов, можно сказать так.

Имеется нагруженный связный граф. Требуется найти каркас с минимальным суммарным весом его ребер.

Таких каркасов у данного нагруженного связного графа может быть несколько. На рисунке 6.16, *б* и *в* приведены два каркаса минимального веса для графа, изображенного на рисунке 6.16, *а*. Разумеется, вес обоих минимальных каркасов одинаков.

Опишем метод построения хотя бы одного каркаса минимального веса, известный как **алгоритм Краскала**. Будем считать, что исходный связный граф G задан списком ребер. Получающийся каркас тоже будет задан списком ребер. Алгоритм Краскала предусматривает построение двух последовательностей множеств ребер исходного графа: T_1, T_2, T_3, ... и E_1, E_2, E_3, ..., при этом на

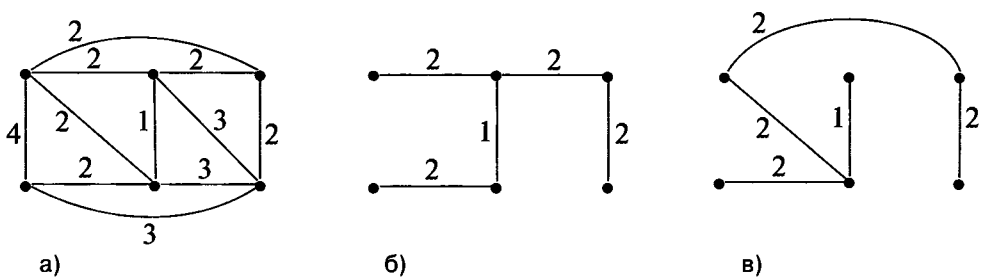


Рис. 6.16 Граф и два каркаса для него

некотором шаге одно из множеств T_k оказывается каркасом минимального веса, после чего исполнение алгоритма завершается.

На первом шаге выбираем ребро e_1 наименьшего веса (если таких ребер несколько, то берем любое из них) и полагаем $T_1 = \{e_1\}$. В качестве множества E_1 строится множество ребер, каждое из которых не содержится в T_1 и при добавлении к T_1 не образует цикл.

Пусть уже построены множества T_1, T_2, ..., T_k и E_1, E_2, ..., E_k. Если множество E_k не содержит ребер, то в качестве искомого каркаса берем множество T_k. Если же множество E_k не пусто, то строим T_{k+1} и E_{k+1} по следующему правилу: в множестве E_k выбираем ребро наименьшего веса (если таких ребер несколько, то снова берем любое из них) и добавляем его в множество T_k, это и будет множество T_{k+1}; множество E_{k+1} состоит из таких ребер, что каждое из них не содержится в T_{k+1} и при добавлении любого из них к множеству T_{k+1} не образуется цикл. Такое построение последовательности множеств T_k и E_k повторяется, пока множество E_k не станет пустым.

Алгоритм, реализующий эту идею, вы составите самостоятельно, выполнив задание 5. Но нельзя не задаться вопросами:

1. Почему соответствующий алгоритм конечен?
2. Почему соответствующий алгоритм результативен?
3. Почему результатом является требуемый каркас?

Конечность алгоритма обеспечивается тем, что количество ребер, остающихся вне множества T_k, на каждом шаге уменьшается на 1. Следовательно, и в множество E_k в некоторый момент нельзя будет добавить ни одного ребра.

Предложенный алгоритм не является детерминированным, поэтому надо убедиться, что после его завершения образуется каркас исходного графа. Пусть исполнение алгоритма завершилось построением графа T_k, т. е. множество E_k не содержит ребер. По построению в T_k нет циклов. Если T_k не содержит хотя бы одну вершину исходного графа G, то ребро, выходящее из этой вершины, не принадлежит T_k, а при включении его в T_k не может образоваться цикл. Тогда это ребро обязано было попасть в E_k, что противоречит тому, что множество E_k пусто. Значит, T_k содержит все вершины графа G. Осталось показать, что T_k связен. Допустим снова, что это не так. Тогда T_k содержит по крайней мере две компоненты связности, скажем, A и B. Поскольку исходный граф G связен, существуют смежные вершины a из A и b из B. Пусть e — соединяющее их ребро. Тогда e не принадлежит T_k. Более того, по определению e является мостом исходного графа G. Значит, в графе, полученном из T_k добавлением этого ребра, не может образоваться

цикл, ибо иначе цикл, содержащий ребро e, был бы и в самом графе G. Снова получаем, что ребро e должно было бы попасть в множество E_k, что противоречит отсутствию в E_k элементов. Это противоречие показывает, что на самом деле граф T_k связен и, следовательно, T_k — дерево, содержащее все вершины графа G, т. е. T_k — каркас графа G.

Осталось ответить на третий вопрос: почему вес у построенного каркаса минимален?

Допустим, что это не так. Выберем каркас T, имеющий наименьший суммарный вес. Как мы уже говорили, таких каркасов с наименьшим весом может оказаться несколько. Выберем среди них такой, у которого наибольшее количество общих с каркасом T_k ребер. Будем по-прежнему обозначать этот минимальный каркас буквой T.

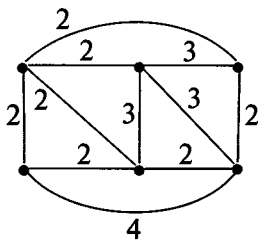
Можно считать, что в каркасе T_k все ребра перенумерованы в порядке поступления их в T_k. Ясно, что T_k не может целиком содержаться в T, ибо иначе эти два каркаса просто бы совпали — ведь у них одно и то же множество вершин. Пусть e_m — ребро с наименьшим номером, которое содержится в T_k, но не содержится в T. Добавим это ребро к графу T. В получившемся связном графе F, очевидно, образовался некоторый цикл C. В этом цикле есть ребро e, не принадлежащее каркасу T_k, ибо в T_k вообще нет циклов. Рассмотрим граф T', полученный из графа T изъятием ребра e и добавлением ребра e_m. Граф T' связен, поскольку получен из связного графа F удалением ребра, принадлежащего циклу (см. задание 19 к § 51). Граф T' является деревом, поскольку в нем столько же ребер и вершин, сколько в дереве T, т. е. вершин на 1 больше, чем ребер (см. задание 7 к § 55). Наконец, все вершины графа G принадлежат графу T', поскольку у T и T' одно и то же множество вершин. Тем самым T' тоже каркас графа G.

Сравним вес ребер e и e_m. Поскольку ребра e_1, e_2, ..., e_{m-1} принадлежат T, добавление ребра e к множеству T_{m-1} не создает цикла в получающемся графе (ибо T — дерево). Правило выбора e_m показывает, что вес e не меньше веса e_m (иначе в T_m было бы включено другое ребро, нежели e_m). Данное сравнение весов показывает, что вес графа T' не больше веса графа T. Но каркас T имеет наименьший вес среди всех каркасов графа G. Значит, вес каркаса T' совпадает с весом каркаса T, т. е. тоже является наименьшим. Однако у каркаса T' количество общих ребер с каркасом T_k на единицу больше, чем у каркаса T, что противоречит выбору T. Полученное противоречие показывает, что каркас T_k сам является каркасом минимального веса.

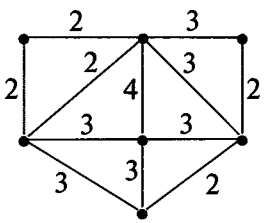
Теперь можно приступать к выполнению заданий.

Вопросы и задания

1. Какая функция выступает лимитирующей в алгоритме Краскала?
2. Используя метод Краскала, найдите каркас минимального веса для графов, представленных на рисунке 6.17.



а)



б)

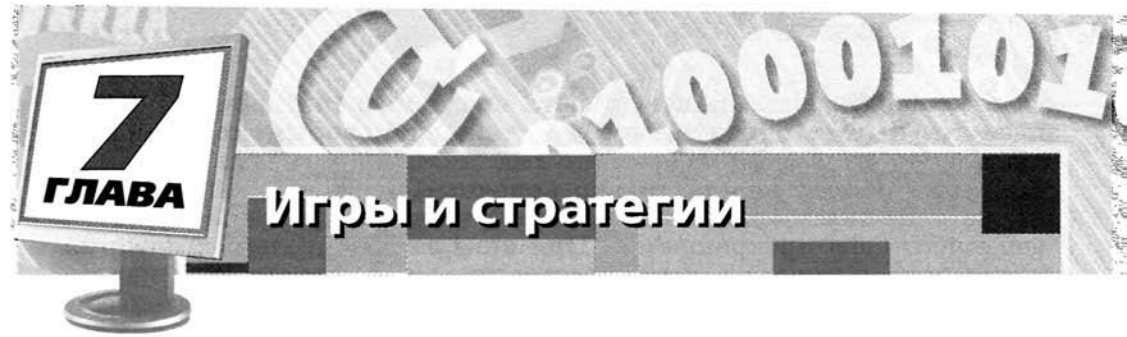
Рис. 6.17

3* Алгоритм Краскала, как отмечалось в объяснительном тексте, не является однозначным. Рассмотрим множество всех каркасов минимального веса, которые могут получиться в результате применения этого алгоритма к заданному графу G. Верно ли, что это множество содержит все каркасы минимального веса, имеющиеся в графе G?

4. Как нужно изменить алгоритм Краскала, чтобы его можно было использовать для построения каркаса максимального суммарного веса?
5. Запишите алгоритм, реализующий метод Краскала. Для этого определите, как именно будет выбираться очередное ребро для построения множества T_m из множества T_{m-1}.
6. Составьте алгоритм, реализующий метод Краскала, если граф задан таблицей смежности.
7. **Алгоритм Прима**. В этом методе поиска каркаса минимального веса при построении очередного множества E_k в него включаются только те ребра, которые не содержатся в T_k, не образуют цикл при добавлении к T_k и имеют общую вершину хотя бы с одним ребром из T_k. Постройте этим методом каркасы минимального веса для графа, изображенного на рисунке 6.17, *б*.

8* Попытайтесь обосновать, что алгоритм Прима, описанный в задании 7, всегда дает каркас минимального веса.

9. Составьте алгоритм, реализующий метод Прима. Рассмотрите два варианта задания исходного графа — списком ребер и таблицей смежности.



Наука и искусство ведения борьбы по определенным правилам — так истолковывается значение слова **стратегия** в «Словаре иностранных слов». Искусство, будучи неформализуемым видом человеческой деятельности, мы детально рассматривать не будем, хотя это, разумеется, тоже информационная деятельность. А вот наука, цель которой, как мы уже неоднократно говорили, — построение моделей на подходящем формализованном языке, представляет для нас особый интерес.

Всякая борьба предполагает взаимодействие по крайней мере двух сторон. Иногда мы говорим, что зима борется с летом или ветер с дождем, но нас будет интересовать тот случай, когда одной из борющихся сторон является управляемый формальный исполнитель. Другая сторона при этом может быть представлена стихией, т. е. быть неуправляемой, или тоже являться какой-либо совокупностью управляемых исполнителей. Типичный пример борьбы управляемых исполнителей — военные действия двух или большего числа противников. Надо честно признать, что своим происхождением само слово «стратегия» (в переводе с древнегреческого — веду войска) обязано именно военной деятельности.

В любой борьбе всегда преследуются какие-то цели — иногда хорошо осознанные, иногда нет. Вопрос о том, как достичь этих целей, — типичная жизненная (т. е. плохо поставленная) задача. В большинстве случаев нечетко определены силы противников (исходные данные), не вполне ясен результат, который нужно получить, и уж совсем неведомы связи между исходными данными и результатом. Формализация этой жизненной задачи, как обычно, сводится к построению соответствующей модели. Такую модель называют **игрой**.

§ 57 Дерево игры

В любой игре всегда точно определены исходные позиции (в том числе игровой материал), результат игры (что именно считать выигрышем, а что поражением), какие действия можно совершать в ходе игры каждому из игроков. Всякая игра состоит из последовательности ходов, поочередно совершаемых каждым

из игроков. Правилами может быть предусмотрен «пропуск хода», но тогда можно считать, что это тоже некоторый ход данного игрока, не меняющий сложившуюся к этому моменту позицию. Мы будем рассматривать только **конечные игры**, т. е. игры, которые за конечное число шагов приводят к заключительной позиции, после чего игра прекращается. При этом заключительных позиций может быть несколько, а количество шагов, приводящих к какой-либо заключительной позиции, вполне может оказаться неизвестным заранее.

Игры могут быть разными: игра в теннис отличается от игры в шахматы, хотя в обеих играх взаимодействуют два игрока, есть строгие правила чередования ходов, совершенно ясно, как определяется выигрыш того или иного игрока. В чем же отличие?

Перед каждым ходом игрок должен принять решение, какое действие он предпримет. Если игрок всегда точно знает, к какой позиции приведет выбранный им ход, то такая игра называется **игрой с полной информацией**. К играм с полной информацией как раз и относятся шахматы, шашки, Го, крестики-нолики и т. п. А домино и преферанс — игры с неполной информацией. Про теннис уж и говорить нечего — в ход игры может вмешаться даже природа. Наше обсуждение годится для игр как с полной, так и с неполной информацией, но мы в основном будем рассматривать игры с полной информацией.

Начнем с простой игры. Имеется кучка из n камней, где $n > 3$. Первый игрок своим первым ходом делит эту кучку на две так, чтобы в каждой было не менее двух камней. Затем ход делает второй игрок, потом снова первый и т. д. Своим ходом игрок выбирает произвольную кучку, содержащую не менее четырех камней, и делит ее на две части так, чтобы в каждой было не менее двух камней. Проигрывает игрок, который не может сделать очередной ход.

Ясно, что мы сейчас описали не одну игру, а много игр: для каждого конкретного n будет свой вариант игры. Пусть для примера $n = 9$. Проанализируем, как может проходить игра.

После хода первого игрока может получиться одна из трех позиций: (7; 2), (6; 3) или (5; 4). Изобразим это графом (рис. 7.1). Теперь для каждой позиции рассмотрим, что получится после хода второго игрока. Результат представлен на рисунке 7.2.

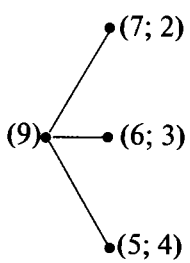


Рис. 7.1

Из позиции (3; 3; 3) дальнейший ход невозможен, т. е. в этой позиции выигрывает второй игрок. В остальных случаях возможно продолжение. Результат представлен на рисунке 7.3. Позиции, которые возникли в этих вариантах, заключительные. В них выигрывает первый игрок.

Построенное дерево вариантов называется **деревом игры**. Для любой игры с полной информацией существует дерево игры.

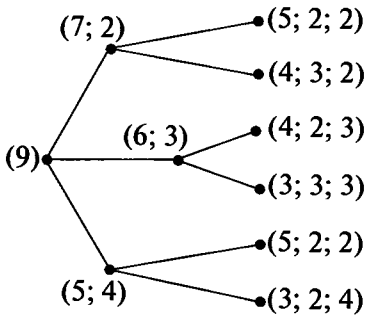


Рис. 7.2

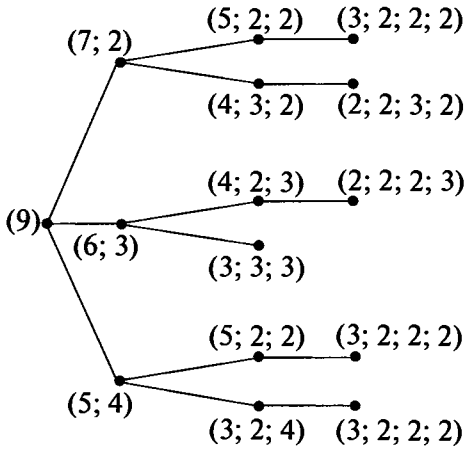


Рис. 7.3

Анализ дерева игры показывает, что у первого игрока есть выигрышная стратегия: если первым ходом он разбивает кучку на две любым из двух способов — (7; 2) или (5; 4), то при любом ходе второго игрока он выигрывает.

Выигрышной стратегией для данного игрока называется такое правило совершения ходов этим игроком, при соблюдении которого он добивается выигрыша при любых ответных ходах другого игрока (или других игроков, если их несколько).

Для тех игр, где возможна ничья, цель игры может состоять в том, чтобы не проиграть противнику. Тогда естественно говорить о непроигрышной стратегии.

Вопросы и задания

1. Почему игру можно считать моделью борьбы противостоящих сторон?
2. Чем характеризуется любая игра?
3. Какая игра называется конечной?
4. Являются ли шахматы и шашки конечными играми?
5. Какую игру называют игрой с полной информацией?
6. Что такое дерево игры?
7. Что называют выигрышной стратегией?
8. Сформулируйте определение непроигрышной стратегии.
9. Постройте дерево игры, описанной в объяснительном тексте, если: а) $n = 8$; б) $n = 11$; в) $n = 13$. Для каждого из указанных n определите, какой из игроков имеет выигрышную стратегию.

⑩ Из клетчатой бумаги вырезан прямоугольник размером $m \times n$ клеток, где $m > 1$ и $n > 1$. Играют двое. Игроки делают ходы по очереди. Первым ходом прямоугольник разрезается на две части по прямой, идущей по сторонам клеток. За один ход разрешается разрезать любой из имеющихся прямоугольников на две части по прямой, идущей по сторонам клеток. Выигрывает тот, кому удастся своим ходом отрезать квадратик со стороной 1. Постройте дерево игры, если: а) $m = 5$, $n = 7$; б) $m = 9$, $n = 10$; в) $m = n = 11$. Для каждой пары указанных значений m и n определите, какой из игроков имеет выигрышную стратегию.

⑪ Перед двумя игроками лежат две кучки камней: в одной m камней, в другой n камней. За один ход разрешается увеличить число камней в любой из двух кучек в 3 раза или добавить в любую из кучек 1 камень. Игроки делают ходы по очереди. Выигрывает тот, кому удастся своим ходом получить суммарное количество камней в обеих кучках, большее 20. Постройте дерево игры, если: а) $m = n = 4$; б) $m = 2$, $n = 3$. Для каждой пары указанных значений m и n определите, какой из игроков имеет выигрышную стратегию.

⑫ Как можно доказать, что некоторая игра является конечной?

⑬ Докажите, что при любом начальном значении n игра, описанная в объяснительном тексте, является конечной.

⑭ Докажите, что игры, описанные в заданиях 10 и 11, при любых начальных значениях m и n являются конечными.

§ 58 Построение стратегии

Для большинства игр осуществить полный перебор всевозможных последовательностей ходов, как правило, практически невозможно. Даже если в каждой позиции нужно делать выбор всего лишь из двух вариантов, то игра двух игроков, продолжающаяся n ходов, потребует рассмотрения 2^n последовательностей. Уже при $n = 30$ (т. е. каждый игрок сделает всего по 15 ходов) требуется перебрать более миллиарда вариантов.

Давайте еще раз посмотрим на рисунки 7.2 и 7.3. Обратите внимание, что в различных последовательностях ходов возникают одинаковые позиции. Так, после второго хода (рис. 7.2) возможны всего три различные позиции: (5; 2; 2), (4; 3; 2) и (3; 3; 3). Позиции, записи которых отличаются только порядком чисел, одинаковы. А после третьего хода, если он возможен, вообще оказалась только одна позиция: (3; 2; 2; 2). Но если позиции одинаковы, то и решение игрока, каким должен быть очередной ход, будет для них одинаковым. Тем самым напрашивается вывод: анализировать надо не последовательности ходов, а позиции, которые могут реализоваться в ходе игры.

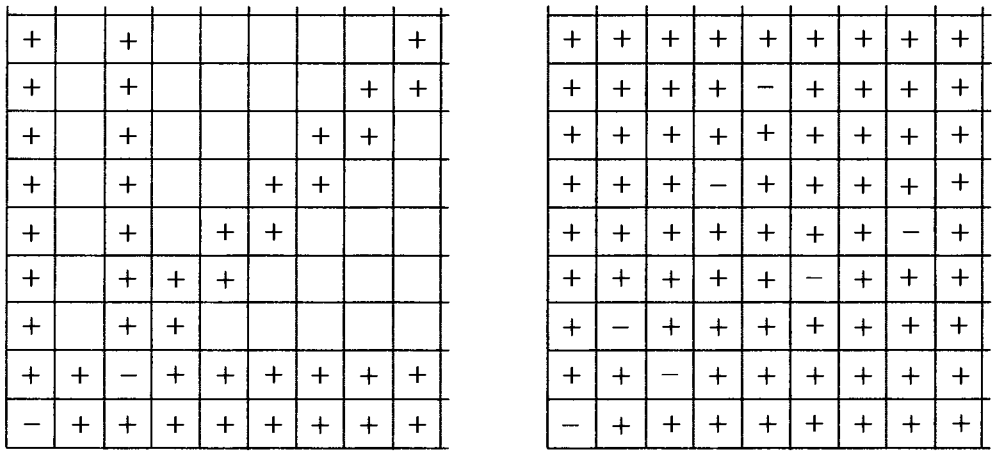
В конечной игре с полной информацией для каждой позиции существует такая последовательность ходов, которая приводит к гарантированному оптимальному результату.

Чтобы проиллюстрировать обсуждаемые идеи, рассмотрим следующую простую игру. На прямоугольном клетчатом поле в некоторой клетке ставится фишка. За один ход ее разрешается переместить на любое количество клеток либо по горизонтали влево, либо по вертикали вниз, либо по диагонали влево и вниз. Два игрока по очереди делают ходы фишкой. Проигрывает тот, кто не сможет сделать очередной ход.

Нетрудно видеть, что в этой игре ровно одна заключительная позиция — фишка стоит в левом нижнем углу. Для каждого игрока заключительная позиция имеет ровно одно значение — тот, кто привел фишку на это поле, выиграл; другой игрок проиграл. Каждая незаключительная позиция для игрока, который из этой позиции должен делать ход, будет выигрышной или проигрышной в зависимости от того, имеется ли для нее последовательность ходов, гарантирующая выигрыш. Давайте разметим клетки игрового поля, поставив знак «+», если позиция, когда фишка стоит на этой клетке, является выигрышной, и знак «−», если позиция проигрышная. Столбцы будем нумеровать слева направо, строки — снизу вверх. Клетка в первом ряду и в первом столбце — это левая нижняя клетка: там, очевидно, стоит знак «−». Но тогда во всех клетках вертикального, горизонтального и диагонального рядов, начинающихся с этой клетки, надо поставить знак «+». Клетка, стоящая во втором ряду и третьем столбце, должна быть отмечена знаком «−», поскольку из нее фишку можно передвинуть только в клетку, уже отмеченную знаком «+», т. е. в клетку гарантированного выигрыша противника. Но тогда сразу же знаком «+» надо отметить все еще не помеченные клетки второго ряда, третьего столбца и диагонали, проходящей через эту клетку (рис. 7.4, *а*). На рисунке 7.4, *б* приведена полная разметка некоторой части доски как угодно большого размера.

Теперь вместо того, чтобы анализировать всевозможные последовательности ходов, достаточно знать проигрышные позиции: стратегия игры состоит в том, чтобы своим ходом поставить фишку на клетку, отмеченную знаком «−». Эту стратегию можно оформить следующим алгоритмом:

```
Алгоритм Стратегия+
{ Делать пока (фишка не стоит в левом нижнем углу)
   { Если (фишка стоит на клетке, отмеченной знаком «+») то
      { перевести ее, следуя правилам игры, на клетку,
        отмеченную знаком «−»;
      }
```



а) б)

Рис. 7.4 Разметка игрового поля выигрышными (+) и проигрышными (–) позициями

```
            иначе
            { перевести ее, следуя правилам игры, на любую другую клетку;
            }
                        (* Легко видеть, что по способу заполнения
                           поля невозможно перевести фишку
                           из «проигрышной» клетки в «проигрышную» *)
        }
    }
```

Отметим, что это еще не алгоритм игры. Ведь алгоритм игры должен состоять из допустимых действий исполнителя (в данном случае игрока).

Предположим, что размеры поля изначально фиксированы. Тогда можно составить список проигрышных позиций. Такой список может состоять, например, из пар номеров строк и столбцов тех клеток, в которых стоит знак «–»: (1; 1), (2; 3), (3; 2), (4; 6), (5; 8), (6; 4), (7; 11) и т. д. Для краткости будем называть эти пары чисел координатами. В памяти компьютера список проигрышных позиций можно хранить в виде двумерного массива. Приведем алгоритм для одного игрока; при этом мы не проверяем корректность ходов второго игрока, считая, что он поступает честно и все ходы делает строго по правилам.

```
Алгоритм Игра
цел: maxx, maxy;    (* размеры поля по горизонтали и вертикали *)
цел: n, a[1:n; 1:2];                (* таблица проигрышных позиций *)
```

```
цел: k, m, x, y;
{ Запросить n;
  Запросить a[1:n; 1:2];      (* реально ввод таблицы оформляется
                                 двойным циклом *)
  Запросить maxx;
  Запросить maxy;
  x := rand(maxx);
  y := rand(maxy));           (* rand(N) — генератор случайных
                                 чисел из промежутка [0; N] *)
  Делать пока (x > 1 и y > 1)
  { Сообщить "Фишка стоит на поле", x, y;
    Сообщить "Введите координаты фишки после вашего хода";
    Запросить x, y;
    Если (x = 1 и y = 1) то { Сообщить "Вы выиграли. Поздравляю!"; }
    иначе
    { ХОД (x, y, k, m);
      Сообщить "Мой ход – фишка на поле", k, m;
      Если (k = 1 и m = 1) то
      { Сообщить "Я выиграл. Поздравьте меня!";
      }
    }
  }
}
```

В этом алгоритме использован вспомогательный алгоритм ХОД. Вот как он выглядит:

```
Алгоритм ХОД (арг: x, y; рез: k, m)
цел: i;
{ Если (x = 1 или y = 1) то
  { k := 1;
    m := 1;
  }
  иначе
  { i :=1;
    Делать пока (a[i, 1] < x или a[i, 1] > x или i = n + 1)
    { i := i :+ 1;
    }              (* поиск знака «–» на вертикали с номером x *)
    Если (i < n + 1 и a[i, 2] < y) то
    { m := a[i, 2];
      k := x;
    }      (* знак «–» оказался ниже клетки, где стоит фишка *)
    иначе
    {  Если (i < n + 1 и a[i, 2] = y) то
       {   m := y – 1;
           k := x;
       }       (* знак «–» оказался в клетке, где стоит фишка *)
```

```
            иначе
            {  i := 1;
               Делать пока (a[i, 2] < y или a[i, 2] > y или i = n + 1)
               { i := i :+ 1;
               }        (* поиск знака «–» на горизонтали с номером y *)
               Если (i < n + 1 и a[i, 2] < y) то
               {  k := a[i, 1];
                  m := y;
               }                          (* знак «–» оказался левее клетки,
                                             где стоит фишка *)
               иначе
               {  i := 1;
               Делать пока (a[i, 2] – a[i, 1]<y – x или a[i, 2] – a[i, 1]>y – x)
                {  i := i :+ 1;}          (*поиск знака «–» на диагонали *)
                  k := a[i, 1];
                  m := a[i, 2];
               }                                    (* ход по диагонали *)
            }
         }
      }
   }
```

Как видите, от стратегии игры до алгоритма игры дистанция огромного размера.

Можно играть на клетчатом поле, у которого нет верхней и правой границ. Игровое пространство (т. е. совокупность клеток, на которые можно ставить фишки в ходе игры) определяется начальным положением фишки, которое выбирается случайным образом. При этом, однако, мы вынуждены поручить компьютеру самому отыскивать поля, которые должны быть отмечены знаком «–». Алгоритм, реализующий описанный выше подход к выявлению проигрышных полей, работает довольно долго и требует, хотя бы однократно, использовать двумерный массив размерности $m \times n$, где (m, n) — начальное положение фишки. При больших значениях m и n могут возникнуть проблемы, связанные с ограничениями по памяти, да и время заполнения такого массива может стать значительным. Хотелось бы иметь алгоритм заполнения массива проигрышных позиций без предварительного изучения всех позиций игрового пространства. И такой алгоритм есть. Но о нем мы поговорим чуть позже.

Найти стратегию и показать, что она приводит к успеху, далеко не всегда так просто, как в только что рассмотренной игре. Рассмотрим другую игру.

Имеется два набора камней. Игроку за один ход разрешается взять либо любое количество камней из одного набора, либо из двух наборов сразу одинаковое количество камней. Проигрывает тот, кому на очередном ходе будет нечего брать.

Каждую позицию в этой игре естественно описать парой чисел, указывающих, сколько камней осталось в каждом наборе после очередного хода. Попытаемся составить список проигрышных позиций. Первая из них очевидна: (0, 0). Следующая, как нетрудно видеть, (1, 2), а также симметричная ей (2, 1). Затем идет пара позиций (3, 5) и (5, 3). Потом (4, 7) и (7, 4). Далее... Тут приходится задуматься надолго.

Но если задуматься, то можно заметить, что эта игра, по существу, не отличается от предыдущей: каждой позиции (x, y) данной игры мы можем сопоставить положение $(x + 1, y + 1)$ фишки в игре на клетчатом поле. Допустимые действия игрока в каждой из игр в точности соответствуют друг другу. Так что эти две игры можно назвать **эквивалентными**.

Каждая проигрышная позиция в одной игре в точности соответствует проигрышной позиции в другой. Теперь уже несложно продолжить список проигрышных позиций в игре с двумя кучками камней: (6, 10) и (10, 6), (8, 13) и (13, 8) и т. д.

Такая замена одного процесса (и вовсе не обязательно игрового) другим довольно часто оказывается весьма полезной для исследовательских целей.

Обдумывая рассмотренный пример игры, можно прийти к выводу, что стратегией естественно называть некоторый **алгоритм планирования**. Это определение годится и для игр с неполной информацией — мы все равно можем (и должны) каким-то образом планировать, какой ход сделать в той или иной позиции. Правда, для таких игр бывает удобным рассматривать не какую-то одну стратегию, а некоторое множество стратегий S, каждая из которых однозначно определяет, какой ход следует выбрать в каждой позиции, возникающей в игре. Для каждой стратегии существует некоторый гарантированный результат игры — это минимальный результат среди всех результатов, которые получаются, если рассмотреть все варианты игры противника при данной стратегии. Дж. фон Нейман (уже известный вам как автор принципов архитектуры ЭВМ) предложил выбирать такую стратегию, которая обеспечивает наибольший гарантированный результат. Такое определение наилучшей стратегии предполагает проведение полного перебора вариантов игры. Однако количество таких вариантов обычно так велико, что с их перебором не может справиться ни человек, ни компьютер. Человек интуитивно отбрасывает неперспективные, по его мнению, варианты. При этом слово «интуитивно» вовсе не является синонимом слова «неосознанно»: человек может руководствоваться вполне четко сформулированными критериями отбора вариантов, но сами эти критерии не являются логически обоснованными. Такие критерии называют эвристиками. Иными словами, можно сказать, что **эвристика** — это некое правило, сокращающее число потенциальных вариантов перебора. Формализованные эвристики нередко служат основой для создания эффективных

алгоритмов, однако надо помнить, что применение эвристик не гарантирует получение правильного результата. Огромна роль эвристик в творчестве. Как отмечал один из величайших математиков А. Пуанкаре, основное в творчестве — это умение обнаруживать полезные комбинации без перебора всех возможных.

Вопросы и задания

1. Почему в игре с полной информацией для построения стратегии можно вместо дерева игры рассматривать множество возникающих позиций?
2. Что такое эвристика?
3. В главе 1 учебника для 10 класса говорилось, что способ обработки информации может быть алгоритмическим, а может быть эвристическим. В чем, на ваш взгляд, сходство и в чем различие между эвристическим способом обработки информации и эвристикой?
4. Рассмотрите игру, описанную в задании 11 к § 57. Составьте список проигрышных позиций для этой игры для всех таких пар (m, n), у которых $m + n < 20$. (Совет. Позиции удобно изображать точками на координатной плоскости.)
5. Игра «Ним». Два игрока играют в следующую игру. Перед ними лежат три кучки камней, в одной k камней, в другой m камней, в третьей n камней. За один ход разрешается из любой кучки взять любое количество камней. Проигрывает тот, кто не может сделать очередной ход (т. е. камни кончились). Составьте список проигрышных позиций, если: а) $k = 2$, $m = 4$, $n = 6$; б) $k = 3$, $m = 5$, $n = 7$.
6. Всем, наверно, известна игра в крестики-нолики на девятиклеточном квадратном поле. Напомним правила. Два игрока по очереди ставят в клетки крестик или нолик — один игрок ставит только крестик, другой — только нолик. Выигрывает тот, кому удастся выстроить ряд из своих трех символов по горизонтали, вертикали или диагонали.

   а) Сколько ребер содержит дерево этой игры?

   б) Договоримся не различать позиции, которые возникают одна из другой поворотом вокруг центра на 90°, 180°, 270°, а также осевой симметрией относительно горизонтальной, вертикальной и диагональных осей симметрии квадрата. Составьте списки различных позиций, возникающих после каждого хода.

   в) Докажите, что у игрока, делающего ход первым, есть стратегия, позволяющая ему не проиграть второму игроку.
7. * а) Два игрока играют в крестики-нолики на доске размером 4×4. Условие выигрыша то же самое — три своих символа в ряду, параллельном стороне квадрата или его диагонали. Исследуйте эту игру по образцу задания 6.

б) Два игрока играют в крестики-нолики на доске размером 4×4. Условие выигрыша — четыре своих символа по горизонтали, вертикали или диагонали. Исследуйте эту игру по образцу задания 6.

8 Составьте алгоритм, позволяющий сформировать массив $a[1{:}\,n;\ 1{:}2]$ проигрышных позиций в игре, описанной в объяснительном тексте этого параграфа, если поле, на котором идет игра, имеет размер $k \times m$ клеток. Для этого вам необходимо оценить число n — возможное количество проигрышных позиций. Проигрышных позиций не может быть больше, чем общее число клеток на поле, поэтому годится число $n = km$. Но проигрышных позиций намного меньше, так что организовывать такой большой массив весьма неэкономно. Подумайте, каким может быть наименьшее n. Обоснуйте свою точку зрения.

§ 59 Инвариант стратегии

В главе 5 вы познакомились с двумя важными инструментами исследования алгоритмов: лимитирующей функцией и инвариантом цикла. Оказывается, что понятие инварианта тоже весьма полезно при изучении выигрышных стратегий. Впрочем, это можно было предвидеть: ведь стратегия обычно выглядит как некий циклический алгоритм перехода от одной проигрышной ситуации к другой. Стратегия, собственно, в том и состоит, чтобы каждый раз создавать противнику проигрышную позицию. Для этого можно просто располагать списком таких позиций (как это и делалось в предыдущем параграфе), а можно попытаться найти такое свойство, которым обладают все проигрышные позиции, но не обладает ни одна выигрышная. Вот такое свойство, неизменное для всех проигрышных позиций, и называют **инвариантом стратегии.** Знание инварианта стратегии намного эффективнее: ведь количество проигрышных позиций может быть как угодно велико. Но надо уметь находить этот инвариант всех проигрышных позиций. Покажем, как это делается.

Начнем с примера — с игры «Ним». Перед игроками лежат три кучки камней: в одной k камней, в другой m камней, в третьей n камней. За один ход разрешается из любой кучки взять любое количество камней. Проигрывает тот, кто не может сделать очередной ход. Проигрышные позиции для случая $k = 3$, $m = 5$ и $n = 7$ вы выписали, выполнив задание 5б к § 58. Вот какой список у вас должен был получиться:

(0; 0; 0), (0; 1; 1), (0; 2; 2), (0; 3; 3), (0; 4; 4), (0; 5; 5),
(1; 2; 3), (1; 4; 5), (2; 4; 6), (2; 5; 7), (3; 4; 7), (3; 5; 6).

Мы выписали позиции, расположив внутри каждой из них числа в порядке неубывания. Проигрышными будут и любые по-

зиции, полученные из перечисленных произвольной перестановкой чисел.

Запишем каждое число в двоичной системе. К примеру, для тройки (2; 4; 6), получится так: (010; 100; 110). А теперь выполним поразрядное сложение по модулю 2 чисел этой тройки. Поразрядное значит без переноса единицы в старший разряд. Напомним, что операцию сложения по модулю 2 мы обозначали символом $\oplus$. Имеем

$$010 \oplus 100 \oplus 110 = 000.$$

Проделайте это для любой позиции из приведенного списка, и вы убедитесь, что в результате всегда будут получаться только нули. Возникает гипотеза, что нами обнаружен инвариант стратегии — поразрядная сумма двоично записанных чисел данной позиции должна быть нулевой.

Как же обосновать нашу гипотезу? Как вообще обосновывать гипотезу, что некоторое свойство P определяет проигрышные позиции? Для этого надо убедиться в справедливости двух утверждений.

1. Из любой позиции, обладающей свойством P, можно перейти только в позицию, не обладающую этим свойством (т. е. противник не может перевести вас в проигрышную позицию).
2. Из любой позиции, не обладающей свойством P, можно перейти в позицию, обладающую этим свойством (т. е. своим очередным ходом вы можете поставить противника в проигрышную ситуацию).

Справедливость первого утверждения для нашей гипотезы почти очевидна. Действительно, уменьшив какое-либо число в позиции, обладающей сформулированным свойством, мы изменим цифру, как минимум, в одном разряде. И следовательно, сумма цифр в этом разряде уже перестанет быть нулем.

Пусть теперь перед нами позиция, не обладающая данным свойством. Рассмотрим самый левый из разрядов, в которых сумма отлична от нуля. Выберем то число из трех, для которого в этом разряде стоит 1, — такое число есть, ибо иначе сумма цифр в этом разряде была бы равна нулю. Перестроим выбранное нами число следующим образом. В каждом разряде этого числа, где сумма отлична от нуля, заменим цифру на «противоположную», т. е. 1 заменяем на 0, а 0 — на 1. Разряды, в которых сумма равна нулю, оставляем без изменения. Получившаяся запись — это и есть то количество камней, которое нам требуется иметь в данной кучке, чтобы получить позицию, удовлетворяющую инварианту. К примеру, для позиции (1; 3; 4) имеем

$$001 \oplus 011 \oplus 100 = 110.$$

В результате поразрядного сложения самая левая цифра 1 стоит в первом разряде. Из наших трех чисел в первом разряде цифру 1 имеет число 100. Производим в нем замену цифр: первая цифра заменяется на 0, вторая — на 1, а третья остается без изменений. В итоге получается 010, т. е. число 2 в десятичной системе счисления. Значит, в третьей кучке надо оставить два камня из четырех. Получится позиция (1; 3; 2), проигрышность которой нам уже известна ($001 \oplus 011 \oplus 010 = 000$).

А вот другой пример — позиция (2; 5; 6). Находим поразрядную сумму:

$$010 \oplus 101 \oplus 110 = 001.$$

В записи 001 самая левая цифра 1 стоит в третьем разряде. Из наших трех чисел в третьем разряде цифру 1 имеет число 101. Производим в нем замену цифр: первая и вторая цифры остаются без изменений, а третья цифра заменяется на 0. В итоге получается 100, т. е. число 4 в десятичной системе счисления. Значит, во второй кучке надо оставить четыре камня из пяти. Получится позиция (2; 4; 6), проигрышность которой нам уже известна ($010 \oplus 100 \oplus 110 = 000$).

После того как найден инвариант, стратегию построить легко: надо всегда для противника создавать позицию, обладающую найденным инвариантом.

Найти стратегический инвариант непросто. Нередко в этом помогают рассуждения, связанные со свойством симметрии. Рассмотрим для примера такую игру.

В каждой клетке доски размером 11×11 стоит шашка. За ход разрешается снять с доски любое количество, но не менее двух подряд идущих шашек либо из одного вертикального, либо из одного горизонтального ряда. Играют двое, ходы делают по очереди. Выигрывает снявший последнюю шашку. Кто выигрывает при правильной игре?

Выигрывает игрок, делающий первый ход. Этим ходом он снимает все шашки, стоящие на вертикали, проходящей через центр доски. После этого конфигурация шашек распадается на две симметричные части: левую и правую. На любой ход второго игрока первый отвечает ходом в другой половине доски так, чтобы сохранялась симметрия расположения шашек. Пока у второго игрока будет возможность сделать ход, у первого игрока также будет возможность хода. А стратегическим инвариантом здесь является осевая симметрия конфигурации игрового материала.

Впрочем, симметрия может применяться не в буквальном, геометрическом смысле, а как некое наводящее соображение. Вот пример еще одной игры.

Имеется два одинаковых набора конфет. Каждый из двух игроков по очереди, делая ход, съедает из какого-либо набора любое

количество конфет (после первого хода наборы конфет могут быть разными). Проигрывает тот, кому не удается сделать очередной ход, поскольку конфеты кончились.

Каждая позиция в этой игре после очередного хода какого-либо игрока описывается парой натуральных чисел (m, n), где m — число конфет в первом наборе, а n — число конфет во втором наборе. Начальная позиция описывается парой вида (a, a).

Проанализируем несколько позиций. Ясно, что позиции $(0, n)$ и $(m, 0)$ выигрышны для игрока, который будет делать ход — он просто забирает все оставшиеся конфеты. А вот позиция (1, 1) для такого игрока проигрышна — он ведь может съесть конфету только из одного набора, после чего его противник выигрывает. И позиция (2, 2) тоже проигрышная — съешь две конфеты, сразу проиграешь, а если съешь одну, то противник съест тоже одну из другого набора, после чего попадаешь в проигрышную позицию (1, 1). Легко теперь догадаться, что любая позиция вида (x, x) проигрышная для того, кто в этой позиции будет делать ход: сколько бы конфет ни съесть из одного набора, противник съест столько же из другого. В этом как раз и проявляется некая симметрия ответного хода второго игрока по отношению к ходу первого игрока. Вывод же таков: если игрок, делающий в начальной ситуации ход вторым, придерживается указанного правила, то он выигрывает, как бы ни играл первый игрок. Иными словами, у второго игрока имеется выигрышная стратегия.

Стратегии, в которых инвариантом является симметрия в том или ином смысле, называются **симметричными стратегиями.**

Вопросы и задания

1. Какое свойство позиций называют инвариантом стратегии?
2. Как инвариант стратегии используется для ее построения?
3. Пусть в игре с конфетами наборы первоначально содержат разное количество конфет. Определите, имеется ли в этом случае выигрышная стратегия и если имеется, то для какого игрока.
4. На основе стратегического инварианта игры «Ним» составьте алгоритм игры. (С о в е т. Для записи действия поразрядного сложения целых чисел используйте операцию XOR, имеющуюся в большинстве языков программирования. При выполнении этой операции каждое слагаемое автоматически переводится в двоичную систему счисления, поразрядно складывается и результат переводится обратно в десятичную систему счисления.)
5. Обобщение игры «Ним». Пусть имеется s кучек камней. За один ход разрешается из любой кучки взять любое количество камней. Играют

двое; ходы делают по очереди. Проигрывает тот, кто не может сделать очередной ход.

а) Найдите инвариант стратегии и сформулируйте выигрышную стратегию.

б) Составьте алгоритм для обобщенной игры «Ним».

6 Игра «Баше». Имеется кучка из n камней. За один ход разрешается взять из нее не более k камней. Играют двое, ходы делают по очереди. Проигрывает тот, кто не может сделать ход.

а) Выпишите проигрышные позиции для $k = 4$ и $n = 20$; для $k = 7$ и $n = 30$. У кого из играющих есть выигрышная стратегия в каждом из указанных случаев?

б) Найдите инвариант стратегии и сформулируйте выигрышную стратегию для произвольной пары (k, n).

в) Составьте алгоритм, реализующий игру по найденной вами стратегии.

7 Обобщенная игра «Баше». Имеется кучка из n камней. Играют двое, ходы делают по очереди. За один ход первому игроку разрешается взять из нее не более k камней, второму игроку — не более m камней. Проигрывает тот, кто не может сделать ход.

а) Определите, какой из игроков выигрывает в этой игре, и сформулируйте выигрышную стратегию для произвольной тройки (k, m, n).

б) Составьте алгоритм, реализующий игру по найденной вами стратегии.

8 На окружности расставлено 20 точек. За один ход разрешается соединить любые две из них отрезком, не пересекающим отрезков, проведенных ранее. Играют двое, ходы делают по очереди. Проигрывает тот, кто не может сделать ход. У какого игрока в этой игре имеется выигрышная стратегия?

9 а) В ряд лежат несколько монет, каждые две соседние монеты касаются друг друга. За ход разрешается брать одну монету или две соприкасающиеся монеты. Играют двое, ходы делают по очереди. Проигрывает тот, кому нечего брать. У какого игрока в этой игре имеется выигрышная стратегия?

б) По окружности лежат несколько монет, каждые две соседние монеты касаются друг друга. За ход разрешается брать одну монету или две соприкасающиеся монеты. Проигрывает тот, кому нечего брать. Выигрывает тот, кто берет последнюю монету. У какого игрока в этой игре имеется выигрышная стратегия?

10 Перечитайте еще раз условие игры, описанной в задании 10 к § 57. Пусть одна из сторон прямоугольника имеет четную длину. Определите, у какого игрока в таком случае имеется выигрышная стратегия.

11 Перечитайте еще раз условие игры, описанной в объяснительном тексте § 57.

а) Пусть первоначально в кучке имеется четное число камней. Определите, у какого игрока в таком случае имеется выигрышная стратегия.

б)* Для произвольного начального числа камней в кучке определите в зависимости от этого числа, у какого игрока имеется выигрышная стратегия.

в)* Составьте алгоритм, реализующий игру по найденной вами стратегии.

§ 60 Игра как модель управления

В начале этой главы было сказано, что игра — это модель управления. Чтобы убедиться в этом, достаточно еще раз проанализировать игры, которые рассматривались нами в предыдущих параграфах. Они совсем простые, но в них отражены принципы любой игры — воздействие на некоторую ситуацию двух сторон с целью приведения ее в наиболее благоприятное для себя состояние. А целенаправленное воздействие называется управлением. Действия противника можно рассматривать как внешние факторы. Каждая позиция описывается подходящим набором параметров, изменение которых обусловлено как действием противника, так и управляющим воздействием игрока. Получается схема, изображенная на рисунке 7.5.

Реализация обратной связи — это оценка позиции, сложившейся после хода противника. На основе этой оценки вырабатывается управляющее воздействие. К примеру, в игре с конфетами, описанной в конце § 59, оценивается единственный параметр — равно число m числу n или нет. А управляющее воздействие — съесть из большего набора $|m - n|$ конфет.

В этой игре оценочная функция описывается совсем просто. Но обычно поиск оценочной функции бывает более сложен. Рассмотрим еще одну игру, которая, наверно, многим хорошо известна, — крестики-нолики. Правила этой игры такие. Два игрока по очереди ставят на квадратном поле из n^2 клеток крестик и нолик —

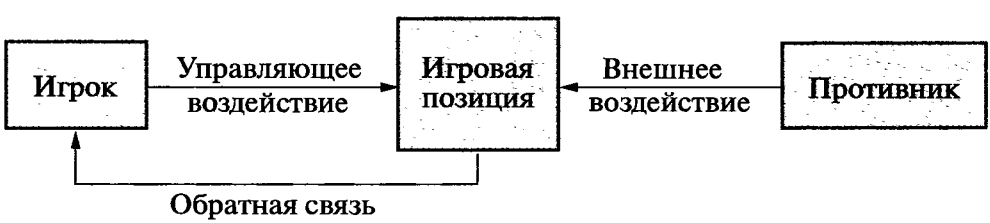


Рис. 7.5 Игра как модель управления

каждый из них свой символ. Выигрывает тот, кто первым составит сплошной горизонтальный, вертикальный или диагональный ряд заданной длины m из своих символов.

В лабораторной работе № 29 мы научим компьютер играть в эту игру на поле из 9 клеток с длиной выигрышного ряда 3 символа. Для определенности договоримся, что компьютер ставит нолик, а его противник — человек — ставит крестик. Право выбора, кому ходить первым, оставлено за человеком.

Общие принципы того, как это сделать, мы изложили выше. Для каждой позиции вычисляется ее оценка и на основе оценки дается указание, куда поставить очередной нолик. Сейчас мы расскажем, как для игры в крестики-нолики строится оценочная функция и как вырабатывается управляющее воздействие.

Прежде всего уточним, что такое оценочная функция. Вы, изучая школьный курс математики, рассматривали функцию как правило, по которому одному числу из области определения функции однозначно сопоставляется другое число. Правда, в § 35 нашего учебника для 10 класса мы ввели более общее понятие функционального отношения. Напомним, что отношение называется **функциональным**, если его атрибуты можно разбить на две группы так, чтобы значения одной группы атрибутов однозначно определяли значения другой группы. Иными словами, первая группа атрибутов может рассматриваться как аргумент некоторой функции, а вторая группа определяет значение этой функции. Оценочная функция — это некоторое функциональное отношение, которое сопоставляет набору характеристик, описывающих игровую позицию, набор числовых параметров.

Функциональное отношение удобно представлять себе в виде некоторого преобразователя со входами и выходами (рис. 7.6). На входы подаются значения атрибутов, относящихся к аргументу, а на выходе получаются соответствующие значения атрибутов, являющихся результатом.

Будем и нашу оценочную функцию рассматривать как устройство со входами и выходами. Его входами являются 9 клеток поля. На каждый вход может быть подан один из трех сигналов: «крестик», «нолик» или «пусто». Первоначально на все входы подан сигнал «пусто».

Что является выходами этого устройства? Выходов столько же, сколько и входов, и каждый как бы отвечает за соответствующую клетку. Представим, что на каждом выходе появляется новое число, как только сделан ход человеком, т. е. изменилось состояние одного из входов. Вот этот набор и реализует оценочный блок.

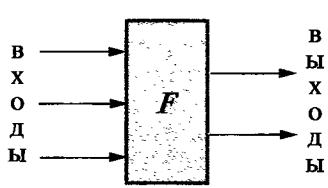


Рис. 7.6 Функциональное отношение как преобразователь

Теперь обсудим, какова роль блока выбора. Фактически он по выработанной

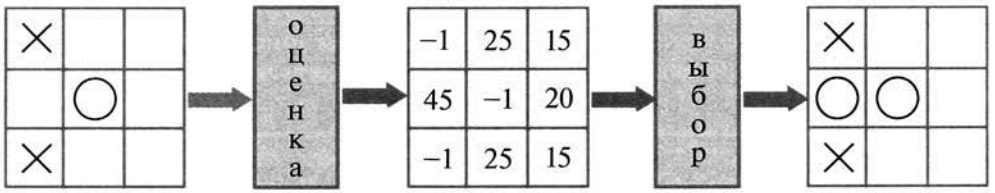


Рис. 7.7 Игра в крестики-нолики как управление неким устройством

оценке должен указать, в какую клетку поставить очередной нолик. Этим и реализуется управляющее воздействие. Описанный процесс можно изобразить так, как показано на рисунке 7.7.

Блок выбора реализован очень просто: нолик ставится в клетку с наибольшим значением оценочной функции. А вот блок оценивания заслуживает отдельного разговора.

Если клетка уже занята ноликом или крестиком, то в нее нолик ставить нельзя. Поэтому оценим ее числом −1. Все свободные клетки будут иметь неотрицательную оценку.

Для каждой свободной клетки будем рассматривать всевозможные содержащие ее трехклеточные ряды — горизонтальные, вертикальные и диагональные. Для каждого ряда могут встретиться варианты (напомним, что рассматривается ряд, содержащий пустую клетку):

1) в ряду все клетки пустые;
2) в ряду стоит ровно один нолик;
3) в ряду стоит ровно один крестик;
4) в ряду стоит один крестик и один нолик;
5) в ряду стоит два крестика;
6) в ряду стоит два нолика.

Если в ряду уже есть два нолика, то надо немедленно ставить третий и тем самым выигрывать. Так что шестой вариант имеет наивысший приоритет. Если шестой вариант не осуществился, но через клетку проходит ряд с двумя крестиками (пятый вариант), то нолик надо ставить в эту клетку, иначе следующий ход противника приведет к поражению. Если в ряду стоит один крестик и один нолик, то в этом ряду нельзя ни выиграть, ни проиграть. Этот вариант имеет самый низкий приоритет. Осталось упорядочить три первых варианта. Здесь трудно высказать какое-либо определенное мнение о приоритете. Можно, к примеру, считать его одинаковым для всех этих вариантов, а можно присвоить в том порядке, как они перечислены. В первом случае будет реализована оборонительная тактика, а во втором — наступательная.

Перечисленные варианты — это факторы, влияющие на принятие решения об управляющем воздействии. Теперь их действие надо описать числовыми параметрами. Сделаем это так, как показано в таблице 7.1.

Таблица 7.1

Вариант	Оценка
В ряду стоит один крестик и один нолик	0
В ряду все клетки пустые	5
В ряду стоит ровно один крестик	10
В ряду стоит ровно один нолик	15
В ряду стоит два крестика	30
В ряду стоит два нолика	60

Следующий шаг состоит в установлении связей между параметрами, описывающими действие факторов, и выходными параметрами. В нашем случае это значение оценочной функции для каждой клетки. Эту функцию тоже можно построить по-разному. Можно для выбранной клетки взять наибольшее значение по тем рядам, которые проходят через эту клетку. Например, для правой угловой клетки в нижнем ряду в позиции, изображенной на рисунке 7.7, в соответствии с таблицей 7.1 получаем число 10. Можно взять другую функцию — суммировать значения по всем рядам, проходящим через эту клетку. Тогда для той же клетки в той же позиции получается 15. В средней таблице на рисунке 7.7 приведены значения как раз для такой оценочной функции.

Удачно ли составлена таблица 7.1, правильно ли выбрана оценочная функция, иными словами, адекватна ли построенная нами модель, можно узнать, только проведя эксперимент, т. е. поиграв с компьютером в эту игру. Вы займетесь этим, выполняя лабораторную работу, и, может быть, придется внести какие-нибудь коррективы.

Для игр, рассмотренных ранее, можно доказать, что построенная стратегия наверняка приведет к успеху. В игре в крестики-нолики сформулированные правила вовсе не гарантируют получение наилучшего результата. Но их преимущество в том, что они позволяют избежать полного перебора вариантов развития игры. Такие правила, не обоснованные строгими доказательствами, но сокращающие перебор вариантов, как мы уже говорили, называют **эвристиками**.

В игре может участвовать не два игрока, а большее число. Но с точки зрения построения управления игрой это не имеет принципиального значения: можно считать, что все остальные игроки — это один «коллективный» игрок. Обратите внимание: в игре в крестики-нолики, как и положено в системе с обратной связью, мы не анализировали причины, по которым противник делал тот или иной ход, ставя нолик в какую-либо клетку, мы анализировали

лишь создавшуюся после его хода ситуацию. Поэтому безразлично, кем является второй игрок — человеком, который со своей стороны пытается построить управление игрой, или безрассудными силами природы, подчиненными воздействию случайных факторов.

При этом не возбраняется и даже нередко оказывается полезным при построении оценочной функции учитывать специфику действий противника. Но это означает, что при моделировании требуется учитывать значительно большее число факторов в качестве существенных.

И наконец, отметим, что оценочную функцию вполне можно применять и для игр с неполной информацией, реализуя подход, предложенный Дж. фон Нейманом, о котором мы говорили в § 58.

Вопросы и задания

1. Почему задачу выиграть некоторую игру можно рассматривать как задачу управления?

2. В правиле выбора хода, описанном при рассмотрении игры в крестики-нолики, не сказано, в какую клетку компьютер должен поставить нолик, если максимальное значение оценочной функции окажется не в одной, а в нескольких клетках. Предложите какой-либо способ, устраняющий эту неопределенность, недопустимую для формального исполнителя, каковым является компьютер.

3. Начертите два поля размером 3×3 для игры в крестики-нолики и заполните одно из них значениями первой из описанных в объяснительном тексте оценочных функций, другое значениями второй. Какая из оценочных функций, на ваш взгляд, больше подходит для выбора первого хода компьютером? Ответ обоснуйте.

4. Выполнив задание 6 к § 58, вы подсчитали, сколько ребер имеет дерево игры в обычные в крестики-нолики. Это число едва ли вселило в вас оптимизм строить стратегию, основываясь на таком дереве. Из-за результата выполнения задания 7 из того же параграфа вы могли только еще больше впасть в пессимизм. Поэтому мы предлагаем вам реализовать для этих игр оценочный подход.

   а) Постройте модель управления с помощью оценочной функции для игры в крестики-нолики на поле размером 4×4, если по-прежнему выигрывает тот, кто первым составил сплошной горизонтальный, вертикальный или диагональный ряд длины 3 из своих символов.

   б) Постройте модель управления для игры в крестики-нолики на поле размером 4×4, если выигрывает тот, кто первым составил сплошной горизонтальный, вертикальный или диагональный ряд длины 4 из своих символов.

5* Игра Реверси. В эту игру играют двое, выставляя по очереди на клетки квадратного поля фишки своего цвета. У одного игрока фишки зеленые, а у другого красные. Размеры поля — 8×8. В начале игры в центре поля стоят по две фишки каждого цвета так, как показано на рисунке 7.8. Если при очередном ходе только что поставленная фишка и какая-то другая фишка того же цвета ограничивают ряд фишек противника (этот ряд не должен содержать пустых клеток), то все фишки ряда заменяются на фишки игрока, сделавшего ход (ряд может быть горизонтальным, вертикальным или диагональным). Может так оказаться, что вновь поставленная фишка ограничивает сразу несколько рядов, тогда это правило применяется ко всем таким рядам. Игра заканчивается, когда на поле не осталось свободных клеток. Выигрывает тот игрок, чьих фишек оказалось больше.

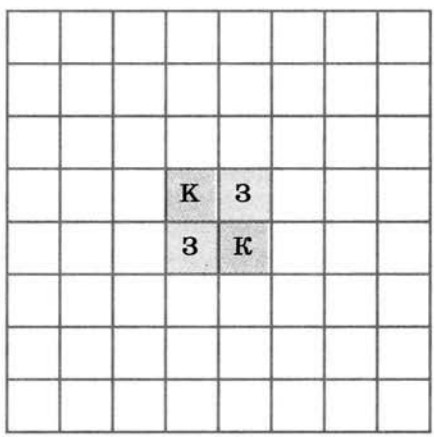
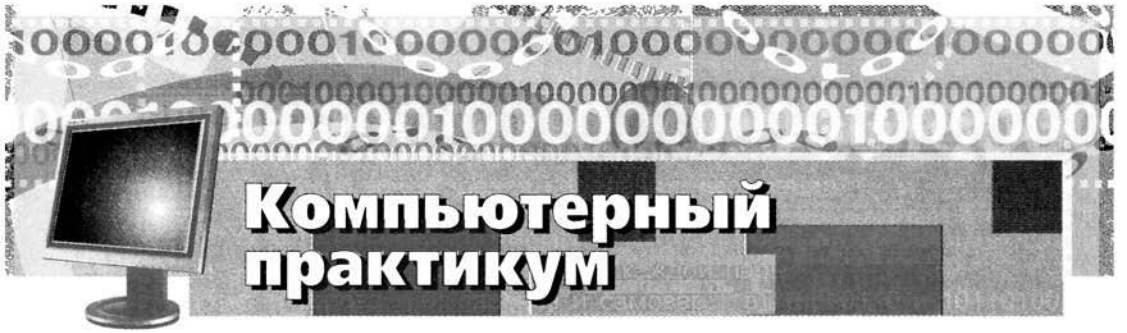


Рис. 7.8

а) Придумайте оценочную функцию, которая позволяла бы определить, на какое поле нужно поставить фишку при очередном ходе.

б) Составьте алгоритм, который бы реализовывал стратегию игры на основании вашей оценочной функции.

в) Реализуйте ваш алгоритм в виде программы.



Компьютерный практикум

Выполнять задания лабораторных работ вы будете в компьютерном классе. Поэтому нелишне еще раз напомнить правила техники безопасности.

1. Если вы обнаружили какую-либо неисправность, немедленно сообщите об этом преподавателю. Не работайте на неисправном оборудовании!
2. Не включайте и не выключайте компьютеры самостоятельно.
3. Не дергайте и вообще не трогайте различные провода.
4. Не стучите по клавиатуре и мышке.
5. Соблюдайте правила охраны здоровья: не работайте за компьютером на расстоянии менее 50 см от монитора, следите за осанкой, в перерывах делайте гимнастику для глаз и опорно-двигательного аппарата.

Каждый раз, приступая к выполнению лабораторной работы, помните и выполняйте эти несложные правила.

Таблица КП.1

	A	B	C
1	Точки	Высота	Скорость
2	0	1,5	0
3	A2+F3		=Корень(2*9,8*(B2–B3))
4	A3+F3		=Корень(2*9,8*(B2–B4))
...	...		...
...	...		...
9	A8+F3		=Корень(2*9,8*(B2–B9))
10	A9+F3	0	=Корень(2*9,8*B2)

Лабораторная работа № 1 (к § 6)

Модель горки. Проверка адекватности модели

В § 6 мы построили модель, позволяющую, как нам кажется, определить оптимальную конфигурацию горки для того, чтобы кататься на ней на санях. Приступим к компьютерной реализации этой модели с помощью электронной таблицы.

1. В таблице КП.1 представлен пример электронной таблицы. Исходными данными служат высота горки $h = 1{,}5$ м, длина основания горки 6 м и число точек разбиения n. Для начала возьмем n равным 7 и запишем его в ячейку F2. Таким образом, у нас будет 9 значений высот, которые мы обозначали буквой y с соответствующим индексом. Эти значения будем записывать в столбце B, начиная со второй строки. При этом в ячейку B2 запишется число 1,5 — высота всей горки, а в ячейку B10 — число 0. В ячейки B3 — B9 можно пока записать любые числа между 0 и 1,5. В ячейку F3 запишем формулу для вычисления длины каждой из частей разбиения. Столбец C отведем для вычисления значений скорости в конце каждого участка разбиения. В столбцах D и E запишем формулы для вычисления длины каждого участка и времени спуска по этому участку. (Глядя на формулы, обдумайте, как их записать в таблицу, чтобы было удобно копировать.) В ячейку F1 запишем суммарное время спуска с горки.

   Теперь наша задача — подобрать такие значения в ячейках B3—B9, чтобы значение в ячейке F1 было наименьшим.

D	E	F
Длина участка	Время	=СУММ(E2:E9)
=Корень((B2–B3)^2+F3^2)	=2*D2/(C2+C3)	7
=Корень((B3–B4)^2+F3^2)	=2*D3/(C3+C4)	6/(F2+1)
=Корень((B4–B5)^2+F3^2)	=2*D4/(C4+C5)	
...	...	
...	...	
=Корень((B9–B10)^2+F3^2)	=2*D9/(C9+C10)	

Таблица КП.2

Номер точки разбиения	0	1	2	3	4	5	6	7
Высота в точке разбиения	1,5	0,096145	−0,37637					0

Мы будем делать это, используя специальную программу в Excel — надстройку *Поиск решения*. Эта программа является дополнительной, поэтому может потребоваться ее установка. Выберите меню *Сервис*. Если в нем отсутствует пункт *Поиск решения*, то выберите в нем пункт *Надстройки* и в появившемся диалоговом окне отметьте галочкой пункт *Поиск решения*, после чего щелкните на кнопке OK. Программа установлена, и теперь ее имя появляется в меню *Сервис* автоматически.

Запустите эту программу. В появившемся диалоговом окне в целевую ячейку запишите адрес ячейки целевой функции (т. е. F1). Затем укажите, чего требуется добиться от целевой функции (в нашем случае минимального значения). Наконец, в окне *Изменяя ячейки* надо поместить указание на блок ячеек B3—B9 — именно их значения надо менять, чтобы добиться нужного результата. Дополнительных ограничений у нас нет, так что пришла пора щелкнуть на кнопке *Выполнить*. Через непродолжительное время компьютер отрапортует: *Решение найдено* — и спросит, сохранить ли полученное решение. Ответьте щелчком на кнопке OK. В ячейках B3—B9 вы увидите искомые значения промежуточных высот.

2 Начертите в тетради таблицу по образцу таблицы КП.2 и впишите в нее полученные результаты. С помощью мастера диаграмм постройте график изменения высоты горки (он будет выглядеть, вероятнее всего, так, как показано на рисунке КП.1).

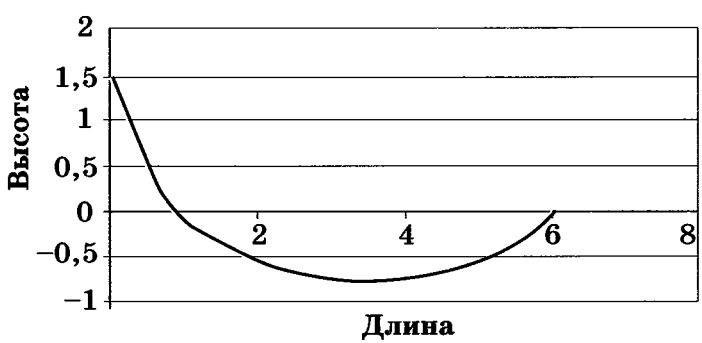


Рис. КП.1

Однако результаты могут вас разочаровать — появились отрицательные значения высот. Это означает, что придется почти на метр углублять горку. Ясно, что такое решение неприемлемо. Значит, построенная модель неадекватна и, следовательно, не были учтены какие-то существенные факторы.

3 Уточним модель, указав, что высота в любой точке разбиения должна быть заключена в пределах от 0 до 1,5. Эти условия нужно записать в окне *Ограничения*. Сделайте это и найдите оптимальное решение для такой модели. Имеющуюся у вас таблицу дополните еще одной строкой и впишите в нее результаты. Постройте график, который покажет вам, как в этом случае будет выглядеть горка.

4 В начале построения модели мы обсудили, что форма горки будет тем ближе к оптимальной, чем больше точек разбиения мы возьмем. Увеличьте количество точек разбиения до 9 и найдите оптимальное решение в этом случае сначала без учета ограничений (вдруг повезет!), а затем с учетом ограничений. Результаты впишите в новую таблицу (теперь в ней должно быть 11 столбцов) и постройте график, демонстрирующий профиль горки. Сравните его с графиком для семи точек разбиения.

5 Выполните ту же работу, взяв сначала 11 точек разбиения, а затем 14. Постройте графики и сравните их. Удобнее сравнивать графики, если они построены на одной и той же области диаграммы. Если вы считаете, что форма горки все еще существенно меняется при увеличении числа точек, продолжите компьютерный эксперимент, увеличивая количество точек разбиения.

Лабораторная работа № 2 (к § 8)

Задача о ценообразовании

В § 8 мы почти построили модель, с помощью которой надеемся определить оптимальную цену билетов в кинотеатр. Почти — потому, что осталось ненайденным значение параметра a в формуле для функции $f(x)$, показывающей количество проданных билетов по цене x рублей. Мы предлагаем для определения по экспериментальным данным значения параметра a использовать электронную таблицу. Заполнить ее можно, например, так, как показано в таблице КП.3. В ячейке D1 записано значение коэффициента c, а в ячейку D2, отведенную под коэффициент a, мы пока записали наугад взятое число.

1 В столбец E, начиная со второй ячейки, запишите отклонения расчетных данных от экспериментальных (при заданном значении цены). Так, в E2 нужно поместить форму-

Таблица КП.3

	A	B	C	D	E
1	Значение цены	Экспериментальные данные	$f(x)$	100	
2	0	100	=D1–D2*A2*A2	0,1	
3	50	98	=D1–D2*A3*A3		
4	80	94	...		
...			...		
...			...		
10	260	32	...		
11	290	16	=D1–D2*A11*A11		

лу =ABS(C2–B2). А в ячейку E1 запишите формулу МАКС(E2:E11). Наша задача – подобрать такое значение параметра a, при котором в E1 будет наименьшее возможное число. Сделать это можно, применяя, как и в лабораторной работе № 1, надстройку *Поиск решения*. Только на этот раз значение надо менять не в блоке ячеек, а всего лишь в одной ячейке — D2. Найдите искомое значение параметра a.

2 Теперь можно приступать к решению основной задачи — поиску такого значения цены, при которой выручка будет наибольшей. Для этого надо выделить ячейку для аргумента x и ячейку для подсчета выручки, т. е. значения функции $xf(x)$. Мы предлагаем для x отвести ячейку D3, для значения $f(x)$ — ячейку D4, для значения $xf(x)$ — ячейку D5, т. е. в D5 нужно записать формулу =D3*D4. Снова применяя надстройку *Поиск решения*, найдите значение x, при котором в ячейке D5 будет максимальное значение.

3 Когда мы провели этот вычислительный эксперимент, у нас получилось 182 р. 70 к. Но давайте обратим внимание на количество зрителей — 67 человек. Может быть, можно кино показывать в меньшем зале и получать максимальную выручку в нем? Ведь и расходы на аренду меньшего зала будут меньше. Измените значение в ячейке D1, записав туда, например, число 70. Проведите вычислительный эксперимент с этим значением (не меняя пока коэффициента a). Окажется, что максимальная выручка снова будет, когда зал заполнится на 2/3. Может быть, дело в том, что коэффициент a остался неизмен-

ным? Измените коэффициент a и повторите вычислительный эксперимент. Получился ли у вас другой результат?
Вооружившись математическими методами исследования функций, можно доказать, что максимальная выручка всегда будет получаться при заполнении кинозала на 2/3. Однако, как мы говорили в объяснительном тексте § 8, построенная модель годится не только для кинотеатров. Так что, господа будущие бизнесмены, запомните: максимальный доход получается тогда, когда предложение в полтора раза превышает спрос. Впрочем, капиталисты Запада давно это уже знают. Поэтому там вы никогда не увидите переполненный кинозал или забитый до отказа самолет, в любой гостинице найдутся свободные номера, а в кафе — свободный столик.

4 В вопросе 3 к § 8 мы предложили вам подумать о том, какой смысл имеет коэффициент a с экономической точки зрения. Проведите вычисления, меняя значения коэффициента a и наблюдая за изменением оптимальной цены билетов.

Вы обнаружили, что с уменьшением этого коэффициента оптимальная цена возрастает. В жизни, конечно, все наоборот: если человек готов платить за услугу или товар больше, то коэффициент a в нашей модели будет уменьшаться. Иными словами, этот коэффициент отражает изменение покупательной способности населения. Ну, возможно, не всего, а той его части, которая все еще ходит в кино.

Лабораторная работа № 3 (к § 11)

Системы счисления с основанием, равным степени числа 2

В памяти компьютера вся информация кодируется двумя символами. При этом для чисел обычно используют их запись в двоичной системе счисления. Но современные компьютеры (точнее, их программное обеспечение) очень дружески относятся к людям, ими пользующимися, и поэтому всю числовую информацию любезно предоставляют в десятичной системе счисления. Впрочем, по вашему желанию вы можете увидеть представление числа и в двоичной системе. Да и не только в двоичной.

1 Откройте приложение *Инженерный калькулятор*. На панели, расположенной слева под табло, вы видите трехбуквенные сочетания «Hex», «Dec», «Oct» и «Bin». Это сокращения латинских слов, которые переводятся на русский как «шестнадцатеричный», «десятичный», «восьмеричный» и «двоичный». Речь идет, как вы понимаете, о системах счисления.

По умолчанию включается представление чисел в десятичной системе счисления. Наберите число 19. Переключите калькулятор в режим «Bin». На табло появилось представление этого числа в двоичной системе счисления. Выполняя задание 2 к § 11, вы уже перевели это число в двоичную систему счисления. Сверьте свой ответ с ответом компьютера. Сошлось? Если нет, найдите у себя ошибку.

Переведите в двоичную систему счисления остальные числа из того же задания: 44, 129, 561, 1322. Сверьте результат компьютера со своим.

2 С помощью приложения *Инженерный калькулятор* переведите десятичные числа 19, 44, 129, 561, 1322 в шестнадцатеричную систему счисления. Сверьте компьютерные результаты с теми, которые получены вами.

3 А теперь с помощью того же приложения переведите числа 1001, 10101, 111001, 10111101 из двоичной системы счисления в десятичную (задание 3 из § 11). Сравните свой результат с результатом компьютера.

4 С помощью *Инженерного калькулятора* переведите числа 25, 4F, 1A7, ABC, D1AE, FFFF из шестнадцатеричной системы в десятичную (задание 4 из § 11). Сравните свой результат с результатом компьютера.

5 С помощью *Инженерного калькулятора*, а потом без него сложите и перемножьте числа 101_2 и 1101_2. Сверьте компьютерные результаты со своими.

6 Применять калькулятор для выполнения действий с небольшими числами все равно что стрелять из пушки по воробьям. Найдите сумму чисел 10 457 939 926 978 229 600 и 7 988 804 146 731 322 020, записанных в десятичной системе счисления, с помощью *Инженерного калькулятора*. Запишите полученный результат. (А можете ли вы прочитать каждое из слагаемых, правильно называя разряды и классы?) Теперь с помощью того же калькулятора переведите каждое из чисел в двоичную систему счисления. Запишите получившиеся числа и найдите их сумму. Вряд ли вы ожидали увидеть то, что появилось на табло калькулятора. В чем дело? Может быть, калькулятор сломался?

Введите сумму этих чисел, вычисленную в десятичной системе счисления, и переведите ее в двоичную систему. Изменился ли от этого результат?

▶ Чтобы понять причину наблюдаемого явления, давайте уменьшим второе слагаемое на 10, т. е. возьмем его равным 7 988 804 146 731 322 010. Теперь сложите это число с числом 10 457 939 926 978 229 600. Переведите результат в двоичную си-

стему. А теперь прибавьте к полученному числу 1010_2 — это как раз число 10, которое мы отняли у одного из слагаемых. Теперь стало видно, что сумма просто не уместилась на табло калькулятора — единица старшего разряда оказалась левее, чем это допускает количество мест, отведенных на табло. Такое явление называется переполнением разрядной сетки. Подробнее мы обсудим его в § 23, компьютерный эксперимент проведем в лабораторной работе № 5. ◀

7 Закройте приложение. Лабораторная работа завершена.

Лабораторная работа № 4 (к § 17)

Коды, обнаруживающие и исправляющие ошибки

Чтобы выполнить эту лабораторную работу, вам надо вспомнить операторы изучаемого вами языка программирования, позволяющие обрабатывать символьные переменные. В таблице КП.4 даны обозначения основных операций и функций.

1 Напишите алгоритм, который создает двумерный символьный массив KOD[1:10; 1:2], соответствующий таблице КП.5. Эта таблица реализует код Хэмминга для цифр 0, ..., 9, описанный в § 17 (см. табл. 2.11).
Запрограммируйте этот алгоритм и с помощью этой программы создайте указанный массив.

Таблица КП.4

Алгоритмы	Basic	Pascal	Пояснения
+	+	+	Операция соединения (конкатенации) слов
LEN(*s*)	LEN(*s*$)	length(*s*)	Функция, вычисляющая количество символов символьной переменной *s*
Часть (*s*, *a*, *b*)	MID$(*s*$, *a*, *b*)	COPY(*s*, *a*, *b*)	Функция, выделяющая в значении символьной переменной *s* часть, начинающуюся с символа с номером *a* и содержащую *b* символов

2 Составьте алгоритм, позволяющий запросить натуральное число и закодировать его кодом Хэмминга. Результат кодирования договоримся хранить в символьной переменной.
Запрограммируйте этот алгоритм и отладьте получившуюся программу.
Оформите эту программу как подпрограмму-функцию (процедуру) с именем KODN; ее аргументом является натуральное число, подлежащее кодированию.

Таблица КП.5

0	0000000
1	0001111
2	0010110
3	0011001
4	0100101
5	0101010
6	0110011
7	0111100
8	1000011
9	1001100

▶ Для проведения дальнейших экспериментов договоримся, что кодировать будем десятизначные числа. Сами эти числа будем получать с помощью датчика случайных чисел (сокращенно ДСЧ). Обычно он выполнен в виде подпрограммы-функции. В алгоритмах функцию получения случайного вещественного числа из промежутка [0; t) мы будем обозначать rand(t). Например, оператор

$$x := \text{rand}(t);$$

присваивает переменной x случайное значение из промежутка [0; t). Если требуется присвоить случайное значение из промежутка [a; b), то можно использовать оператор

$$x := a + \text{rand}(b - a);.$$

Иногда требуется получить целое случайное число x, удовлетворяющее неравенству $m \leqslant x \leqslant n$, где m и n — целые числа. Тогда можно использовать оператор

$$x := m + \text{INT}(\text{rand}(n + 0{,}5 - m));.$$

Запишем алгоритм получения десятизначного случайного числа в виде подпрограммы-функции:

```
Функция СЧ: цел
{  знач := 100000000 + INT(rand(899999999,5));
}
```

После кодирования десятизначного числа кодом Хэмминга получится символьная константа, содержащая 70 символов. Составим алгоритм, вносящий в этот код 6 ошибок, и запишем его тоже в виде подпрограммы функции:

```
Функция Помехи (сим: W) : сим
цел: k, m;
сим: b;
```

```
{ Делать от m := 1 до 6
  { k : = 1 + INT (rand(69,5));
    b : = Часть (W, k, 1)
    Если (b = "1") то { b := "0";}
    иначе { b := "1"; }
    знач := Часть (W, 1, k − 1) + b + Часть (W, k + 1, 70 − k);
  }
}
```

Составим теперь алгоритм кодирования с последующими искажениями:

```
Функция Искажение_кода: сим
цел: m;
сим: c;
{ m := СЧ;
  Сообщить m;                    (* печать передаваемого числа *)
  c := KODN(m);          (* кодирование числа m кодом Хэмминга)
  знач := Помехи (c);
} ◀
```

3. Запрограммируйте эти алгоритмы и отладьте их. Не забудьте, что для использования датчика случайных чисел его в самом начале программы надо инициализировать.

4. Составьте алгоритм, позволяющий для каждой семибитовой последовательности нулей и единиц находить ближайшее к ней слово из кода Хэмминга. Реализуйте этот алгоритм в виде подпрограммы-функции с именем Распознавание и аргументом W символьного типа. Результат также имеет символьный тип.

Составим теперь алгоритм, который имитирует передачу числовых данных с помехами с последующим исправлением ошибок:

```
Алгоритм Декодирование
цел: k, m, n;
сим: b, d;
{ b := Искажение_кода;
  n := 0;
  Делать от m := 0 до 9
  { n := n*10;
    d := Часть (b, 1 + 10*m, 10);
    k := 0;
    Делать пока (d ≠ KOD(k, 2))
    { k := k + 1; }
      n := n + k;
  }
      Сообщить n;
}
```

5 Запрограммируйте этот алгоритм.
Запустите несколько раз программу и проверьте, совпадает ли исходное число с тем, которое получается после декодирования.

6 Имитируемый нами канал связи весьма слабо защищен от помех — на 70 двоичных символов допускается до 6 ошибок. Поэтому можно ожидать, что иногда декодирование все-таки будет происходить с ошибками. Организуйте эксперимент для статистического определения того, как часто декодирование происходит с ошибкой. Создайте программу, которая позволила бы провести серию из 1000 испытаний; 10 000 испытаний. На основе этих данных оцените надежность такого кодирования.

Лабораторная работа № 5 (к § 23)

Представление целых чисел в памяти компьютера. Особенности компьютерной арифметики

Выполняя лабораторную работу № 3, вы уже обнаружили явление, которое было названо переполнением разрядной сетки. Это далеко не единственный эффект компьютерной арифметики. Давайте продолжим эксперименты с *Инженерным калькулятором*.

1 Наберите на табло *Инженерного калькулятора* отрицательное десятичное число −19. Перейдите в двоичную систему счисления. Как объяснить полученный вами результат? Сколько двоичных разрядов содержит табло *Инженерного калькулятора*? Какое самое большое двоичное число может быть записано на табло *Инженерного калькулятора*? Переведите это число в десятичную систему счисления.

2 Вычислите с помощью *Инженерного калькулятора* значения следующих разностей:
а) $111011101_2 - 1101110110_2$;
б) $11011010001_2 - 11101010001_2$.

Объясните вид получившихся результатов. Переведите каждый из результатов в десятичную систему счисления.

3 Найдите с помощью *Инженерного калькулятора* сумму
$11111111111111101111111101110_2 + 101111111101111_2$.

Переведите каждое из слагаемых в десятичную систему счисления. Найдите требуемую сумму. Попытайтесь перевести ее в двоичную систему счисления.

4 Переведите (вручную или с помощью *Инженерного калькулятора*) каждое из чисел из пункта 2 в шестнадцатеричную

систему счисления. Выполните над этими числами задание пункта 2 в шестнадцатеричной системе счисления. Объясните получившиеся результаты.

5 Запишите в шестнадцатеричной системе счисления наибольшее число, помещающееся на табло *Инженерного калькулятора*. Переведите это число в десятичную систему счисления. Совпало ли это число с тем числом, которое вы получили для двоичной системы счисления, выполнив пункт 1 этой лабораторной работы?

6 Закройте приложение. Лабораторная работа завершена.

Лабораторная работа № 6 (к § 24 и 25)

Представление вещественных чисел в памяти компьютера. Особенности компьютерной арифметики

Мы начнем с представления чисел с плавающей запятой, записанных в десятичной системе счисления, в приложениях *Калькулятор* и *Инженерный калькулятор*.

1 Запустите приложение *Калькулятор*, выбрав для него вид *Обычный*. Нажимая несколько раз клавишу с цифрой 1, выясните, сколько разрядов имеет этот калькулятор для чисел с фиксированной запятой. Умножьте набранное вами число на 10. Объясните высветившийся на табло результат операции.
В дальнейшем мы будем экспериментировать с получившимся числом, поэтому запишите его в память калькулятора.

2 К полученному числу прибавьте число 3. Объясните получившийся результат.

3 Возведите полученное число в квадрат (для этого достаточно нажать клавишу «*», а затем клавишу «=»). Придумайте, как можно выяснить наивысшее значение порядка, допустимое в данном калькуляторе. Найдите значение этого порядка.

4 Требуется вычислить значение

$$3{,}1 \cdot 10^{192} \times 7{,}6 \cdot 10^{114} \times 2{,}8 \cdot 10^{-52}.$$

К какому виду нужно привести сомножители (каким должен быть порядок чисел?), чтобы получить требуемый результат? Произведите эти вычисления.

5 Наберите число 0,101. Возведите полученное число в квадрат. Полученный результат еще раз возведите в квадрат и т. д. После нескольких повторений таких действий на табло появится 0. Объясните этот результат.

6 Найдите наименьшее (отрицательное) значение порядка, допустимое в данном калькуляторе.

7 Требуется вычислить значение

$$3{,}1 \cdot 10^{-200} \times 2{,}6 \cdot 10^{-120} \times 7{,}8 \cdot 10^{50}.$$

Каким будет результат, если вычисления производить на обычном калькуляторе с указанным в данном выражении порядком? Проверьте свою гипотезу. С каким порядком надо производить вычисления, чтобы получить правильный результат? Получите этот результат.

8 Выберите вид *Инженерный*. Калькулятор в этом виде имеет специальную клавишу для перехода от записи числа с фиксированной запятой к записи в нормализованном виде и обратно. На этой клавише написано F-E. Щелкнув по этой клавише правой кнопкой мыши, вы получите подсказку, в которой указано, для каких чисел *Инженерный калькулятор* автоматически переходит из режима с фиксированной запятой в режим с плавающей запятой.

Проверьте справедливость утверждения, сформулированного в этой подсказке. Какое наибольшее число разрядов допускает *Инженерный калькулятор* для чисел с фиксированной запятой?

9 В задании 8 из § 25 приведены два алгоритма решения одной и той же задачи. Запрограммируйте каждый из этих алгоритмов на изучаемом вами языке программирования. Отладьте программы на малых значениях N. Проверьте, действительно ли при $N = 1\,000\,000\,000$ эти программы дают разный результат. Найдите наибольшее N, для которого программы дают одинаковый результат.

10 Математики доказали, что сумма

$$1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5}+\frac{1}{6}+\frac{1}{7}+\ldots+\frac{1}{N}$$

становится как угодно большой при неограниченном увеличении N. Поставим задачу найти такое N, при котором указанная сумма станет больше 20. Можно попытаться сделать это с помощью следующего алгоритма:

Алгоритм Поиск
цел: N;
вещ: S;
{ $S := 1$;
$N := 1$;

```
        Делать пока (S < 20)
        { N :=N + 1;
          S := S + 1/N;
        }                          (*конец цикла*)
        Сообщить N;
    }
```

Запрограммируйте этот алгоритм и попытайтесь с помощью составленной вами программы найти N.
Скорее всего, вас постигла неудача — программа безостановочно работает, не выдавая ответа. Попытайтесь объяснить, почему произошло зацикливание программы.

11 Поменяйте константу 20 на меньшую, например на константу 10. Какое N сообщает программа для этой константы? Сообщим, что для константы 20 значение N приближенно равно квадрату значения N, полученного для константы 10. Подтверждает ли эта информация вашу гипотезу о причине зацикливания программы для 20?

12 Попытайтесь найти наибольшее значение константы, при котором алгоритм еще не зацикливается.

Лабораторная работа № 7 (к § 26)

Создание текстовых информационных объектов

Выполняя задание 11 к § 26, вы выбрали литературное произведение. Ваша задача — составить и оформить подробный план этого произведения. Прежде всего определим уровень детализации. Вы знаете, что в литературном произведении каждый абзац обладает свойством логической завершенности. Поэтому самый нижний уровень детализации, представленный в вашем будущем плане, — это заголовок для абзаца. В свою очередь, несколько соседних между собой абзацев могут быть объединены в группу, которая также обладает неким логическим единством. Для них естественно иметь свой пункт плана более высокого ранга, нежели для абзацев, и т. д.

1 Наметьте не менее трех уровней в вашем будущем плане и приступайте к реализации. Получившийся у вас план должен иметь структуру многоуровневого списка (рис. КП 2).

▶ Чтобы создать такой список, надо сначала воспользоваться кнопкой *Нумерованный список*. Чтобы создать в этом списке элементы разного уровня, надо установить курсор в начало строки и нажать клавишу Tab. Это переведет элемент списка на следующий уровень. И каждое нажатие клавиши Tab будет переводить элемент

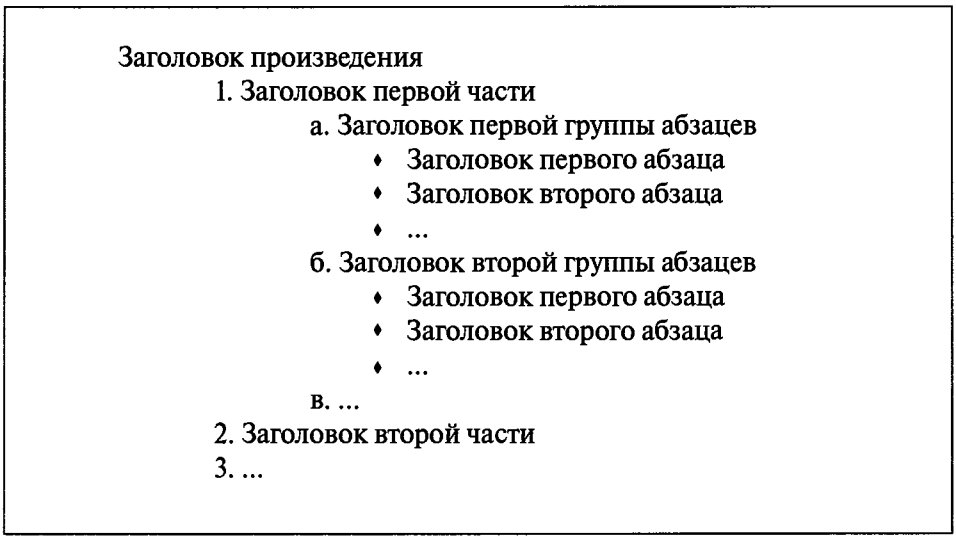


Рис. КП.2 Многоуровневый список

списка на следующий уровень. Правда, может оказаться, что символы нумерации и маркировки будут совсем не те, какие вы планировали. Чтобы поменять их, нужно в меню *Формат* выбрать пункт *Список*, а в открывшемся диалоговом окне выбрать режим *Многоуровневый*. Кнопка *Изменить* в окне этого режима позволит вам в диалоге с компьютером сформировать список таким, каким вы бы хотели его видеть. ◀

2 С помощью режима *Многоуровневый* добейтесь того, чтобы ваш план выглядел так, как показано на рисунке КП.2.

3 Поэкспериментируйте с этим режимом и выберите такой вариант оформления, который вам больше всего нравится. Обсудите с одноклассниками ваше решение. Для большей выразительности можете использовать шрифтовые выделения.

4 Выберите кегль, межстрочный интервал и формат страницы таким, чтобы ваш план занимал не менее двух страниц. Впрочем, так оно, скорее всего, и будет, если выбрать кегль 14 пунктов и полуторный интервал между строками.

5 Включите автоматическую нумерацию страниц. Обсудите с одноклассниками, следует ли ставить номер на первой странице.

6 Создайте верхний колонтитул. Для этого воспользуйтесь пунктом *Колонтитулы* в меню *Вид*. Сделайте так, чтобы на

четных страницах стояла ваша фамилия, а на нечетных — заголовок «План ...».
Создание информационного текстового объекта закончено. Сохраните его в виде файла.

7 Теперь создайте еще один информационный объект. Попытайтесь, используя составленный вами план, передать в 3—4 строках содержание произведения. Как следует, по вашему мнению, назвать созданный вами информационный объект? Подберите для него подходящий шаблон документа.

8 Еще раз просмотрите составленный вами план. Постарайтесь для каждого пункта, отмеченного маркером (т. е. для каждого абзаца исходного текста), подобрать одно слово — имя существительное, — которое, на ваш взгляд, отражает суть этого пункта. Составьте из этих слов сплошной текст, разделив их точкой. Как бы вы озаглавили получившийся информационный объект? Сохраните созданный вами объект в файле с подходящим именем.

9 Выполните такое же задание, подбирая для каждого абзаца характеризующий его глагол. Как, по вашему мнению, можно озаглавить такой информационный объект? Сохраните созданный вами объект в файле с подходящим именем.

Лабораторная работа № 8 (к § 27)

Вставка объектов в текст

Выполнив лабораторную работу № 7, вы создали текстовый информационный объект — план художественного произведения. Усовершенствуем этот объект.

1 Создайте таблицу по образцу таблицы КП.5.

2 В правом столбце в ячейке той же строки, где стоят маркированные пункты плана, запишите цитату из произведения, которая, на ваш взгляд, наиболее точно отражает содержание этой группы абзацев.

3 Посмотрите, соответствует ли смысл цитаты заголовку группы абзацев. Если не соответствует, то скорректируйте заголовок группы абзацев.

4 Сохраните получившийся информационный объект под другим именем.

▶ А теперь от литературы перейдем к физике и создадим небольшой справочник законов и формул. Такой справочник тоже удобно оформить в виде таблицы. Например, такой, как таблица КП.6. ◀

Таблица КП.6

План	Цитата
1. Заголовок первой части	
а. Заголовок первой группы абзацев	
♦ Заголовок первого абзаца ♦ Заголовок второго абзаца ♦ ...	
б. Заголовок второй группы абзацев	
♦ Заголовок первого абзаца ♦ Заголовок второго абзаца ♦ ...	
в. ...	
2. Заголовок второй части	
...	

5 Создайте таблицу КП.7 в текстовом процессоре Word. В эту таблицу вы можете добавить еще несколько формул, которые вам лично интересны.

Таблица КП.7

№ п/п	Название закона	Основная формула	Производные формулы
1	Закон всемирного тяготения Ньютона	$F = k\frac{m_1 m_2}{R^2}$	$R = \sqrt{k\frac{m_1 m_2}{F}}$
2	Закон Кулона	$F = k\frac{q_1 q_2}{R^2}$	$R = \sqrt{k\frac{q_1 q_2}{F}}$
3	Закон теплообмена	$Q = cm(t_2 - t_1)$	$t_2 = \frac{Q}{cm} + t_1$
4	Первое начало термодинамики для одномолекулярного газа	$U_2 - U_1 = \frac{3}{2}\upsilon R(t_2 - t_1)$	$t_2 - t_1 = \frac{2}{3} \cdot \frac{U_2 - U_1}{\upsilon R}$
5	Релятивистское изменение массы	$m = \frac{m_0}{\sqrt{1 - \frac{\upsilon^2}{c^2}}}$	$\upsilon = c\sqrt{1 - \left(\frac{m_0}{m}\right)^2}$

Лабораторная работа № 9 (к § 28)

Создание гиперссылок в тексте

Таблица с планом и сопровождающими его цитатами весьма удобна, если затем по этому плану писать сочинение. Но как самостоятельный информационный объект такая таблица выглядит не очень изящно. Хорошо бы и план не перегружать дополнительной информацией, и цитаты подтверждающие получать, как только в них возникнет потребность.

Поступим так. Создадим отдельный файл с цитатами. Затем в файле с планом произведения создадим гиперссылки на цитаты. Как только потребуется цитата, достаточно будет щелкнуть по гиперссылке. Приступим к реализации этой идеи.

1 Загрузите файл с таблицей, содержащей цитаты. Выделите цитаты из таблицы (для этого можно в меню *Таблица* воспользоваться пунктом *Преобразовать таблицу в текст*). Соберите их в отдельный документ и сохраните его.

2 Загрузите файл с планом, составленным вами во время выполнения лабораторной работы № 7.

3 Для каждого заголовка группы абзацев создайте закладку и свяжите ее гипертекстовой ссылкой с соответствующей цитатой.

4 Проверьте, как работают созданные вами гиперсвязи.

5 А теперь таким же способом создайте гиперсвязи между подборкой существительных, которую вы записали во время лабораторной работы № 7, и заголовками абзацев в плане произведения.

Лабораторная работа № 10 (к § 29 и 30)

Знакомство с HTML

Выполняя эту работу, вы создадите свою первую HTML-страницу. Тема не так уж важна. Это может быть ваш класс, город или просто ваша личная страница. Продумайте, что вы хотите поместить на первой странице, а что будет на следующих.

На первой странице можно расположить следующую информацию:

- название;
- девиз;
- о себе;
- мой класс (переход на 2-й уровень);
- мои интересы (переход на 2-й уровень);
- мое фото.

Вы можете расширить этот список или убрать из него что-то по своему вкусу. Для удобства полезно около каждого пункта пометить, какими командами языка HTML вы будете пользоваться для его создания. Не забудьте, что подобные страницы преследуют чисто *информационные* цели, и, как правило, ни к чему демонстрировать изощренные технологии и громоздкие ненужные красивости. Составленный вами план, пусть даже очень краткий, оформите и сохраните в виде текстового файла.

А теперь приступим к его реализации. Для оформления страницы вам понадобится графический файл с какой-нибудь картинкой, которая на вашей странице станет фоном, какая-нибудь фотография, относящаяся к теме страницы.

1 Создайте собственный каталог и скопируйте в него файлы с картинкой и фотографией. (Ниже мы условно назвали эти файлы как KLEN.GIF и DEER.JPG. У вас они, скорее всего, будут иными.)

2 С помощью Блокнота создайте текстовый файл следующего содержания:

```
<HTML>
<HEAD><TITLE> Моя первая страница </TITLE></HEAD>
<BODY BACKGROUND="KLEN.GIF">
<CENTER>
<FRONT SIZE=5>
<B>
Добро пожаловать на страничку
</B>
<BR>
</FONT>
<IMG SRC="DEER.JPG">
</CENTER>
</BODY>
</HTML>
```

3 Сохраните текстовый файл с расширением HTM или HTML.

4 Посмотрите, как выглядит ваша страница в окне браузера.

5 Пользуясь описанием тегов, приведенных в учебнике, добавьте на страницу надписи, выполненные разными видами шрифтов и различным цветом.

6 Добавьте на страницу еще несколько картинок.

7 Поэкспериментируйте с атрибутами <HEIGHT> и <WIDTH>. Что произойдет, если попытаться применить их одновременно?

Лабораторная работа № 11 (к § 31 и 32)

Использование тега <Table> для формирования HTML-страницы. Использование документов, подготовленных в Microsoft Word?

1 Расположите на своей странице любые две картинки или фотографии так, как на рисунке 3.6, с учетом требований задания 4 из § 31.

2 Реализуйте проект, намеченный вами в лабораторной работе № 9. Используйте таблицу для того, чтобы аккуратно разместить в нем ссылки и другую информацию.

3 Разместите на одной из ваших страниц какой-либо подготовленный в документе Word. Сделайте это двумя способами: используя технологию ActiveX и путем преобразования документа в HTML-формат. Сравните работу браузера в обоих случаях.

4 Оцените, насколько удачным получился HTML-документ, созданный в автоматическом режиме. Если необходимо, подправьте его, добавив фон и другие элементы оформления.

Лабораторная работа № 12 (к § 34)

Знакомство с Adobe Photoshop

Взаимодействие любого программного продукта с пользователем определяется интерфейсом этого продукта. Поэтому освоение Adobe Photoshop мы предлагаем начать со знакомства с его интерфейсом, с тем, как осуществляется доступ к функциям программы посредством выбора команд в меню или нажатием комбинаций клавиш на клавиатуре.

1 Загрузите Adobe Photoshop.

▶ Окно программы имеет типовое для Windows оформление.

Панель *Инструменты* состоит из кнопок. Щелчок по кнопке включает соответствующий ей инструмент. Если на кнопке, имеющей внизу маленький треугольник, подольше подержать мышь с нажатой клавишей, то появится всплывающая панель с кнопками дополнительных родственных инструментов.

Панель *Параметры*, располагающаяся в верхней части экрана, служит для настройки параметров используемого в данный момент инструмента.

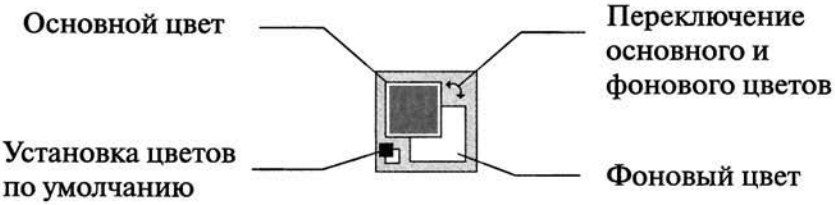


Рис. КП.3 Переключатель основного и фонового цветов

В панели инструментов находятся образцы **основного** и **фонового** цветов (рис. КП.3). По умолчанию это черный и белый цвета. Образец основного цвета на верхнем квадрате. Этим цветом закрашивают точки изображения команды заливки и инструменты рисования. Фоновым цветом закрашиваются точки после удаления фрагментов изображения. Двунаправленная изогнутая стрелка в верхнем правом углу около этих квадратов позволяет поменять образцы местами.

В любой момент вы можете произвольно менять основной и фоновый цвета. Возврат к цветам по умолчанию — щелчок по кнопке слева в виде двух маленьких квадратиков черного и белого цветов.

На рабочем поле редактора (по умолчанию в его правой части) располагаются **палитры.** Палитры — это диалоговые окна, предназначенные для задания параметров и выбора режимов работы. В одном диалоговом окне может быть несколько вкладок — отдельных палитр. Переход между ними осуществляется щелчком по ярлыку с названием. Вызов палитры или их удаление с рабочего поля осуществляется командами из меню *Окно*.

Кнопки режимов окна программы в нижней части панели инструментов доступны, если имеется открытый документ. Левая предназначена для возврата в привычный стандартный вид окна программы. Две другие открывают окно программы на весь экран, эти режимы работы удобны для просмотра больших рисунков или рисунков в увеличенном масштабе.

Для создания рисунка используют команду меню *Файл / Новый*. В открывшемся диалоговом окне задают размеры листа, разрешение, цветовой режим, цвет фона. Для редактирования уже готового рисунка используют команду меню *Файл / Открыть*. Для получения изображения со сканера или цифровой фотокамеры — команду меню *Файл / Импорт*.

В Adobe Photoshop существует несколько способов выбора текущих цветов.

1-й способ. Выбор основного цвета в палитре *Образцы* осуществляется щелчком левой клавишей по выбранному цвету, а выбор фонового цвета — Ctrl + щелчок левой клавишей.

2-й способ. В палитре *Цвет* сначала выберите редактируемый цвет (основной или фоновый). Квадрат редактируемого цвета будет очерчен двойной рамкой. Грубый выбор цвета осуществляется щелчком левой клавишей в нижней цветовой полосе, затем с помощью движков производится точная настройка цвета.

3-й способ. Выбор цвета на рисунке выполняется непосредственно на изображении при помощи инструмента *Пипетка* — . Щелчок левой клавишей по выбранному цвету в изображении меняет основной цвет на заданный, а щелчок левой клавишей при нажатой клавише Alt меняет цвет фона.

4-й способ. Щелчок по образцам основного или фонового цвета в панели инструментов (см. рис. КП.3) выводит на экран палитру соответственно для выбора основного или фонового цвета. ◀

2 Создайте какой-нибудь несложный рисунок при помощи различных инструментов для рисования. Восемь пиктограмм рисующих инструментов расположены единой группой, второй сверху.

▶ Любой выбранный инструмент перед использованием необходимо настроить с помощью кнопок и раскрывающихся списков на панели *Параметры*. Например, вы можете выбирать размер и форму кисти в раскрывающемся списке *Кисть* (рис. КП.4), в нем справа имеется кнопка вызова меню для настройки этой панели и формы кистей. Таким же образом на панели *Параметры* можно выбрать *режим*, *прозрачность* и *нажим* инструмента кисть. ◀

3 Поэкспериментируйте с различными формами кисти и режимами наложения цветов. При затруднениях поищите ответы в справочной системе программы.

▶ Если что-то не понравилось, напоминаем, что последнее выполненное действие всегда можно отменить командами меню *Редактирование / Отменить*.

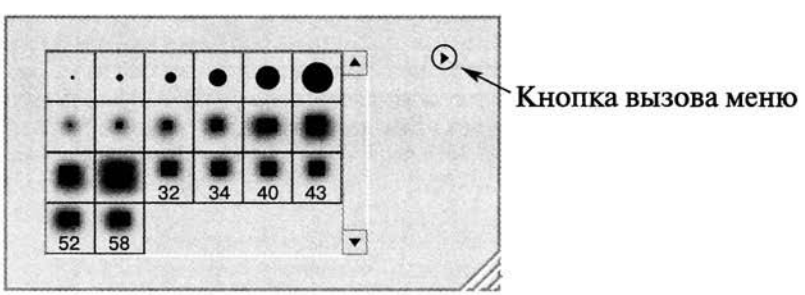


Рис. КП.4 Панель *Форма кисти*

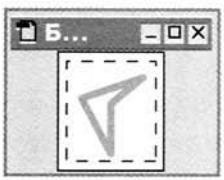


Рис. КП.5

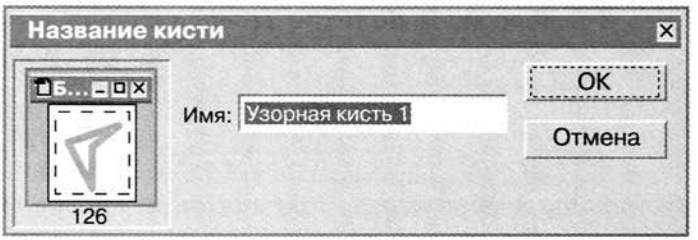


Рис. КП.6

Можно создать собственную кисть. Вот как это делается.

1 Нарисуйте кистью или карандашом (создавая отрезки щелчками левой клавишей при нажатой клавише Shift) простую фигуру (рис. КП.5).

2 Создайте вокруг нее прямоугольное выделение.

3 Выполните команду меню *Редактирование / Определить установки кисти...* В открывшемся диалоговом окне можно набрать произвольное имя кисти и оценить создаваемый элемент слева в поле просмотра (рис. КП.6). Число в поле просмотра означает размер кисти. ◀

4 Создайте кисть. Порисуйте созданной кистью (рис. КП.7). Можете сохранить созданную кисть в составе существующего комплекта кистей или создать новый комплект из ваших собственных кистей, открыв список кистей и выбрав в меню настроек формы кистей команду *Сохранить кисти*. Восстановите стандартный набор кистей командой *Восстановить кисти*.
Попробуем создать что-нибудь полезное.

5 Обратившись к меню *Файл / Новый*, создайте новый рисунок. Задайте его характеристики: *Размер*: 10 × 10 см; *Разрешение*: 28 пикс/см (или, что то же самое, 72 пикс/дюйм); *Цветовой режим*: RGB; *Содержимое фона*: Белый.

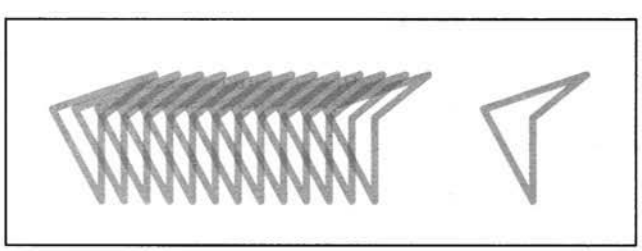


Рис. КП.7 Рисунок созданной кистью

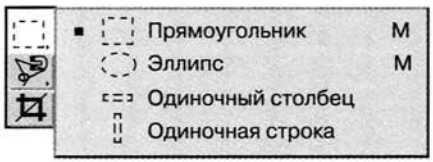


Рис. КП.8 Простые инструменты выделения

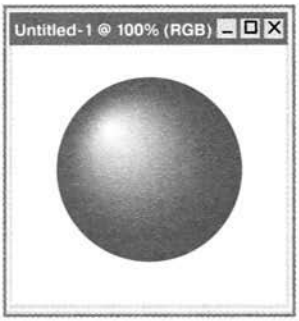


Рис. КП.9

▶ А теперь поговорим об инструментах выделения, которыми вы будете пользоваться неоднократно. В Adobe Photoshop имеется целый набор инструментов и функций, которые позволяют не только выделять фрагменты, но и выполнять определенные действия над ними.

При удержании мыши на верхней левой кнопке в панели инструментов открывается панель с простейшими инструментами выделения (рис. КП.7). Однократный щелчок в ней выбирает (активизирует) инструмент.

Если выбрать инструмент выделения *Прямоугольная область* и задать на панели *Параметры* параметру *Стиль* значение *Нормальный*, а параметру *Растушевка* значение 0 px, то можно при нажатой клавише мыши выделить фрагмент произвольного размера, протягивая мышь по диагонали выделяемой области. Созданную область можно затем перемещать по листу, установив курсор мыши внутри этой области и перемещая его при нажатой левой клавише мыши. Сказанное справедливо, если на панели *Параметры* включен режим *Новая выделенная область*.

Значение параметра *Растушевка* определяет четкость граничных линий, отсутствие размытия границ. При увеличении значения этого параметра границы прямоугольной области будут скруглены, а при ее закрашивании края будут размыты. Ненулевое значение параметра *Растушевка* позволяет смягчить границы выделенной области, с тем чтобы сделать менее резким и заметным переход между изображением в выделенной области и окружающим рисунком. Особенно это важно при вклеивании изображения из другого рисунка.

Если для параметра *Стиль* выбрать значение *Фиксированный размер*, то необходимо указать ширину и высоту инструмента выделения в пикселях. Каждый раз область выделения указанного размера будет строиться автоматически. ◀

6 Задайте основной цвет зеленый, а фоновый — белый.

7 Выберите инструмент выделения *Овальная область* и задайте для него следующие параметры: *Растушевка*: 0 px; *Стиль*: Нормальный, для параметра *Сглаживать* поставьте галочку.

8 Нарисуйте область выделения в виде окружности, для этого во время рисования нужно держать нажатой клавишу Shift.

9 Выберите инструмент *Градиент*  . Задайте для него следующие параметры: *Вид градиента*: Радиальный градиент; *Режим*: Нормальный; *Непрозрачность*: 100%, для параметра *Инверсия* поставьте галочку. Проведите курсором мыши изнутри круга к его границе, нажав левую клавишу мыши. У вас должен получиться зеленый шарик (рис. КП.9).
Если для параметра *Инверсия* галочка не поставлена, то нужно в качестве основного цвета выбрать белый, а в качестве фонового — зеленый.

10 Уменьшите размер рисунка, использовав команду меню *Изображение / Размер изображения...* В открывшемся диалоговом окне *Размер изображения* измените показанные ширину и высоту до 1 см.

11 Сохраните рисунок.

▶ Получился хороший элемент для оформления, например, списка или кнопки. ◀

Лабораторная работа № 13 (к § 34)

Работа со слоями

Для выполнения действий над слоями имеется специальная палитра, вызов которой осуществляется через команду меню *Окно / Слои*. В палитре имеется список слоев с их уменьшенными изображениями. Справа от изображения — название слоя. Редактировать можно только один активный слой. Его строка в палитре подсвечена синим цветом. Для активизации другого слоя следует щелкнуть по строке с его названием в палитре.

Вы будете редактировать уже готовые изображения. Поэтому надо иметь два-три файла с рисунками или фотографиями. Начнем с простого задания — создадим вокруг имеющегося рисунка романтический фон.

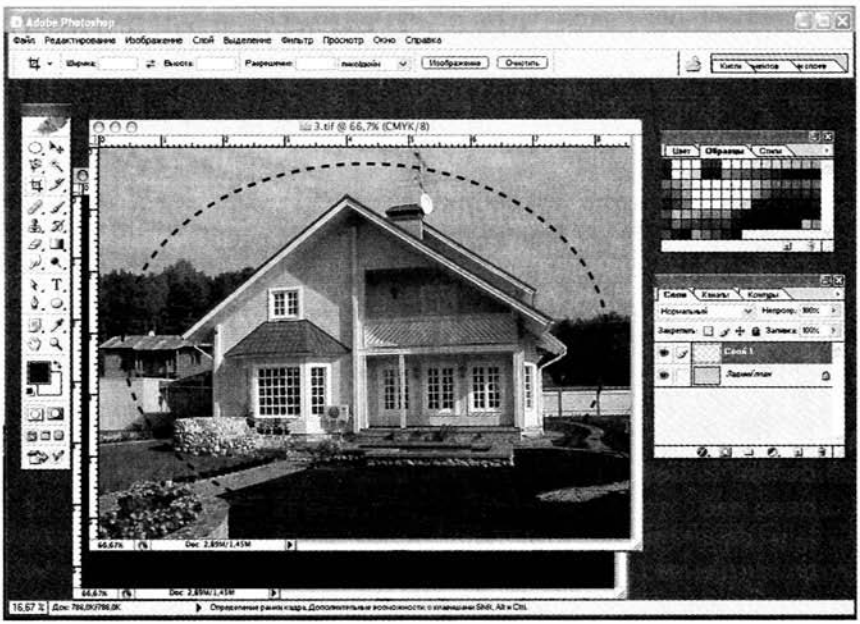


Рис. КП.10 Выделение области

1 Откройте файл с каким-либо рисунком или фотографией.

2 Выберите инструмент выделения *Эллипс*. На панели *Параметры* задайте: *Растушевка*: 15 пикс; *Стиль*: Нормальный, для параметра *Сглаживать* поставьте галочку.

3 Постройте эллиптическую область выделения на фотографии, перемещая мышь при нажатой левой клавише по диагонали воображаемого описанного прямоугольника (рис. КП.10).

4 Скопируйте выделение в буфер через меню *Редактирование / Скопировать*.

5 Создайте новый рисунок размером не менее размера фотографии. (Истинный размер фотографии можно посмотреть, использовав меню *Изображение / Размер холста...*) При создании задайте следующие характеристики: *Разрешение*: 72 пикс/дюйм, *Цветовой режим*: RGB; *Содержимое фона:* Белый.

6 Выберите основной цвет (например, зеленый).

7 Выберите инструмент *Заливка* и закрасьте лист.

Рис. КП.11

8 Вызовите на экран палитру слоев (*Окно / Показать слои*).

9 Выполните команду меню *Редактирование / Вклеить*.

10 Получилось изображение сложной структуры (рис. КП.11). Объедините все его элементы на одном листе — одном слое командой меню *Слой / Объединить видимые*.

11 Сохраните рисунок, используя команду меню *Файл / Сохранить...*, в открывшемся диалоговом окне задайте *Имя файла* (например, рисунок 1). В поле *Тип файлов* выберите формат сохранения jpeg (*.jpg, *.jpe).

12 В открывшемся диалоговом окне *Параметры JPEG* задайте характеристики сохранения: *Качество:* 5 (среднее); *Разновидность формата*: Baseline Optimized («Оптимизированный»).

13 Определите размер получившегося файла.

14 Используя механизм слоев, попытайтесь сделать вид из окна. Для этого именно контуры окна должны быть созданы в отдельном слое.

Лабораторная работа № 14 (к § 35)

Редактирование фотографий

Поработаем теперь над улучшением цифровой фотографии.

1 Загрузите файл с фотографией, требующей редактирования.

2 Измените яркость рисунка командой *Изображение / Коррекция / Уровни...* Посмотрите, как изменяется фотография.

▶ Рисунок можно отретушировать, для этого можно использовать несколько способов. Вот два из них.

1-й способ. Выберите инструмент *Палец*, настройте его параметры: *Кисть:* 13 пикс.; *Режим:* Нормальный; *Интенсивность:* 50%. Протяжкой мыши по отдельным участкам определенного цвета уберите чрезмерную резкость рисунка.

2-й способ. Выберите инструмент *Многоугольное лассо*. Перемещая мышь вдоль граничной линии и щелкая левой клавишей в точках изгиба, получите выделение корректируемой области. Примените к выделенной области фильтр (меню *Фильтр / Резкость*). ◀

3 Отретушируйте свою фотографию.

▶ Возможно, вас все еще не устраивает качество изображения. В этом случае можно воспользоваться **гамма-коррекцией**. Она служит для настройки тонального изображения на отдельных участках цветового диапазона. Гамма-коррекцию осуществляют в диалоговом окне, вызываемом командой меню *Изображение / Коррекция / Кривые...*

В появившемся диалоговом окне *Кривые* изображен график, на котором по горизонтальной оси отображаются входные значения яркости, а по вертикальной — выходные. Первоначально это прямая линия. Коррекция осуществляется установкой нелинейной зависимости, в результате которой некоторые участки входного диапазона яркостей сужаются в выходном диапазоне и за счет этого происходит расширение других участков. Проблема состоит в определении этих участков.

Чтобы определить расположение участка, который следует корректировать, нужно при открытом окне *Кривые* установить курсор мыши на исправляемый участок, перемещать по нему мышь при нажатой левой клавише. При этом на графике по линии будет перемещаться кружочек, показывающий тональный диапазон этого участка рисунка.

Щелчками на линии устанавливаются предельные точки изменяемого диапазона. Можно зафиксировать до 15 точек. Лишние точки удаляются перемещением их по линии за пределы графика. Для корректировки диапазона нужно щелкнуть в центре диапазона на линии и при нажатой левой клавише переместить ее в нужном направлении до достижения приемлемого результата. У вас может получиться, например, такая кривая, какая показана на рисунке КП.12. ◀

4 Выделите редактируемый участок при помощи любого инструмента выделения. Настройте тональность изображения, используя гамма-коррекцию.

5 Сохраните отредактированную фотографию в формате jpeg.

6 Задайте в режиме гамма-коррекции какой-либо экзотический график, например с несколькими разрывами. Посмотрите, как это отразилось на выделенном фрагменте изображения. Такие парадоксальные соединения красок тем не менее могут быть использованы для создания декоративных полотен.

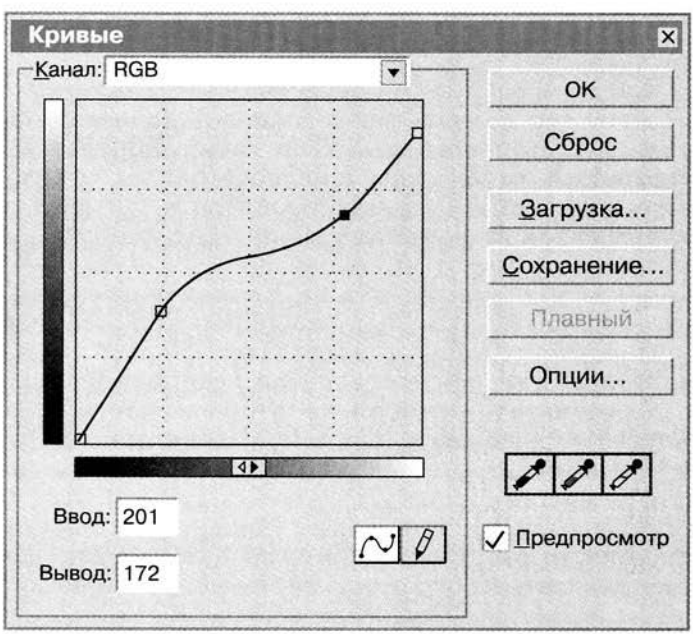


Рис. КП.12 Окно режима гамма-коррекции

Лабораторная работа № 15 (к § 36)

Создаем презентацию в PowerPoint

Выполнив задание 5 из § 36, вы реализовали первый этап разработки презентации. Теперь очередь за следующими этапами.

1 Подготовьте в каком-либо графическом редакторе иллюстрации для вашей будущей презентации.

2 Создайте последовательно слайды и определите порядок их появления в презентации.

3 Выполнив задание 6 из § 36, вы подготовили проект рекламы некоторого продукта. Создайте эту рекламу с помощью PowerPoint. Устройте конкурс получившихся реклам среди своих одноклассников.

4 Создайте «Живую азбуку» для первоклассников. Здесь вы в полной мере сможете проявить свои навыки работы в Adobe Photoshop, комбинируя рисунки, фотографии, текст.

Лабораторная работа № 16 (к § 37 и 39)

Знакомимся с компьютерными сетями

Начиная с этой лабораторной работы, вы будете работать с сетевыми ресурсами. Мы не рекомендуем вам самостоятельно (без указания учителя) изменять настройки сети, добавлять или удалять ее компоненты, изменять их свойства. Нарушение этих правил может привести к отключению вашего компьютера от сети и другим неприятностям.

1 Выберите одну из ваших папок на том компьютере, где вы работаете. Например, это может быть папка с проектом «Живая азбука», который вы разрабатывали на предыдущей лабораторной работе. Щелкните правой клавишей мыши и в появившемся контекстном меню выберите пункт *Общий доступ и безопасность...* (можно выбрать пункт *Свойства*, а затем в появившемся окне выбрать закладку *Доступ*). Установите флажок в окошке «Открыть общий доступ к этой папке». Если при этом вы хотите, чтобы файлы этой папки могли обрабатываться другими пользователями сети, то установите флажок в окошке «Разрешить изменение файлов по сети». Но в целях безопасности мы не советуем вам это делать. После установки всех параметров доступа значок объекта несколько меняет свой облик —

внизу появляется изображение руки. Это означает, что данный объект является общедоступным для других пользователей сети.

Поскольку вы делаете лабораторную работу все вместе, то теперь можно посмотреть содержание доступных папок на других компьютерах сети. И не надо перемещаться от компьютера к компьютеру, чтобы сравнивать созданные вами проекты.

▶ Теперь познакомимся с тем, как «чувствует» себя ваш компьютер в глобальной сети Интернет. Вы уже знаете, что во время работы в Интернете вашему компьютеру присвоен уникальный IP-адрес. Сейчас вы узнаете, какой именно. Для версий Windows NT/2000/XP нужно предпринять следующие действия. ◀

2 В меню *Пуск* выберите строку *Выполнить* и в появившемся окне введите символы *cmd*. В результате запустится программа *Командная строка* — это служебная программа для наблюдения и настройки компьютера и сети. Если в появившемся окне ввести команду ipconfig, то вы увидите полную информацию о подключении к Интернету, в том числе и IP-адрес вашего компьютера. Вполне возможно, что вы увидите не постоянный, а динамический IP-адрес, и при следующем подключении к Интернету он окажется иным.

▶ Обычно к Интернету подключается не один отдельно взятый компьютер, а некоторая локальная сеть, в которую входит ваш компьютер. Тогда естественно, чтобы IP-адреса всех компьютеров такой сети образовывали некий единый блок. Чтобы выделить некоторое множество адресов для одной локальной подсети, используют **маски** IP-адресов. Посмотрите еще раз на экран вашего компьютера с информацией о подключении, и вы увидите слова «Маска подсети». Она имеет такую же структуру, что и IP-адрес. Если записать маску в двоичном коде, то получится 32-символьное двоичное число. Те биты, где в маске стоят единицы, являются общими для всех IP-адресов данной подсети. Остальные биты могут варьироваться для создания индивидуальных IP-адресов компьютеров, входящих в данную подсеть. ◀

3 Определите по маске IP-адресов, какой диапазон адресов предоставлен для компьютеров той локальной сети, в которой вы работаете. Вычислите, какое максимальное число компьютеров может быть в вашей локальной сети.

▶ Вы можете увидеть еще один код, имеющий структуру IP-адреса, и рядом слова «Основной шлюз» (в англоязычном вариан-

те Gateway). На самом деле это IP-адрес того компьютера, который обеспечивает взаимодействие различных подсетей (как было сказано в § 38, наличие шлюзов — один из принципов объединения различных сетей в Интернете). Если сетевая программа по IP-адресу и маске определит, что компьютер, которому предназначено послание, не входит в локальную подсеть, то она отправит его на этот компьютер для передачи в глобальную сеть.

Подключение к Интернету может произойти только после того, как определены все три параметра: IP-адрес, маска подсети и основной шлюз.

Познакомимся теперь с маршрутными и скоростными характеристиками глобальной сети. Пусть мы хотим узнать, через какие узлы и как быстро происходит соединение вашего компьютера с компьютером какой-либо поисковой системы, например Rambler.◀

4 Введите команду tracert www.rambler.ru. На экране появится список узлов (обычно 5—7), через которые проходит информация, и время передачи между узлами. Проведите еще несколько экспериментов с адресами других host-компьютеров, например www.google.ru или www.microsoft.com.

▶ Возможно, что в представленном вам протоколе прохождения узлов в некоторых строках стоят звездочки. Если звездочка одна, то это означает, что ответ от этого узла не получен в отведенное время ожидания; если же в строке нет ничего, кроме звездочек, это означает, что администраторы этих хостов не разрешили прохождение через данный узел и пришлось искать обходной путь. ◀

5 Определите по протоколу, в каких узлах скорость передачи информации самая низкая.

Лабораторная работа № 17 (к § 41)

Путешествие по страницам Интернета

Начнем с простого: пусть вам известно, *где* взять нужную информацию. Это значит, вы знаете, на каком сервере хранится данная информация, в какой папке (или в каком каталоге) она располагается и, быть может, каково имя файла. Иными словами, вы знаете универсальный указатель ресурса, предоставляющего нужную информацию. Но мы для учебных целей предлагаем воспользоваться следующим адресом:

http://www.hermitage.ru

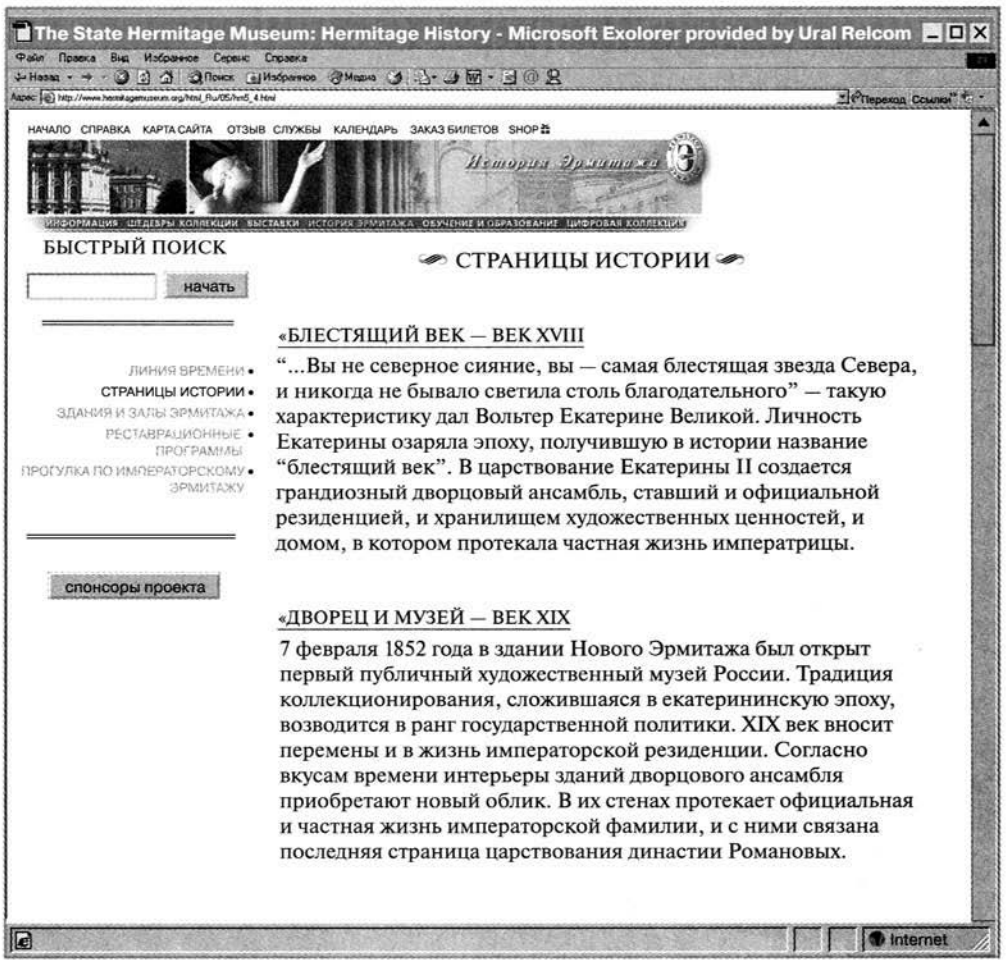


Рис. КП.13 Страница сайта музея Эрмитаж

1 Вызовите имеющийся на вашем компьютере браузер и запишите в строку вызова указанный адрес.
Перед вами главная страница одного из самых известных музеев мира. Пользуясь гиперссылками, просмотрите страницы, на которых говорится об истории этого уникального хранилища (рис. КП.13).
С помощью буфера обмена скопируйте текст из нескольких страниц, посвященных истории музея, и вставьте его в документ какого-либо текстового редактора, например Micro-

soft Word. Дополните текст рисунками. Сохраните получившийся документ под каким-либо именем.
Если вы чувствуете, что полностью разобрались с первым заданием, то переходите ко второму.

2 Запишите в окне вызова браузера следующий адрес:

http://president.kremlin.ru

Выберите наиболее интересные, на ваш взгляд, гиперссылки и посмотрите содержание соответствующих страниц.

3 Найдите описание символики Российского государства — герба и флага. Скопируйте текстовые фрагменты, описывающие историю российской символики, а также сопровождающие изображения.

4 Дополните полученную информацию тем, что вам кажется интересным. Сохраните созданный документ под каким-либо именем.

Лабораторная работа № 18 (к § 40)

Поиск в Интернете

Сегодня вы будете разыскивать информацию с помощью поисковой системы. Для начала мы предлагаем вам найти фотографию Винтона Серфа — как-никак считается отцом Интернета!

1 Вызовите с помощью браузера какую-либо поисковую систему. Можно воспользоваться поисковыми системами, приведенными в § 40. Следуя указаниям, сформируйте запрос на поиск информации о Винтоне Серфе.

▶ Скорее всего, вас постигла неудача — слишком много документов предлагает поисковая система на такой запрос. Это имя встречается в тысячах публикаций. ◀

2 Добавьте в запрос ключевое слово «фото» (без кавычек, разумеется). Если и это не помогло, подскажем, что существует фотография с президентом Клинтоном. Сформируйте нужный запрос.

3 Запишите URL того сайта, где вы нашли подходящую фотографию.

4 Говорят, что в Интернете есть все. Давайте это проверим. Засеките время и попробуйте найти информацию о том, как

решить квадратное уравнение

$$ax^2 + bx + c = 0.$$

5 Наконец, задание посложнее: составьте список школ в вашем городе (или области), имеющих в Интернете свои сайты. Сформируйте нужный запрос и полученные результаты сохраните в виде текстового документа.

Лабораторная работа № 19 (к § 42)

Выбор профессии и трудоустройство через Интернет

Вы заканчиваете школу. Впереди огромный мир разнообразных профессий. Кто-то продолжит обучение в вузе или колледже, кто-то пойдет работать. Чтобы эти первые шаги в новую жизнь были уверенными, надо располагать информацией о том, какие профессии востребованы, какие возможности предоставляют образовательные учреждения, какие требования предъявляют работодатели.

Начнем с изучения рынка труда. Какие специалисты сегодня нужней? Вот адрес сайта Федеральной службы по труду и занятости:

http://www.rostrud.info/press/news

1 Зайдите на этот сайт и выясните спрос на различные профессии. Проследите динамику этого спроса. Составьте с помощью Excel диаграмму изменения спроса за последние 5 лет для нескольких разных профессий (например, программиста, экономиста, журналиста, секретаря-референта и т. д.).

2 Найдите с помощью поисковой системы сайт службы занятости населения вашего региона (например, для уральского региона таким может оказаться сайт с адресом

http://www.sverdl.rostrud.ru/portal/r/main).

Выполните задание, аналогичное заданию пункта 1, для вашего региона. Сравните получившиеся результаты.

3 Интерес к профессии — главный ориентир в будущей карьере. Но и зарплата играет не последнюю роль. Попытайтесь с помощью тех же сайтов (а может быть, и других) проследить, как изменялась оплата труда выбранных вами профессий в течение тех же 5 лет.

4 На многих сайтах региональных новостей есть раздел «Вакансии». В этом разделе работодатели помещают объявления

о том, какие специалисты нужны их предприятию или фирме. Найдите такие объявления.

5 Обычно желающим занять вакансию предлагается тут же заполнить анкету и написать так называемое резюме о самом себе. Скопируйте в текстовом редакторе анкету и вопросы резюме и попытайтесь их заполнить. Обсудите с товарищами по классу и учителем, как лучше написать резюме, чтобы работодатель по достоинству и адекватно оценил ваши возможности.

6 Посмотрев данные о рынке труда, вы, весьма вероятно, придете к выводу, что вам надо продолжить свое образование. С помощью поисковой системы найдите учебные заведения, где готовили бы специалистов по выбранной вами профессии. Не забудьте сформулировать запрос так, чтобы не получить рекомендации уехать за тридевять земель.

7 Выбрав учебное заведение, зайдите на его сайт и уточните правила приема, план приема (т. е. сколько человек будет принято на 1-й курс данным учебным заведением), продолжительность обучения по выбранной вами специальности.

Лабораторная работа № 20 (к § 48)

Исследование алгоритмов и программ

1 Перечитайте условие задачи 2 к § 48. Запишите программу, реализующую этот алгоритм, и отладьте ее.

▶ А теперь давайте поинтересуемся не только показателем степени N, для которого 3^N оканчивается на 00001, но и самим этим числом. Для этого надо в нужное место алгоритма поставить команду **Сообщить** X. ◀

2 Модифицируйте свою программу и попытайтесь получить требуемое число.

▶ Скорее всего, вас постигла неудача — число, которое вы собирались получить, не умещается в разрядной сетке при объявленном типе «целое».

Вывод из проведенного компьютерного эксперимента таков: реализуя алгоритм в виде программы, надо учитывать ограничения, возникающие из физических особенностей вычислительной техники, — ограниченность разрядной сетки, ограниченность памяти, продолжительность вычислительного процесса. Выполняя

лабораторную работу № 6, вы наблюдали влияние ошибок округления.

Тем не менее поставленную задачу — найти значение N, при котором 3^N оканчивается на 00001, — решить все равно нужно. Для этого придется придумать другой алгоритм.

Для нахождения последних пяти цифр степеней числа 3 достаточно вычислять только эти пять цифр. В предложенном алгоритме достаточно поменять одну строку. Вот как выглядит преобразованный алгоритм:

```
Алгоритм Показатель степени-2
цел: X, N;
{  X := 3;
   N := 1;
   Делать пока (X > 1)
   {  X := 3*X;
      X := mod(X, 100000);
      N := N + 1;
   }
   Сообщить N;
}
```

Неудача, постигшая вас при реализации исходного алгоритма, должна была научить осмотрительности — вдруг разыскиваемое число N настолько велико, что и оно не уместится в заданную разрядную сетку. Надеемся, что предложенное вами обоснование применимости исходного алгоритма содержит все необходимые аргументы того, что число N не так уж велико (по компьютерным меркам). ◀

3 Напишите программу, реализующую алгоритм Показатель степени-2, отладьте ее и получите ответ на вопрос задачи.

4 Рассмотрите следующий алгоритм, обрабатывающий натуральное число x:

```
Алгоритм
цел: x, y, z;
{ Запросить x;
  y := x + P(x);
  Делать пока (x ≠ y)
   { z := y + P(y);
     x := y;
     y := z;
   }                          (*конец цикла*)
  Сообщить x;
}
```

В нем используется подпрограмма-функция:

```
Функция P (цел: x): цел
{ Если x < 10 то { знач := x; }
  иначе { знач := P(x div 10)*(x mod 10); }
}
```

Применим ли этот алгоритм к любому натуральному числу x? Ответить на этот вопрос, ограничившись изучением только текста алгоритма, непросто. Давайте поэкспериментируем. Напишите на изучаемом вами языке программирования программу, реализующую данный алгоритм, и отладьте ее. Проведите вычислительный эксперимент, придавая x различные значения (например, 9, 99, 999, 9999 и т. д.). Попытайтесь по результатам эксперимента определить, ограничено ли число исполнений тела цикла в основном алгоритме какой-нибудь константой, не зависящей от x.
А теперь попытайтесь результаты своего эксперимента объяснить теоретически и получить ответ на вопрос: «Верно ли, что при любом натуральном значении x алгоритм заканчивает работу за конечное число шагов?»

5 Предлагаем вам попытаться выполнить еще одно задание. Перечитайте задание 4 из § 48. Записанный там алгоритм никогда не завершит работу — нетрудно доказать, что значение переменной S в ходе выполнения алгоритма никогда не превысит числа 2. Еще в XVIII в. математики установили, что значение S, вычисляемое при помощи этого алгоритма, никогда не превосходит $\frac{\pi^2}{6}$, хотя с увеличением N становится как угодно близким к этому числу. Ваша задача — определить, при каком наименьшем значении N значение переменной S будет отличаться от $\frac{\pi^2}{6}$ не более чем на 0,0000001. Казалось бы, для этого годится следующий алгоритм:

```
Алгоритм Хорошее приближение к π²/6
вещ: S; цел: N;
{ S := 1;
  N := 1;
 Делать пока (S < π*π/6 − 0,0000001)
      { N := N + 1;
        S := S + 1/N²;
      }                                  (*конец цикла*)
 Сообщить N;
}
```

Проведите вычислительный эксперимент с этим алгоритмом. Удалось ли вам найти нужное N? Подумайте и модифицируйте алгоритм так, чтобы решить поставленную задачу. Возможно, для этого вам пригодится информация о том, что при $N = 10\,000\,000$ значение S уже больше, чем $\frac{\pi^2}{6} - 0{,}0000001$.

Лабораторная работа № 21 (к § 52)

Способы представления графов

Чтобы тестировать составляемые алгоритмы обработки графов, надо уметь создавать в памяти компьютера подходящие графы. Пусть требуется граф с n вершинами и m ребрами. Прежде всего оценим число m. Поскольку из каждой вершины выходит не более чем $n - 1$ ребер, а сумма всех степеней вершин равна удвоенному числу ребер, имеем соотношение $2m \leqslant (n - 1)\,n$. Равенство достигается, если каждая вершина графа соединена со всеми остальными вершинами. Такой граф называется **полным**.

Граф будем строить с использованием датчика случайных чисел, одновременно получая для него представление и в виде списка ребер, и в виде таблицы смежности. Ниже приведен алгоритм, в котором через $SR[1{:}m;\ 1{:}2]$ обозначен массив, содержащий список ребер, а через $TS[1{:}n;\ 1{:}n]$ обозначен массив, содержащий таблицу смежности.

```
Алгоритм Случайный_граф
цел: n, m, i, j, k, SR[1:m; 1:2]; TS[1:n; 1:n];
{ Запросить n;                  (* запрашивается количество вершин *)
  Запросить m;                    (* запрашивается количество ребер *)
  Если (m > n*(n - 1)/2)
    то { Сообщить "Такой граф построить нельзя"; }
    иначе
    { Делать от i := 1 до n
      { Делать от j := 1 до n
        { TS[i; j] := 0;
        }
      }          (* Первоначальное заполнение таблицы смежности
                                                          нулями *)
      Делать от i := 1 до m
      { j := INT((n - 1)*rand(1)) + 1;
        k := INT((n - j)*rand(1)) + j + 1;
        Делать пока TS[j; k] := 1
        { j := INT((n - 1)*rand(1)) + 1;
```

```
            k := INT((n − j)*rand(1)) + j + 1;
          }
          TS [j; k] := 1;
          TS [k; j] := 1;
          SR[i; 1] := j;
          SR[i; 2] := k;
        }                          (*случайное добавление одного ребра *)
        Сообщить "Вот таблица смежности";
        Сообщить TS[1:n; 1:n];          (*в реальной программе это
                                  действие оформляется двойным циклом *)
        Сообщить "Вот список ребер";
        Сообщить SR[1:m; 1:2];          (*в реальной программе это
                                  действие оформляется двойным циклом *)
    }
}
```

1 Запрограммируйте этот алгоритм, отладьте его для небольших значений m и n (например, $m = 5$ и $n = 6$; $m = 9$ и $n = 7$ и т. п.).

2 В последующих пунктах вам потребуются графы, созданные случайным образом для разных значений m и n. Поэтому оформите созданную вами программу как подпрограмму, для которой аргументами будут переменные m и n, а результатами — массивы $TS[1{:}n;\ 1{:}n]$ и $SR[1{:}m;\ 1{:}2]$.

3 Запрограммируйте алгоритм, составленный вами при выполнении задания 4а из § 52. Отладьте получившуюся программу, подпрограмму из пункта 2.

4 Запрограммируйте алгоритм, составленный вами при выполнении задания 4б из § 52. Отладьте получившуюся программу, используя подпрограмму из пункта 2.

5 Используя графические операторы, составьте алгоритм рисования графа, заданного списком ребер. С помощью подпрограммы, составленной вами при выполнении пункта 2, сгенерируйте несколько небольших графов и изобразите их на экране компьютера.

6 Используя графические операторы, составьте алгоритм рисования графа, заданного таблицей смежности. С помощью подпрограммы, составленной вами при выполнении пункта 2, сгенерируйте несколько небольших графов и изобразите их на экране компьютера.

7 В предыдущих заданиях графы были ненагруженными. Составьте алгоритм генерации нагруженного графа. Для простоты нагрузку считайте целыми числами и генерируйте с помощью ДСЧ в диапазоне от 1 до 20.

8 Запрограммируйте составленный вами алгоритм и отладьте получившуюся программу. Создайте с помощью этой программы 4 графа. Для генерируемых графов значения n и m возьмите из таблицы КП.8. Оформите составленную вами программу в виде подпрограммы, для которой аргументами будут переменные m и n, а результатами — массивы $TSN[1{:}n;\ 1{:}n]$ (таблица смежности с нагрузкой) и $SNR[1{:}m;\ 1{:}3]$ (список нагруженных ребер).

Таблица КП.8

n	8	15	30	50
m	18	70	300	800

9 Нагруженный граф, имеющий n вершин, задан списком ребер. Составьте алгоритм, создающий по этому списку таблицу смежности. Запрограммируйте этот алгоритм и отладьте получившуюся программу.

10 Нагруженный граф, имеющий n вершин, задан таблицей смежности. Составьте алгоритм, создающий по этой таблице список ребер. Запрограммируйте этот алгоритм и отладьте получившуюся программу.

Лабораторная работа № 22 (к § 53)

Поиск в глубину

Для проведения вычислительных экспериментов мы предлагаем воспользоваться графом с таблицей смежности, представленной таблицей КП.9. Сам граф с последовательной нумерацией вершин представлен на рисунке КП.14. Как видно из рисунка, этот граф связен.

1 Запрограммируйте алгоритм поиска в глубину, предложенный в объяснительном тексте § 53, отладьте программу и с ее помощью выполните обход графа с указанной таблицей смежности. Проставьте на изображении графа порядок прохождения вершин при исполнении данного алгоритма.

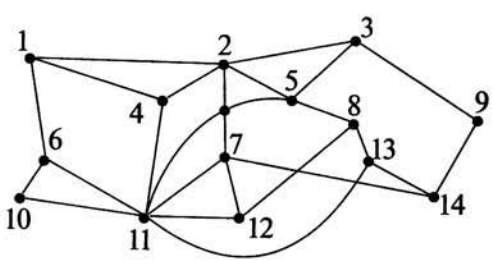


Рис. КП.14

Таблица КП.9

Вершина	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
2	1	0	1	1	1	0	1	0	0	0	0	0	0	0
3	0	1	0	0	1	0	0	0	1	0	0	0	0	0
4	1	1	0	0	0	0	0	0	0	0	1	0	0	0
5	0	1	1	0	0	0	0	1	0	0	1	0	0	0
6	1	0	0	0	0	0	0	0	0	1	1	0	0	0
7	0	1	0	0	0	0	0	0	0	0	1	1	0	1
8	0	0	0	0	1	0	0	0	0	0	0	1	1	0
9	0	0	1	0	0	0	0	0	0	0	0	0	0	1
10	0	0	0	0	0	1	0	0	0	0	1	0	0	0
11	0	0	0	1	1	1	1	0	0	1	0	1	1	0
12	0	0	0	0	0	0	1	1	0	0	1	0	0	0
13	0	0	0	0	0	0	0	1	0	0	1	0	0	1
14	0	0	0	0	0	0	1	0	1	0	0	0	1	0

2 Выполнив задание 2 из § 53, вы модифицировали алгоритм так, чтобы поиск в глубину выполнялся с произвольной вершины. Запрограммируйте составленный вами алгоритм, отладьте получившуюся программу. Выполните поиск в глубину для графа, заданного рисунком КП.14 и таблицей КП.9, начиная с вершин с четными номерами.

▶ В предложенном алгоритме поиска в глубину есть недостаток: возврат к вершинам, в которых не все ребра еще рассмотрены, осуществляется неэффективно, поскольку снова просматриваются все вершины от начала до этой «перспективной» вершины. Было бы хорошо запоминать «перспективную» вершину... Реализуем эту идею. Чтобы хранить список «перспективных» вершин, организуем одномерный массив $St[1{:}n]$.

```
Алгоритм Поиск в глубину с использованием стека
цел: k, m, s, t, u, n, a, G[1:n; 1:n], B[1:n], St[1:n], C[n];
{  Запросить n;
   Запросить G[1:n; 1:n];          (* реально это действие — ввод
         таблицы смежности — оформляется двойным циклом *)
   Запросить m;      (* запрашивается номер исходной вершины *)
   B[1] := m;
   St[1] := m;
   Делать от t := 1 до n
   {  C[t] := 0;
   }                          (* C[t] := 1, если вершина t просмотрена *)
   C[m] := 1;
   k := 1;                    (* счетчик количества пройденных вершин *)
   u := 1;                   (* счетчик количества вершин в массиве St *)
   Делать пока (u > 0)
   {  s := 1;
      Делать от t := 1 до n
      {  Если (G[St[u], t] = 1 и C[t] = 0) то
            {  a := t;
               s := 0;
            }
      }   (* проверка, существует ли вершина, смежная с St[u], кото-
         рая еще не просматривалась: если существует, то s = 0 *)
      Если (s = 0) то
      {  u := u + 1;
         St[u] := a;
         k := k + 1;
         B[k] := a;
         C[a] := 1;
      }
      иначе
      {  u := u - 1;
      }
   }
   Делать от k := 1 до n
   {  Сообщить B[k];
   }
}
```

Как обрабатывается массив St? По мере продвижения в глубину в него заносятся номера пройденных вершин. Как только вершина оказывается тупиковой, т. е. из нее уже нельзя продвинуться дальше к новой вершине, она стирается из массива St (в ячейку заносится 0) и алгоритм начинает работать с предыдущей вершиной. Если и эта вершина оказывается тупиковой, то она тоже стирается и обрабатывается предыдущая вершина и т. д. Количество ненулевых элементов в St, т. е. номеров вершин,

то увеличивается, то уменьшается в зависимости от удаленности от исходной вершины. Все это происходит до тех пор, пока все элементы в St не станут нулевыми. Заметим, что ненулевой элемент, который появляется в St последним, удаляется из него первым. Это напоминает движение патронов в магазине автомата: последний патрон, которым снаряжен магазин, будет первым подан на выстрел. Поэтому такой способ организации данных — последним пришел, первым обработан — называют магазинным, а саму такую структуру данных — **магазином** или, по-другому, **стеком.**

Стек — довольно часто используемая структура данных в различных задачах на алгоритмизацию. Мы продемонстрировали, как организовать стек с помощью массива, и надеемся, что теперь вы будете легко пользоваться этой структурой для решения других задач.

Давайте экспериментально сравним эффективность алгоритма поиска в глубину, предложенного в объяснительном тексте § 53, и алгоритма поиска в глубину со стеком. Для этого в программы надо вставить операторы, фиксирующие время. Напомним, как это делается.

В языке Basic можно воспользоваться функцией TIMER. По команде t = TIMER переменной t будет присвоено значение, равное числу секунд, прошедших от 0 часов текущих суток.

В языке Pascal для вывода времени можно воспользоваться процедурой gettime(h, m, s, ss). В переменной h будет указан текущий час, в переменной m — минуты, в s — секунды, в ss — миллисекунды. ◁

3 Используя программы, разработанные вами при выполнении лабораторной работы № 21, сгенерируйте граф с нужным числом вершин n и числом ребер m (они указаны в таблице КП.10), представив его таблицей смежности.
Проведите вычислительный эксперимент (время на генерацию и преобразование способа представления графа учитывать не следует). По результатам исполнения алгоритмов

Таблица КП.10

n	30	40	45	50
m	300	500	660	800
Время работы алгоритма «Поиск в глубину»				
Время работы второго алгоритма «Поиск в глубину с использованием стека»				

«Поиск в глубину» и «Поиск в глубину с использованием стека» заполните третью и четвертую строки таблицы КП.10. Для каждой пары чисел n и m найдите отношение времени работы первого алгоритма ко времени работы второго алгоритма. Как, на ваш взгляд, можно охарактеризовать это отношение?

4 Выполнив задание 4 из § 53, вы составили алгоритм поиска в глубину для графа, заданного списком ребер. Запрограммируйте ваш алгоритм и отладьте получившуюся программу. С помощью этой программы для графа, изображенного на рисунке КП.14, выполните несколько раз поиск в глубину, начиная с вершин с четными исходными номерами. (Представление этого графа списком ребер получите, либо преобразовав имеющуюся у вас таблицу смежности, либо создав этот список «вручную».)

Лабораторная работа № 23 (к § 53)

Поиск в ширину

Теперь займемся алгоритмом «Поиск в ширину». Для проведения вычислительных экспериментов мы предлагаем воспользоваться графом, который представлен таблицей КП.9. Сам граф с последовательной нумерацией вершин изображен на рисунке КП.14.

1 Запрограммируйте алгоритм поиска в ширину, предложенный в объяснительном тексте § 53, отладьте программу и с ее помощью выполните обход.

2 Выполнив задание 7 из § 53, вы модифицировали алгоритм так, чтобы поиск в ширину выполнялся с произвольной вершины. Запрограммируйте составленный вами алгоритм, отладьте получившуюся программу и выполните несколько раз поиск в ширину для того же графа, начиная с вершин с четными исходными номерами.

3 Выполнив задание 8 из § 53, вы составили алгоритм поиска в ширину для графов, заданных списком ребер. Запрограммируйте этот алгоритм. Преобразуйте таблицу смежности графа в его представление списком ребер и отладьте программу.

4 Проведите эксперименты со своими программами, чтобы выяснить, сколько времени требуется на обход каждого из графов, который фигурировал в задании 4 лабораторной работы № 22. По результатам эксперимента заполните таблицу, аналогичную таблице КП.10.

Лабораторная работа № 24 (к § 53)

Волновой алгоритм

1 Выполнив задание 10 из § 53, вы усовершенствовали волновой алгоритм, предложенный в объяснительном тексте. Модифицируйте его так, чтобы он был применим к графу с любым наперед заданным количеством вершин, а не только к графу с двадцатью вершинами. Запрограммируйте составленный алгоритм и отладьте получившуюся программу. Для отладки воспользуйтесь графом, представленным на рисунке КП.14.

2 Составьте алгоритм и напишите программу, которая позволяла бы для каждой вершины графа находить расстояние до самой удаленной от нее вершины. Получите список, в котором для каждой вершины указана самая от нее далекая вершина и расстояние между ними (если наиболее удаленных вершин несколько, то должны быть указаны все). Используйте волновой алгоритм в качестве вспомогательного. Для отладки можно воспользоваться графом, представленным на рисунке КП.14.

3 Исполнение алгоритма, составленного вами в пункте 2, позволяет каждой вершине связного графа сопоставить число — расстояние до самой далекой вершины. Вершина графа, для которой расстояние до самой удаленной вершины минимально, называется **центром графа**, а расстояние от центра до самой удаленной вершины называется **радиусом графа**. Составьте алгоритм, позволяющий находить центр графа и радиус графа. Имейте в виду, что у графа может оказаться несколько центральных вершин. Например, у графа, изображенного на рисунке КП.15, *а*, их четыре, а у графа, изображенного на рисунке КП.15, *б*, все вершины центральные. Ваш алгоритм должен находить все центральные вершины графа. Для отладки программы воспользуйтесь графами, изображенными на рисунках КП.14 и КП.15.

а) б)

Рис. КП.15

4 Для задач, рассмотренных в пунктах 1—3 данной лабораторной работы, запрограммируйте алгоритмы, обрабатывающие графы, представленные списками ребер.

5 Запрограммируйте алгоритмы, которые вы составили, выполнив задание 12 из § 53. Отладьте получившиеся программы на графах, изображенных на рисунке КП.16. Рассмотрите вершины, принадлежащие разным компонентам связности.

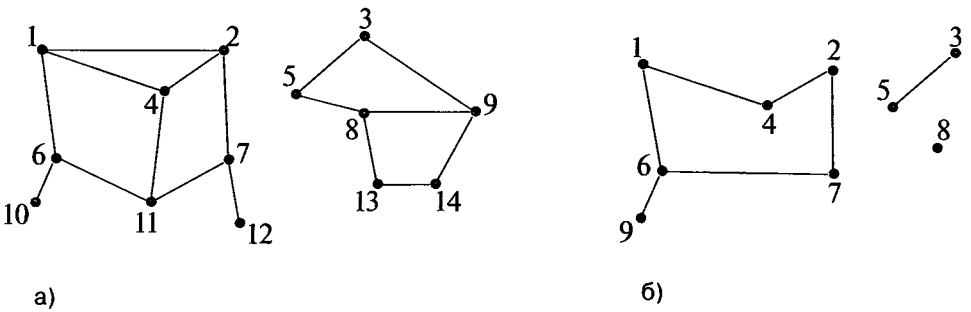


Рис. КП.16

6 Запрограммируйте алгоритмы, которые вы составили, выполнив задание 13 из § 53. Отладьте получившиеся программы на графах, изображенных на рисунке КП.16.

7 Измените программу, составленную вами при выполнении предыдущего пункта, так, чтобы она сообщала только количество связных компонент. Отладьте модифицированную программу.
Сгенерируйте 4 графа в соответствии с данными таблицы КП.11. Найдите для каждого из них количество компонент связности.

Таблица КП.11

Число вершин	8	15	30	50
Число ребер	9	25	80	150

8 Запрограммируйте алгоритмы, которые вы составили, выполнив задание 14 из § 53. Оформите составленную вами программу в виде подпрограммы — она вам понадобится при выполнении лабораторных работ № 25 и 26.

9 Запрограммируйте алгоритм, который вы составили, выполнив задание 15 из § 53. Отладьте получившуюся программу на одном из графов, который вы можете создать, используя подпрограмму, написанную при выполнении задания 8 из лабораторной работы № 21.

10* Запрограммируйте алгоритм, который вы составили, выполнив задание 16 из § 53, и отладьте получившуюся программу.

Лабораторная работа № 25 (к § 54)

Мосты и точки сочленения

1 Выполнив задание 6 из § 54, вы составили два алгоритма, позволяющие находить в графе точки сочленения. Запрограммируйте эти алгоритмы и отладьте получившиеся программы. Для отладки используйте графы, изображенные на рисунке 6.13.

2 Используя подпрограмму, написанную вами при выполнении задания 8 из лабораторной работы № 24, сгенерируйте несколько связных графов и с помощью составленных вами программ найдите для них все точки сочленения.

3 Выполнив задание 7 из § 54, вы составили два алгоритма, позволяющие находить в графе мосты. Запрограммируйте эти алгоритмы и отладьте получившиеся программы. Для отладки снова используйте графы, изображенные на рисунке 6.13.

4 Используя подпрограмму, написанную вами при выполнении задания 8 из лабораторной работы № 24, сгенерируйте несколько связных графов и с помощью составленных вами программ найдите для них все мосты.

Лабораторная работа № 26 (к § 55 и 56)

Построение каркасов

1 Выполнив задание из § 55, вы на основе поиска в глубину составили два алгоритма, позволяющие находить каркас графа, представленного таблицей смежности и списком ребер. Запрограммируйте эти алгоритмы и отладьте получив-

шиеся программы. Для отладки воспользуйтесь графом, представленным на рисунке КП.14.

2 Из составленной вами программы и подпрограммы, созданной при выполнении задания 8 из лабораторной работы № 24, создайте подпрограмму, позволяющую строить случайным образом дерево с заданным числом вершин.

3 Выполните задание, аналогичное заданию 1, взяв за основу построения каркаса алгоритм поиска в ширину.

4 Приведем алгоритм, позволяющий строить дерево с заданным числом вершин. В нем через $SRD[1{:}(n-1);\ 1{:}2]$ обозначен массив, содержащий список ребер создаваемого дерева.

```
Алгоритм Случайное_дерево
цел: n, i, j, k, s, SRD[1:(n − 1); 1:2]; KS[1:n];
{ Запросить n;          (* запрашивается количество вершин *)
  Делать от i := 1 до n
  { KS[i] := i;
  }
  Делать от i := 1 до n − 1
  { j := INT((n − 1)*rand(1)) + 1;
    k := INT((n − j)*rand(1)) + j + 1;
    Делать пока (KS[j] = KS[k])
    { j := INT((n − 1)*rand(1)) + 1;
      k := INT((n − j)*rand(1)) + j + 1;
    }
    t := KS[k];
    SRD[i, 1] := j;
    SRD[i, 2] := k;
    Делать от s := 1 до n
    { Если ks[s] = t то { ks[s] := ks[j]; }
    }                                  (* конец цикла *)
  }
}
```

Запрограммируйте этот алгоритм, отладьте его для небольших значений n. (Например, $n = 6$; $n = 9$ и т. п.)

5 Оформите созданную вами программу как подпрограмму, для которой аргументом будет переменная n, а результатом — массив $SRD[1{:}n-1;\ 1{:}2]$.

6 В объяснительном тексте § 55 рассказано, как изображают деревья: для этого одну из вершин назначают корнем, а остальные вершины распределяются по уровням, определяемым удаленностью вершины от корня. Используя графические операторы, составьте алгоритм изображения дерева —

номер вершины, которая будет служить корнем, запрашивается у пользователя (не забудьте предварительно сообщить пользователю, сколько вершин имеет дерево).
Запрограммируйте составленный вами алгоритм и отладьте получившуюся программу. Используя подпрограмму построения случайного дерева, постройте дерево и получите 2—3 его изображения, по-разному выбирая корневую вершину.

7 Постройте 2—3 дерева (с разным количеством вершин) и для каждого из этих деревьев найдите его центр, воспользовавшись программой, составленной при выполнении задания 3 лабораторной работы № 24. Изобразите каждое дерево, приняв центральную вершину в качестве корня. В чем преимущество изображения дерева, у которого корнем служит центральная вершина, перед другими изображениями дерева?

8* Придумайте способ построения каркаса наименьшего радиуса для заданного графа. Составьте алгоритм построения такого каркаса. Запрограммируйте этот алгоритм и отладьте программу. Для отладки можно воспользоваться графами, изображенными на рисунках КП.14 и КП.15. С помощью подпрограммы, созданной при выполнении задания 8 из лабораторной работы № 24, сгенерируйте связный граф и найдите для него каркас наименьшего радиуса.

9 Запрограммируйте алгоритм Краскала, который вы составили, выполнив задание 5 из § 56. Используя подпрограмму, составленную при выполнении задания 6 из лабораторной работы № 21, сгенерируйте нагруженный граф и найдите какой-либо каркас минимального веса. (Предварительно убедитесь, что эти графы связны.)

10 Модифицируйте программу, составленную вами при выполнении предыдущего пункта лабораторной работы, так, чтобы стало возможным находить каркасы максимального веса. Найдите с помощью полученной программы какой-либо каркас максимального веса для тех же нагруженных графов, которые использовались вами в предыдущем пункте.

11 Запрограммируйте алгоритм Прима (см. задание 9 из § 56). Отладьте программу. Найдите каркас минимального веса с помощью этой программы и с помощью алгоритма Краскала для одного и того же нагруженного графа. Получились ли у вас одинаковые каркасы?

Лабораторная работа № 27 (к § 58)

Построение стратегии на основе списка проигрышных позиций

1 Выполняя задание 8 из § 58, вы составили алгоритм построения массива проигрышных позиций для игры, рассмотренной в объяснительном тексте этого параграфа. Напомним, что в объяснительном тексте массив a описан оператором **цел**:$a[1{:}n;\ 1{:}2]$. При этом вам пришлось принять решение о значении n в зависимости от размера поля. Если поле имеет размер $x \times y$ клеток, то в качестве n мы советуем вам взять минимальное из чисел x и y.
Запрограммируйте свой алгоритм и отладьте получившуюся программу.

2 Модифицируйте алгоритм игры, приведенный в объяснительном тексте § 58, так, чтобы можно было играть на бесконечном поле. Первым ходом фишка ставится на произвольное поле с координатами (k, m). Будьте снисходительны к своему противнику и предоставьте ему право сделать первый ход. Разумеется, надо предварительно сообщить, из какого диапазона следует выбрать числа k и m. Чтобы игра не закончилась на первом ходе, k и m надо выбирать отличными от 1.
Запрограммируйте составленный вами алгоритм. Для этого оформите программу формирования массива проигрышных позиций, которую вы создали в предыдущем пункте, в виде подпрограммы.
Отладьте и протестируйте полученную программу.

3 Алгоритм, приведенный в тексте § 58, не проверяет корректность хода противника. Тем не менее сделать ход неправильно (по ошибке, а вовсе не обязательно со злым умыслом) может каждый человек — он ведь не компьютер. Запрограммируйте блок проверки корректности хода противника (в частности, этот блок должен включать проверку корректности первого хода — попадание в предписанный вами диапазон для k и m). Если ход был сделан неверно, вежливо сообщите об этом противнику и попросите его сделать другой ход.
Отладив и протестировав новую программу, подумайте над тем, достаточно ли комфортно общение пользователя с вашей программой.
А теперь можете предложить поиграть в эту игру своим друзьям и знакомым.

Лабораторная работа № 28 (к § 59)

Построение стратегии на основе инварианта

Построим стратегию для игры «Баше» (см. задание 6 из § 59). Для общего случая, т. е. когда в кучке первоначально лежит n камней и за один ход разрешается брать из нее не более k камней, проигрышные позиции содержат: n камней, $n - (k + 1)$ камней, $n - 2(k + 1)$ камней, ..., r камней, где r — остаток при делении числа n на $k + 1$. Следовательно, позиция проигрышная, если остаток от деления числа камней в позиции на $k + 1$ равен числу r. Это и есть инвариант стратегии.

Можно сделать вывод: если $r = 0$, то исходная позиция проигрышная для игрока, делающего первый ход; если же $r \neq 0$, то, взяв из кучки r камней, первый игрок ставит второго игрока в проигрышную позицию. Далее, если игрок, находящийся в проигрышной позиции, взял s камней, то ответным ходом нужно взять $k + 1 - s$ камней.

1 Мы надеемся, что вами уже составлен алгоритм игры «Баше». Но прежде чем превращать свой алгоритм в программу, доработайте его, ответив на следующие вопросы:

1. Как в вашем алгоритме предусмотрено задание чисел n и k? Например:
   - оба числа запрашиваются у пользователя;
   - оба числа генерируются с помощью датчика случайных чисел;
   - одно число генерируется с помощью датчика случайных чисел, а другое запрашивается у пользователя;
   - одно число запрашивается у пользователя, а другое определяется компьютером в рамках алгоритма игры.

Возможны другие варианты определения начальных параметров игры.

2. Кто будет делать первый ход? Например:
   - определяется случайным образом;
   - выбор будет делать пользователь;
   - делающий первый ход всегда фиксирован.
3. Что нужно сделать раньше: выбор начальных параметров игры или выбор того, кто будет делать первый ход?

Возможно, что в вашем классе реализуются разные варианты ответов на эти вопросы. Так будет даже интереснее — вы сможете поиграть в разные варианы этой игры и оценить, какой из них лучше.

Продумайте интерфейс своей программы — как будет происходить общение с пользователем. Не забывайте, что оно должно быть не только вежливым, но и дружественным.

Предусмотрите проверку корректности хода своего противника.
Запрограммируйте свой алгоритм и отладьте получившуюся программу.

2 В задании 4 из § 59 вам было предложено составить алгоритм игры «Ним». Он сложнее алгоритма игры «Баше». Но ведь пользователю не так интересно, сколько труда и пота потребовалось от вас; он будет оценивать только созданный вами продукт. Поэтому, прежде чем программировать составленный вами алгоритм, доработайте его, ответив на вопросы, аналогичные тем, которые были сформулированы в предыдущем пункте для игры «Баше».
Запрограммируйте свой алгоритм и отладьте получившуюся программу. Поиграйте с компьютером сами и предложите другим. Какие отзывы о своем продукте вы получили?

3 В задании 11 из § 59 вам предложено составить алгоритм игры, описанной в § 57. Продумайте интерфейс. Доработайте свой алгоритм, запрограммируйте его и отладьте получившуюся программу. Поиграйте с компьютером сами и предложите другим. Какие отзывы о своем продукте вы получили?

Лабораторная работа № 29 (к § 60)

Построение стратегии на основе оценочной функции

1 Прежде всего поиграйте в крестики-нолики по той программе, которую составили мы. Для этого введите ее (разумеется, выбрав знакомый вам язык программирования).

Basic

```
SUB proverka (kol)
 FOR i = 1 TO 3
  IF (m(i,1) = 2) AND (m(i,2) = 2)
  AND (m(i,3) = 2) THEN
      PRINT ("Поздравляем!")
  END
  END IF
  IF (m(i,1) = 3) AND (m(i,2) = 3)
  AND (m(i,3) = 3) THEN
      PRINT ("Вы проиграли!")
      END
```

Pascal

```
uses crt;
var
  m, oc:array [1..3, 1..3] of integer;
  new1, new2, hod1, hod2:integer;
  cx, cx1, cx2, max, kol:integer;
  s: array [1..6] of integer;

procedure proverka;
var i:integer;
begin
 for i:= 1 to 3 do
```

Basic

```
  END IF
  IF (m(1,i) = 2) AND (m(2,i) = 2)
  AND (m(3,i) = 2) THEN
      PRINT ("Поздравляем!")
      END
  END IF
  IF (m(1,i) = 3) AND (m(2,i) = 3)
  AND (m(3,i) = 3) THEN
      PRINT ("Вы проиграли!")
      END
  END IF
 NEXT i
  IF (m(1,1) = 2) AND (m(2,2) = 2)
  AND (m(3,3) = 2) THEN
      PRINT ("Поздравляем!")
      END
  END IF
  IF (m(1,1) = 3) AND (m(2,2) = 3)
  AND (m(3,3) = 3) THEN
      PRINT ("Вы проиграли!")
      END
  END IF
  IF (m(1,3) = 2) AND (m(2,2) = 2)
 AND (m(3,1) = 2) THEN
      PRINT ("Поздравляем!")
      END
  END IF
  IF (m(1,3) = 3) AND (m(2,2) = 3)
 AND (m(3,1) = 3) THEN
      PRINT ("Вы проиграли!")
      END
  END IF
  END SUB


SUB WaR (kol)
 PRINT ("Вы            - X")
 PRINT ("Компьютер  - O")
 PRINT
 PRINT ("    1  2  3")
 FOR i2 = 1 TO 3
  PRINT i2; m(i2,1); m(i2,2); m(i2,3)
  PRINT
 NEXT i2
```

Pascal

```
   begin
   if (m[i,1]=2) and (m[i,2]=2) and
(m[i,3]=2) then
   begin
     writeln ('Поздравляем!');
     halt(0);
   end;
    if (m[i,1]=1)and(m[i,2]=1)and
(m[i,3]=1) then
    begin
     writeln ('Вы проиграли!');
     halt(0);
    end;
    if (m[1,i]=2) and (m[2,i]=2) and
(m[3,i]=2) then
    begin
     writeln ('Поздравляем!');
     halt(0);
    end;
    if (m[1,i]=1) and (m[2,i]=1) and
(m[3,i]=1) then
    begin
     writeln ('Вы проиграли!');
     halt(0);
    end;
    end;
    if (m[1,1]=2) and (m[2,2]=2) and
(m[3,3]=2) then
   begin
   writeln ('Поздравляем!');
    halt(0);
   end;
    if (m[1,3]=2) and (m[2,2]=2) and
(m[3,1]=2) then
    begin
     writeln ('Поздравляем!');
     halt(0);
     end;
    if (m[1,1]=1) and (m[2,2]=1) and
(m[3,3]=1) then
    begin
     writeln ('Вы проиграли!');
     halt(0);
     end;
```

Basic

```
 Kol = kol + 1
 CALL proverka(kol)
 IF kol < 10 THEN
  DO
   INPUT "Сделайте следующий
ход", new1, new2
  IF (new1 >= 1) AND (new1 <= 3)


AND (new2 >= 1) AND (new2 <= 3)
THEN
      WHILE m(new1, new2) <> 0
   IF (new1 >= 1) AND (new1 <= 3)
AND (new2 >= 1) AND (new2 <= 3)
THEN
 INPUT "Сделайте следующий ход",
new1, new2
      END IF
    WEND
      END IF
 LOOP WHILE (new1 < 1)
 OR (new1 > 3) OR (new2 < 1)
 OR (new2 > 3)
  m(new1, new2) = 2
  oc(new1, new2) = -1
  CALL proverka(kol)
  kol = kol + 1
  CLS
 ELSE
  PRINT "Ничья!"
  END
 END IF
END SUB


FUNCTION ocenka (a, b)
   Result = 0
    f1 = 0: f2 = 0: f3 = 0: f4 = 0
   FOR i3 = 1 TO 3
      IF (m(a, i3) = 2) THEN
           SELECT CASE f1
           CASE 0
               f1 = 1
           CASE 1
```

Pascal

```
 if (m[1,3]=1) and (m[2,2]=1) and
(m[3,1]=1) then
  begin
    writeln ('Вы проиграли!');
    halt(0);
    end;
end;


procedure WaR;
 var i,j:integer;
 begin
  writeln ('Вы       -   X');
  writeln ('Компьютер - O');
  writeln;
  writeln ('   1 2 3');
  for i:= 1 to 3 do
  begin
    write(i,' ');
  for j:= 1 to 3 do
  begin
    if  m[i,j]=0 then write('- ');
    if m[i,j]=1 then write('0 ');
    if m[i,j]=2 then write('X ');
   end;
    writeln;
 end;
inc(kol);
proverka;
if kol<10 then
begin
repeat
 write('Сделайте следующий ход:');
 readln(new1, new2);
until (new1<=3) and (new1>=1)
and(new2<=3) and (new2>=1)
and (m[new1,new2]=0);
    m[new1,new2]:=2;
    oc[new1,new2]:=-1;
    proverka;
    inc(kol);
    end else
    begin
    writeln('Ничья!'); readln;
    halt(0);
```

Basic

```
            f1 = 4
        CASE 2
            f1 = 3
        END SELECT
    END IF
    IF (m(a, i3) = 3) THEN
        SELECT CASE f1
        CASE 0
            f1 = 2
        CASE 1
            f1 = 3
        CASE 2
            f1 = 5
        END SELECT
    END IF
    IF (m(i3, b) = 2) THEN
        SELECT CASE f2
        CASE 0
            f2 = 1
        CASE 1
            f2 = 4
        CASE 2
            f2 = 3
        END SELECT
    END IF
    IF (m(i3, b) = 3) THEN
        SELECT CASE f2
        CASE 0
            f2 = 2
        CASE 1
            f2 = 3
        CASE 2
            f2 = 5
        END SELECT
    END IF
NEXT i3
IF a = b THEN
  FOR i3 = 1 TO 3
    IF (m(i3, i3) = 2) THEN
        SELECT CASE f3
        CASE 0
            f3 = 1
        CASE 1
            f3 = 4
```

Pascal

```
  end;
end;


function ocenka(a,b:integer):integer;
var i:integer;
  f1,f2,f3,f4: integer;
  result: integer;
begin
  result:=0;
  f1:=2; f2:=2; f3:=2; f4:=2;
  for i:=1 to 3 do
   begin
   if (m[a,i]=1) then
      case f1 of
         2: f1:=4;
         3: f1:=1;
         4: f1:=6;
      end;
    if (m[a,i]=2) then
      case f1 of
         2: f1:=3;
         3: f1:=5;
         4: f1:=1;
      end;
    if (m[i,b]=1) then
      case f2 of
         2: f2:=4;
         3: f2:=1;
         4: f2:=6;
      end;
    if (m[i,b]=2) then
      case f2 of
         2: f2:=3;
         3: f2:=5;
         4: f2:=1;
      end;
   end;
 if a=b then
for i:= 1 to 3 do
  begin
  if (m[i,i]=1) then
     case f3 of
        2: f3:=4;
```

Basic

```
        CASE 2
            f3 = 3
        END SELECT
    END IF
    IF (m(i3, i3) = 3) THEN
        SELECT CASE f3
        CASE 0
            f3 = 2
        CASE 1
            f3 = 3
        CASE 2
            f3 = 5
        END SELECT
    END IF
  NEXT i3
END IF
IF a = 4 - b THEN
  FOR i3 = 1 TO 3
    IF (m(i3, 4 - i3) = 2) THEN
        SELECT CASE f4
        CASE 0
            f4 = 1
        CASE 1
            f4 = 4
        CASE 2
            f4 = 3
        END SELECT
    END IF
    IF (m(i3, 4 - i3) = 3) THEN
        SELECT CASE f4
        CASE 0
            f4 = 2
        CASE 1
            f4 = 3
        CASE 2
            f4 = 5
        END SELECT
    END IF
  NEXT i3
END IF
Result=result+s(f1)+s(f2)
IF a = b THEN
   result=result+s(f3)
END IF
```

Pascal

```
        3: f3:=1;
        4: f3:=6;
      end;
    if (m[i,i]=2) then
      case f3 of
        2: f3:=3;
        3: f3:=5;
        4: f3:=1;
      end;
    end;
if a=4-b then
    for i:= 1 to 3 do
      begin
       if (m[i,4-i]=1) then
         case f4 of
            2: f4:=4;
            3: f4:=1;
            4: f4:=6;
       end;
       if (m[i,4-i]=2) then
        case f4 of
            2: f4:=3;
            3: f4:=5;
            4: f4:=1;
     end;
     end;
 result:=result+s[f1];
 result:=result+s[f2];
 if a=b then
        begin
        result:=result+s[f3];
     end;
 if (a=4-b) then
        begin
        result:=result+s[f4];
      end;
    ocenka:=result;
end;
```

Basic

```
 IF a = 4 - b THEN
    result=result+s(f4)
 END IF
 Ocenka=result
END FUNCTION


DIM SHARED m(3, 3)
DIM SHARED oc(3, 3)
DIM SHARED s(5)
s(0)=5
s(1)=10
s(2)=15
s(4)=45
s(5)=100
CLS
kol = 0
CALL WaR(kol)
FOR cx = 1 TO 5
 Max = 0
 FOR cx1 = 1 TO 3
   FOR cx2 = 1 TO 3
     IF m(cx1, cx2) = 0 THEN
      oc(cx1, cx2) = ocenka(cx1, cx2)
      IF oc(cx1, cx2) > max THEN
      max = oc(cx1, cx2)
      hod1 = cx1
      hod2 = cx2
     END IF
    END IF
  NEXT cx2, cx1
  M(hod1, hod2) = 3
  oc(hod1, hod2) = -1
  CALL WaR(kol)
NEXT cx
```

Pascal

```
begin
s[1]:=0; s[2]:=5; s[3]:=10; s[4]:=15;
s[5]:=45; s[6]:=100;
kolhod:=0;
  WaR;
  for cx:=1 to 5 do
  begin
   max:=0;
   for cx1:= 1 to 3 do
    for cx2:= 1 to 3 do
      if m[cx1,cx2]=0 then
       begin
       oc[cx1,cx2]:=ocenka(cx1,cx2);
       if oc[cx1,cx2]>max then
          begin
          max:=oc[cx1,cx2];
          hod1:=cx1;
          hod2:=cx2;
          end;
        end;
  m[hod1,hod2]:=1;
  oc[hod1,hod2]:=-1;
  WaR;
 end;
end.
```

Для того чтобы не привлекать символьные переменные, пустая клетка обозначена числом 0, клетка, в которой стоит нолик, — числом 1, а клетка с крестиком — числом 2.

В ответ на приглашение компьютера сделать ход вы указываете номер строки и номер столбца, клетки, в которую вы намерены очередным ходом поставить крестик.

2 Поэкспериментируйте с другими значениями оценочной функции. В программе значения оценочной функции хранятся в массиве *s*[5]. Вы можете поменять значения оценочной функции, изменив соответствующие операторы присваивания.

3 Реализуйте игру в крестики-нолики на поле размером 4×4, в которой по-прежнему выигрывает тот, кто первым составил сплошной горизонтальный, вертикальный или диагональный ряд длины 3 из своих символов.

4 Реализуйте игру в крестики-нолики на поле размером 4×4, в которой выигрывает тот, кто первым составил сплошной горизонтальный, вертикальный или диагональный ряд длины 4 из своих символов.

ЕГЭ Готовимся к Единому государственному экзамену по информатике

Вы заканчиваете школу. Мы предлагаем вам выполнить ряд тестовых заданий, которые по уровню сложности и формулировкам подобны заданиям, реально предлагавшимся на ЕГЭ в предшествующие годы. Поэтому в представленных ниже заданиях вы найдете материал, изучавшийся не только в 11 классе, но и в предшествующие годы.

Часть 1. При выполнении предложенных ниже заданий запишите шифр задания и номер правильного ответа. По окончании работы сверьте получившуюся у вас запись с ключами на с. 327.

А1. Цифра 1 в двоичной записи числа 197 встречается:

1) 3 раза; 2) 4 раза; 3) 5 раз;
4) количество раз, отличное от указанных в пунктах 1—3.

А2. Значение суммы $10_2 + 30_8 + 90_{16}$ в двоичной системе счисления равно:

1) 11010; 2) 110010;
3) 10101010; 4) 1100110.

А3. Число ABCD, записанное в шестнадцатеричной системе счисления, в десятичной системе счисления записывается так:

1) 43981; 2) 46813; 3) 34156; 4) 35843.

А4. Среди высказываний $27_9 > 17_{18}$, $16_{13} = 18_{11}$, $10101_2 < 16_{16}$ количество истинных высказываний равно:

1) 0; 2) 1; 3) 2; 4) 3.

А5. В каком порядке расположатся текстовые фрагменты «excel», «бит», «10 Б», «11 А», если их упорядочить по возрастанию в соответствии с ASCII-кодировкой?

1) «10 Б», «11 А», «бит», «excel»;
2) «10 Б», «11 А», «excel», «бит»;
3) «бит», «excel», «10 Б», «11 А»;
4) «11 А», «10 Б», «excel», «бит».

А6. При цветной печати на принтере используется следующая модель цветового кодирования:

1) RGB; 2) HSB; 3) CMYK; 4) CMY.

А7. Скорость передачи данных через ADSL-соединение равна 256000 бит/с. Передача файла через это соединение заняла 4 мин. Размер файла в килобайтах равен:

1) 60 000; 2) 7500; 3) 125;
4) числу, отличному от указанных в пунктах 1—3.

А8. Известны имя почтового сервера (mserver), находящегося в России, и имя почтового ящика (Nike). Укажите адрес электронной почты, соответствующий этим данным:

1) mserver@Nike.ru; 2) mserver.Nike@ru;
3) Nike.mserver@ru; 4) Nike@mserver.ru.

А9. Разделителем иерархических частей в доменном имени хост-компьютера служит:

1) запятая; 2) точка; 3) двоеточие; 4) символ «/».

А10. Каким условием нужно воспользоваться для поиска в Интернете информации о цветах, растущих на острове Мадагаскар или острове Суматра?

Примечание: знаком «|» обозначена операция «ИЛИ», знаком «&» — операция «И».

1) цветы & (Мадагаскар | Суматра);
2) цветы & Мадагаскар & Суматра;
3) цветы | Мадагаскар | Суматра;
4) цветы & (остров | Мадагаскар | Суматра).

А11. Выберите самый маленький объем информации из приведенных ниже:

1) 1000 Кбайт; 2) 1 000 000 байт;
3) 10 000 000 бит; 4) 1 Мбайт.

А12. Таблица стоимости проезда составлена следующим образом: буквами обозначены станции, а на пересечении строки и столбца указана стоимость проезда между соответствующими соседними станциями. Если на пересечении строки и столбца клетка пустая, то такие станции не являются соседними. Стоимость проезда по маршруту от одной станции до другой равна сумме стоимостей проезда между соседними станциями выбранного маршрута. Укажите таблицу, для которой стоимость проезда из А в Б не больше 6.

1)

	А	Б	В	Г	Д
А			3	1	
Б			4		2
В	3	4		2	2
Г	1		2		
Д		2	2		

2)

	А	Б	В	Г	Д
А			4	1	1
Б			3		
В	4	3			2
Г	1				
Д	1		2		

3)

	А	Б	В	Г	Д
А			4	1	
Б			3		1
В	4	3		2	2
Г	1		2		
Д		1	2		

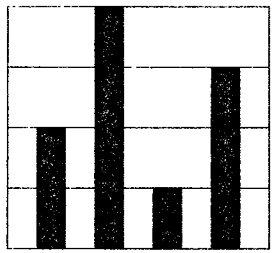
4)

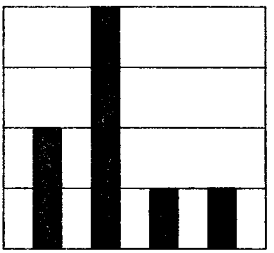
	А	Б	В	Г	Д
А				1	
Б			3		1
В		3		3	2
Г	1		3		
Д		1	2		

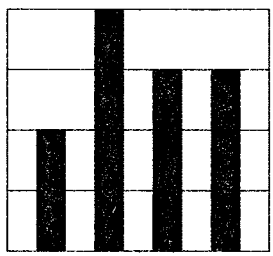
А13. Дан фрагмент электронной таблицы:

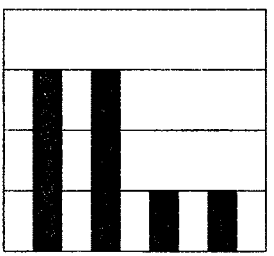
	А	В
1	=B1+1	1
2	=A1+2	2
3	=B2–1	
4	=A3	

После выполнения вычислений была построена диаграмма по значениям диапазона ячеек А1:А4. Укажите получившуюся диаграмму.

1) 

2) 

3) 

4) 

A14. В ячейках A3:A5 изображенного ниже фрагмента таблицы Excel перечислены названия продуктов, продаваемых в хлебном отделе магазина. В соответствующих ячейках столбца B указана цена за 1 штуку товара. Требуется заполнить столбцы C и D так, чтобы в их ячейках подсчитывалась стоимость двух или трех единиц товара соответственно.

	A	B	C	D
1		Стоимость		
2		1	2	3
3	пирожок	5,00		
4	булочка	6,50		
5	ватрушка	6,00		

Какая формула должна быть записана в ячейке C3, чтобы после ее копирования в остальные ячейки столбцов C и D получилась требуемая таблица?

1) =B3*C2; 2) =$B3*C$2;

3) =B$3*$C2; 4) =B3*C2.

А15. Из букв А, В, Д, Е, И составляются четырехсимвольные цепочки. При этом должны соблюдаться следующие правила:
а) на первом месте стоит гласная буква;
б) после гласной буквы не может снова идти гласная буква;
в) последняя буква не совпадает с первой.

Какая из цепочек построена по этим правилам?
1) АИДА; 2) ДЕВА; 3) ЕДВА; 4) АВВА.

А16. Дан алгоритм:

```
Алгоритм
цел: A, B, C;
{   Запросить A;
    Запросить B;
    Запросить C;
   Делать пока (C < 0)
   {   Сообщить C;
       C := A*C + B;
    }
}
```

При некоторых начальных значениях A, B и C к некоторому моменту исполнения алгоритма были напечатаны числа −3, −5, −1. Сколько еще чисел будет напечатано?
1) 0; 2) 1; 3) 3;
4) бесконечно много, так как алгоритм зациклится.

А17. Какому из указанных высказываний равносильно высказывание $(X \vee \overline{Y}) \to (\overline{X} \to Y)$?
1) $X \,\&\, Y$; 2) $X \vee Y$; 3) $X \leftrightarrow Y$;
4) ни одному из перечисленных в пунктах 1—3.

А18. Задана таблица:

X	Y	$F(X, Y)$
И	И	И
И	Л	Л
Л	И	Л
Л	Л	И

Такую же таблицу истинности имеет формула:
1) $((X \to Y) \to X) \vee (\overline{X} \,\&\, \overline{Y})$; 2) $\overline{X \to Y} \vee ((X \vee Y) \,\&\, \overline{X})$;
3) $(Y \to X) \,\&\, (\overline{X} \vee Y)$; 4) ни одна из указанных в пунктах 1—3.

Часть 2. При выполнении предложенных ниже заданий запишите шифр задания и ответ в виде последовательности символов. По окончании работы сверьте получившуюся у вас запись с ключами на с. 327.

В1. В системе счисления с некоторым основанием десятичное число 123 записывается как 234. Укажите основание системы.

В2. Укажите через запятую (без пробелов) в порядке возрастания все основания систем счисления, в которых десятичное число 23 оканчивается на 5.

В3. Сколько существует натуральных чисел, меньших 1000, содержащих в своей записи ровно две единицы после перевода их в двоичную систему счисления?

В4. Доступ к файлу ftp.net, находящемуся в папке txt.org, расположенной на сервере ru.com, осуществляется по протоколу http. В таблице буквами от А до И закодированы фрагменты адреса. Запишите последовательность этих букв, кодирующую адрес указанного файла в сети Интернет.

А	Б	В	Г	Д	Е	Ж	З	И
.net	txt	://	ftp	ru	.org	http	/	.com

В5. Сколько имеется целых значений переменной X, для которых истинно высказывание $(X^2 - 4X - 5 > 0) \Rightarrow (X \geqslant -3)\ \&\ (X \leqslant 2)$?

В6. Для хранения растрового цветного изображения размером 128×128 пикселей отвели 0,5 Мбайт памяти. Каково максимально возможное число цветов в палитре этого изображения?

В7. Дана программа:

Бейсик	Паскаль

PROGRAM Unknown;
DEFINT A, B, C
C = A*B*B
IF (A + B>C) THEN C = C + A
ELSE C = C – B
PRINT C
	Program Unknown;
	var A, B, C: integer;
	begin
	readln (A, B);
	C := A*B*B;
	if (A + B>C) then C := C + A
	else C := C – B;
	writeln (C);
	end.

Эта программа была выполнена для двух наборов значений А и В: А = 2; В = –1 и А = –2; В = 1. Чему равна сумма двух напечатанных программой чисел?

B8. Натуральное число преобразуется по следующим правилам. В нем выбирается наибольшая цифра (если таких несколько, то выбирается самая левая) и уменьшается на 1, если у нее есть цифра, стоящая правее ее и меньшая по значению. С полученным числом проделывается то же самое. Преобразование числа прекращается, если его невозможно выполнить, соблюдая указанные условия. Какое число получится из числа 192 837 465?

B9. У исполнителя есть две команды, которым присвоены номера 1 и 2:

1. Прибавь 5. 2. Умножь на 2.

Выполняя первую из них, исполнитель прибавляет к числу 5, а выполняя вторую, удваивает его. Программа для этого исполнителя — это последовательность цифр 1 и 2. Например, программа 21121 преобразует число 1 в число 29. Напишите программу, которая содержит не более 6 команд и позволяет получить то же число 29 из числа −1.

B10. Дан фрагмент таблицы в режиме отображения формул:

	A	B
2	3	=A$1*B1 + A2
3	1	

В ячейках A1 и B1 записаны некоторые числа. Содержимое ячейки B2 было скопировано в ячейку B3, после чего в режиме отображения результатов тот же фрагмент выглядит так:

	A	B
2	3	1
3	1	0

Какое число занесено в ячейку B1?

B11. В нарушении правил торговли подозреваются четыре работника магазина: Антонов, Борисов, Васильев и Денисов. Торговая инспекция выяснила:

1) если Антонов нарушил правила, то вместе с Борисовым;

2) если Борисов нарушил правила, то Васильев нарушил или Антонов не нарушал их;

3) если Денисов не нарушал правила, то Антонов нарушил, а Васильев не нарушал.

Сколько подозреваемых наверняка нарушили правила торговли?

B12. Известно, что высказывание

$$(A \,\&\, B) \vee (\overline{B} \,\&\, \overline{A} \,\&\, C) \vee (\overline{C} \,\&\, \overline{A})$$

ложно. Какое наибольшее число истинных высказываний может быть среди высказываний A, B и C?

B13. Для последовательности высказываний A_n при $n > 3$ выполнено следующее рекуррентное соотношение:

$$A_n = A_{n-1} \,\&\, (A_{n-2} \vee A_{n-3}).$$

Сколько истинных высказываний будет среди первых 100 членов последовательности, если A_1 и A_3 истинны, а A_2 ложно?

КЛЮЧИ К ТЕСТОВЫМ ЗАДАНИЯМ

A1	A2	A3	A4	A5	A6	A7	A8	A9
2	3	1	3	2	3	2	4	2

A10	A11	A12	A13	A14	A15	A16	A17	A18
1	2	3	2	2	3	2	2	3

B1	B2	B3	B4	B5	B6	B7
7	6,9,18	45	ЖВДИЗБЕЗГА	9	256	−1

B8	B9	B10	B11	B12	B13
142434455	212221	2	1	2	99

Эпилог

Закончилось ваше многолетнее изучение информатики в школе. Оглядываясь на весь пройденный путь (а не только на материал 11 класса), легко различить наиболее значительные вершины, которые, как мы надеемся, вы смогли покорить:

- работа с прикладным программным обеспечением;
- знакомство с миром компьютерных сетей и Интернетом;
- построение компьютерных моделей для решения жизненных задач;
- осознание социальных проблем информатизации общества;
- изучение алгоритмов и знакомство с языками программирования;
- изучение принципов работы самого компьютера.

В современном информационном обществе все вами изученное составляет основу информационной культуры, которой должен владеть каждый.

Оканчивая школу, вы выбираете сферу своих будущих профессиональных интересов. Кто-то ощущает себя гуманитарием, а кого-то влекут науки о природе. Но любому из вас потребуется информатика, поскольку она изучает важнейшие стороны человеческого существования — процессы получения, накопления, передачи и обработки информации. А без информации невозможна никакая продуктивная деятельность!

Возможно, кто-то из вас выберет информатику своей будущей профессией. Тогда вы, вероятнее всего, изучали информатику как профильный курс. Хочется верить, что, знакомясь с этой замечательной наукой по нашему учебнику, вы увидели ее с разных сторон — ведь она столь многогранна! Вы освоили самые разнообразные компьютерные технологии и прикладное программное обеспечение, научились построению информационных моделей процессов и явлений, прикоснулись к социальным аспектам процесса информатизации общества, окунулись в мир компьютерных сетей и Интернета, освоили методы доказательного программирования и глубже изучили принципы работы самого компьютера. Все это хорошая основа для будущей профессиональной деятельности в информационной сфере. Информатика и как наука, и как область человеческой деятельности чрезвычайно востребована в современном обществе. Выбирайте любое ее направление, и вы увидите, как много интересного и полезного ждет вас в этой сфере.

Конечно, для каждой профессиональной сферы характерен свой взгляд на информатику. Но разноликая и в то же время единая в своих основах информатика останется с вами навсегда.

Литература для дополнительного чтения

Андреева Е. Системы счисления и компьютерная арифметика / Е. Андреева, И. Фалина. — М.: ЛБЗ, 2000.

Андреева Е. В. Математические основы информатики / Е. В. Андреева, Л. Л. Босова, И. Н. Фалина. — М.: ЛБЗ, 2005.

Аршинов М. И. Коды и математика / М. И. Аршинов, Л. Е. Садовский. — М.: Наука, 1983.

Введение в криптографию / под общ. ред. В. В. Ященко. — М.: МЦМНО, 1998.

Верещагин Н. К. Вычислимые функции / Н. К. Верещагин, А. Шень. — М.: МЦМНО, 2002.

Гейн А. Г. Задачник-практикум по информатике и информационным технологиям: кн. для учащихся 7—11 кл. общеобразоват. учреждений / А. Г. Гейн, Н. А. Юнерман. — М.: Просвещение, 2003.

Ивашина М. В. Человек и информация: Учеб. пособие для сред. шк. / М. В. Ивашина, А. Г. Гейн, О. В. Брюхова и др.; под ред. А. Г. Гейна, Н. С. Сулимовой. — Екатеринбург: Центр «Учебная книга», 2007.

Гурский Ю. Photoshop 7. Трюки и эффекты / Ю. Гурский, Г. Красильникова. — СПб.: Питер, 2004.

Джордин Д. Использование Adobe Photoshop / Д. Джордин, С. Мониц. — М.: Изд. дом «Вильямс», 1999.

Информатика. Задачник-практикум. В 2 т. / под ред. И. Г. Семакина, Е. К. Хеннера. — М.: ЛБЗ, 1999.

Информатика. Энциклопедический словарь для начинающих / под ред. Д. А. Поспелова. — М.: Педагогика-Пресс, 1994.

Информатика как наука об информации / под ред. Р. С. Гиляревского. — М.: Фаир-Пресс, 2006.

Кенин А. Окно в мир Интернета: практическое руководство / А. Кенин. — Екатеринбург: У-фактория, 2003.

Кольман Э. Занимательная логика / Э. Кольман, О. Зих. — М.: Наука, 1966.

Копылов В. А. Информационное право / В. А. Копылов. — М.: Юристъ, 1997.

Кристофидес Н. Теория графов. Алгоритмический подход / Н. Кристофидес. — М.: Мир, 1978.

Липский В. Комбинаторика для программистов / В. Липский. — М.: Мир, 1988.

Матвеева Т. А. Информационная культура: Учеб. пособие для сред. шк. / Т. А. Матвеева, А. Г. Гейн, В. В. Мачульский и др. — Смоленск: Ассоциация XXI век, 2007.

Окулов С. М. Основы программирования / С. М. Окулов. — М.: ЛБЗ, 2006.

Окулов С. М. Программирование в алгоритмах / С. М. Окулов. — М.: ЛБЗ, 2006.

Шафрин Ю. А. Информационные технологии / Ю. А. Шафрин. — М.: ЛБЗ, 1999.

Шень А. Программирование: теоремы и задачи / А. Шень. — М.: МЦМНО, 1995.

Предметный указатель

Оглавление

УДК 373.167.1:004
ББК 32.81я72
Г29

На учебник получены положительные заключения
Российской академии наук (№ 10106-5215/496 от 22.10.08)
и Российской академии образования (№ 01-5/7д-231 от 02.10.08).

Гейн А. Г.
Г29 Информатика и ИКТ. 11 класс : учеб. для общеобразоват. учреждений : базовый и профил. уровни / А. Г. Гейн, А. И. Сенокосов. — 2-е изд. — М. : Просвещение, 2009. — 336 с. : ил. — ISBN 978-5-09-026867-7.

УДК 373.167.1:004
ББК 32.81я72

Учебное издание

ГЕЙН Александр Георгиевич
СЕНОКОСОВ Александр Иванович

ИНФОРМАТИКА И ИКТ

11 класс

Учебник для общеобразовательных учреждений

Базовый и профильный уровни

Зав. редакцией *Т. А. Бурмистрова.* Редактор *О. В. Платонова.*
Художник *О. П. Богомолова.* Художественный редактор *О. П. Богомолова.*
Компьютерная графика *Г. М. Дмитриева, О. Ю. Тупикиной.*
Техническое редактирование и компьютерная верстка *Н. А. Киселевой.*
Корректоры *А. В. Рудакова, Г. Е. Казанцева, Л. С. Вайтман*

Налоговая льгота — Общероссийский классификатор продукции ОК 005-93—953000.
Изд. лиц. Серия ИД № 05824 от 12.09.01. Подписано в печать 05.12.11. Формат 70×90 1/16.
Бумага офсетная. Гарнитура SchoolBookCSanPin. Печать офсетная.
Уч.-изд. л. 25,16 + 0,5 форз. Тираж 7000 экз. Заказ № 4992.

Открытое акционерное общество «Издательство «Просвещение». 127521, Москва, 3-й проезд Марьиной рощи, 41.

Отпечатано в полном соответствии с качеством предоставленных издательством материалов в ОАО «Тверской ордена Трудового Красного Знамени полиграфкомбинат детской литературы им. 50-летия СССР». 170040, г. Тверь, проспект 50 лет Октября, 46.

ISBN 978-5-09-026867-7

БУЛЕВЫ ФУНКЦИИ ОТ ДВУХ ПЕРЕМЕННЫХ x И y

Функция	xy	00	01	10	11
тождественная единица	λ	1	1	1	1
тождественная	x	0	0	1	1
тождественная	y	0	1	0	1
отрицание	$\overline{x}$	1	1	0	0
отрицание	$\overline{y}$	1	0	1	0
дизъюнкция	$x \vee y$	0	1	1	1
конъюнкция	$x \,\&\, y$	0	0	0	1
импликация	$x \longrightarrow y$	1	1	0	1
импликация	$y \longrightarrow x$	1	0	1	1
коимпликация	$x \not\longrightarrow y$	0	0	1	0
коимпликация	$y \not\longrightarrow x$	0	1	0	0
операция Шеффера	$x \mid y$	1	1	1	0
сложение по модулю 2	$x \oplus y$	0	1	1	0
эквиваленция	$x \sim y$	1	0	0	1
операция Пирса	$x \downarrow y$	1	0	0	0
тождественный нуль	0	0	0	0	0

ОСНОВНЫЕ ЗАКОНЫ БУЛЕВЫХ ОПЕРАЦИЙ

КОММУТАТИВНОСТЬ

$$x \,\&\, y = y \,\&\, x \qquad x \vee y = y \vee x \qquad x \oplus y = y \oplus x$$

АССОЦИАТИВНОСТЬ

$$(x \,\&\, y) \,\&\, z = x \,\&\, (y \,\&\, z) \qquad (x \vee y) \vee z = x \vee (y \vee z)$$

$$(x \oplus y) \oplus z = x \oplus (y \oplus z)$$

ДИСТРИБУТИВНОСТЬ

$$(x \vee y) \,\&\, z = (x \,\&\, z) \vee (y \,\&\, z)$$

$$(x \,\&\, y) \vee z = (x \vee z) \,\&\, (y \vee z)$$

$$(x \oplus y) \,\&\, z = (x \,\&\, z) \oplus (y \,\&\, z)$$

$$(x \oplus y) \vee z = (x \vee z) \oplus (y \vee z)$$

ЗАКОНЫ ДЕ МОРГАНА

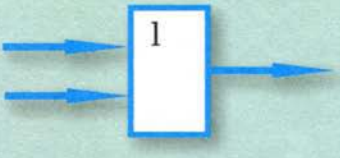
$$\overline{x \,\&\, y} = \bar{x} \vee \bar{y} \qquad \overline{x \vee y} = \bar{x} \,\&\, \bar{y}$$

ИДЕМПОТЕНТНОСТЬ

$$x \,\&\, x = x \qquad x \vee x = x$$

КОНЪЮНКЦИЯ

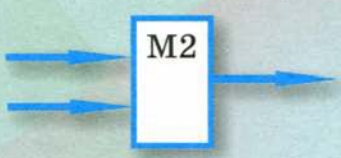
ДИЗЪЮНКЦИЯ



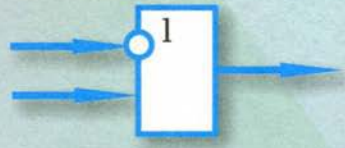
ОТРИЦАНИЕ

ЭКВИВАЛЕНЦИЯ

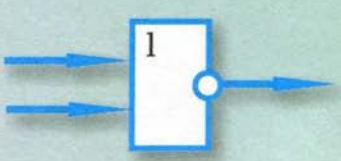
СЛОЖЕНИЕ ПО МОДУЛЮ



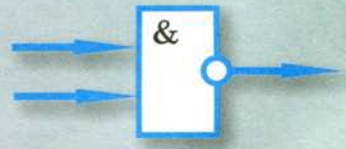
ИМПЛИКАЦИЯ



ОПЕРАЦИЯ ПИРСА



ОПЕРАЦИЯ ШЕФФЕРА



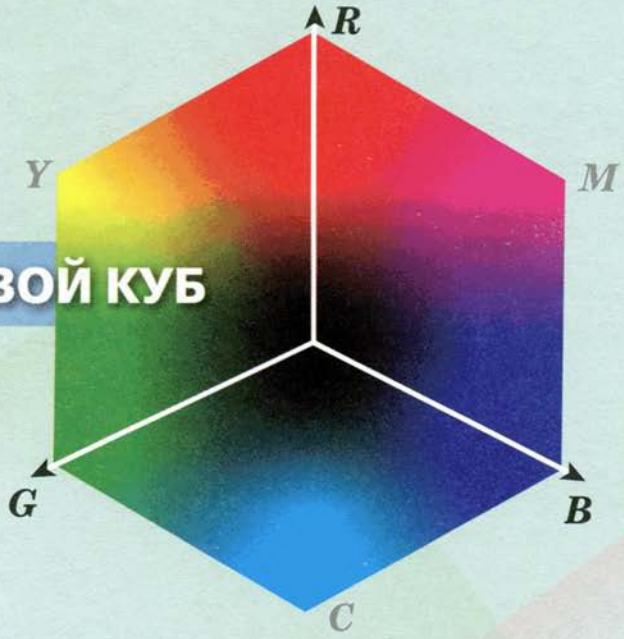
ОСНОВНАЯ ЧАСТЬ ASCII

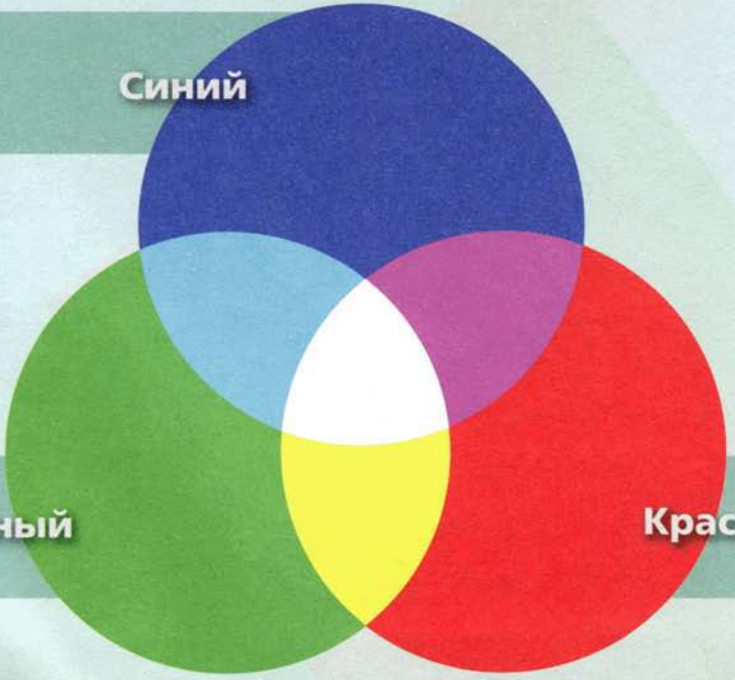
Двоичный код	Десятичный код	Символ
00100000	32	Пробел
00100001	33	!
00100010	34	"
00100011	35	#
00100100	36	$
00100101	37	%
00100110	38	&
00100111	39	'
00101000	40	(
00101001	41	)
00101010	42	*
00101011	43	+
00101100	44	,
00101101	45	-
00101110	46	.
00101111	47	/
00110000	48	0
00110001	49	1
00110010	50	2
00110011	51	3
00110100	52	4
00110101	53	5
00110110	54	6
00110111	55	7
00111000	56	8
00111001	57	9
00111010	58	:
00111011	59	;
00111100	60	<
00111101	61	=
00111110	62	>
00111111	63	?
01000000	64	@
01000001	65	A
01000010	66	B
01000011	67	C
01000100	68	D
01000101	69	E
01000110	70	F
01000111	71	G
01001000	72	H
01001001	73	I
01001010	74	J
01001011	75	K
01001100	76	L
01001101	77	M
01001110	78	N
01001111	79	O

Двоичный код	Десятичный код	Символ
01010000	80	P
01010001	81	Q
01010010	82	R
01010011	83	S
01010100	84	T
01010101	85	U
01010110	86	V
01010111	87	W
01011000	88	X
01011001	89	Y
01011010	90	Z
01011011	91	[
01011100	92	\
01011101	93	]
01011110	94	^
01011111	95	_
01100000	96	`
01100001	97	a
01100010	98	b
01100011	99	c
01100100	100	d
01100101	101	e
01100110	102	f
01100111	103	g
01101000	104	h
01101001	105	i
01101010	106	j
01101011	107	k
01101100	108	l
01101101	109	m
01101110	110	n
01101111	111	o
01110000	112	p
01110001	113	q
01110010	114	r
01110011	115	s
01110100	116	t
01110101	117	u
01110110	118	v
01110111	119	w
01111000	120	x
01111001	121	y
01111010	122	z
01111011	123	{
01111100	124	|
01111101	125	}
01111110	126	~
01111111	127	□

МОДЕЛИ RGB-КОДИРОВАНИЯ

ЦВЕТОВОЙ КУБ






Зеленый
120°
Желтый
60°
МОДЕЛЬ HSB:
круг Манселла
Голубой
180°
Красный
0°
Синий
240°
Пурпурный
300°
Пурпурный
МОДЕЛЬ CMYK
Голубой
Желтый

ИНФОРМАТИКА и ИКТ

БАЗОВЫЙ • ПРОФИЛЬНЫЙ УРОВНИ

Учебно-методический комплект по информатике включает:

- А. Г. Гейн, А. И. Сенокосов
  Учебник для 11 класса
- А. Г. Гейн, Н. А. Юнерман
  Тематические тесты. 11 класс
- А. Г. Гейн
  Задачник-практикум. 10 –11 классы
- А. Г. Гейн, Н. А. Юнерман
  Книга для учителя Методические рекомендации к учебнику 11 класса
- А. Г. Гейн
  Информатика и ИКТ. Программы общеобразовательных учреждений. 10 – 11 классы

Программы для проведения практических занятий и методические рекомендации размещены на сайтах:

http:// kadm.math.usu.ru (на страничке А.Г. Гейна)
http:// prosv. ru

ПРОСВЕЩЕНИЕ
ИЗДАТЕЛЬСТВО

ISBN 978-5-09-026867-7
9 785090 268677