СООТВЕТСТВУЕТ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЕ ПРИМЕРНОЙ ОСНОВНОЙ

10

Н. Д. Угринович

ИНФОРМАТИКА

БАЗОВЫЙ УРОВЕНЬ

НОВАЯ ШКОЛА БИНОМ

ИЗДАТЕЛЬСТВО БИНОМ

Н. Д. Угринович

ИНФОРМАТИКА

10 класс

Базовый уровень

Москва
БИНОМ. Лаборатория знаний
2017

УДК 004.9
ББК 32.97
У27

Угринович Н. Д.
У27 Информатика. 10 класс. Базовый уровень / Н. Д. Угринович. — М. : БИНОМ. Лаборатория знаний, 2017. — 288 с. : ил.

ISBN 978-5-9963-3144-4

Учебное издание ориентировано на преподавание информатики на базовом уровне в 10 классах общеобразовательных организаций. Рассматриваются такие темы, как информация и информационные процессы, информационные и коммуникационные технологии, основы алгоритмизации и программирования. Большое внимание уделяется формированию умений и навыков в процессе выполнения практических компьютерных работ. Учебное издание мультисистемное, так как работы могут выполняться в операционных системах Windows или Linux.

Учебное издание входит в учебно-методический комплект по информатике для 10–11 классов Н. Д. Угриновича наряду с учебным изданием для 11 класса, примерной рабочей программой и методическим пособием для учителя. Электронное приложение размещено в авторской мастерской Н. Д. Угриновича на сайте методической службы издательства (http://metodist.Lbz.ru).

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

УДК 004.9
ББК 32.97

Учебное издание

Угринович Николай Дмитриевич

ИНФОРМАТИКА
10 класс
Базовый уровень

Ведущий редактор *О. А. Полежаева*
Ведущие методисты *И. Ю. Хлобыстова, Е. Б. Животова*
Художник *Н. А. Новак*
Технический редактор *Е. В. Денюкова*
Корректор *Е. Н. Клитина*
Компьютерная верстка: *Л. В. Катуркина*

Подписано в печать 28.02.17. Формат 70x100/16. Усл. печ. л. 23,4.
Тираж 3000 экз. Заказ № м4352.

ООО «БИНОМ. Лаборатория знаний»
127473, Москва, ул. Краснопролетарская, д. 16, стр. 1
тел. (495) 181-5344, e-mail: binom@Lbz.ru
http://www.Lbz.ru, http://metodist.Lbz.ru

Отпечатано в филиале «Смоленский полиграфический комбинат»
ОАО «Издательство «Высшая школа». 214020, г. Смоленск, ул. Смольянинова, 1
Тел.: +7 (4812) 31-11-96. Факс: +7 (4812) 31-31-70
E-mail: spk@smolpk.ru http://www.smolpk.ru

ISBN 978-5-9963-3144-4

РЕКОМЕНДАЦИИ ПО ИСПОЛЬЗОВАНИЮ УЧЕБНИКА

Учебник обеспечивает изучение курса «Информатика» в 10 классе на базовом уровне.

Учебник входит в состав учебно-методического комплекта по информатике для старшей школы, включающего:

- учебники по курсу старшей школы на базовом уровне: **«Информатика. 10 класс»** и «Информатика. 11 класс»;
- методиче ское пособие для учителей.

Учебник содержит практикум.

Компьютерный практикум может проводиться в операционных системах Windows  и Linux  .

Файлы для выполнения практических работ (электронное приложение) размещены в авторской мастерской Н. Д. Угриновича на сайте методической службы издательства **(http://metodist.lbz.ru/)**.

Начало каждой работы компьютерного практикума обозначается значками операционной системы и приложений, для которых приведена подробная пошаговая инструкция выполнения работы.

В конце каждой главы приведён список рекомендуемых электронных образовательных ресурсов с сайта ФЦИОР (Федерального центра информационно-образовательных ресурсов): **http://fcior.edu.ru**.

В тексте учебника приняты следующие шрифтовые выделения:

- шрифтом Arial выделены имена программ, файлов, папок и дисков;
- шрифтом `Courier New` выделены программы на языках программирования;
- **полужирным шрифтом** выделены важные термины и понятия;
- *курсивом* выделены названия диалоговых окон, вкладок и управляющих элементов графического интерфейса операционных систем и приложений.

Звёздочкой (*) отмечены задания повышенной сложности.

Навигационные значки

Обратите внимание на символы навигационной полосы, имеющейся в учебниках. Они означают следующее:

 — важное утверждение или определение;

 — вопросы и задания;

 — материал для подготовки к итоговой аттестации;

 — дополнительный материал;

 — электронное приложение (файлы для выполнения практических работ) на сайте **http://metodist.lbz.ru** в авторской мастерской Н. Д. Угриновича;

 — интернет-ресурсы;

 — проектное или исследовательское задание;

 — практическая работа на компьютере;

 — межпредметные связи;

 — групповая работа.

Глава 1
ИНФОРМАЦИЯ И ИНФОРМАЦИОННЫЕ ПРОЦЕССЫ

При изучении данной главы требуется установить следующее программное обеспечение:

для операционных систем Windows и Linux:

- браузер Mozilla FireFox;

- программу разработки презентаций OpenOffice Impress;

для операционной системы Windows:

- программу разработки презентаций Microsoft PowerPoint.

1.1. Техника безопасности и эргономика рабочего места

Пользователь компьютера должен соблюдать ряд мер по правильной организации своего рабочего места, в том числе:

- меры безопасности;
- санитарно-гигиенические нормы;
- правила эргономики;
- меры ресурсосбережения;
- грамотно выбирать технологические требования к оборудованию.

1.1.1. Безопасная работа с компьютером

Правила безопасной работы в компьютерном классе — это то, с чего начинается работа любого учащегося с школьным компьютером.

Правила безопасной работы в компьютерном классе

1. Не включайте неисправный компьютер. Если компьютер не работает или работает неправильно, то обязательно сообщите об этом учителю.
2. Когда компьютер включён, не трогайте его провода и разъёмы, а также розетки.
3. Нельзя есть, пить или жевать жвачку в компьютерном классе. Нельзя подходить к компьютеру с едой и напитками. При работе с компьютером руки должны быть сухими и чистыми.
4. Нельзя входить в компьютерный класс в уличной одежде, без сменной обуви, с грязным портфелем. Нельзя приносить в компьютерный класс посторонние предметы.
5. В компьютерном классе нужно вести себя дисциплинированно. Не передвигайтесь по классу без разрешения учителя. Не нажимайте кнопки или клавиши без разрешения учителя. Не мешайте другим ученикам.
6. Нажимая на клавиши, не надавливайте на них слишком сильно и не стучите по ним. Обращайтесь аккуратно со всеми устройствами компьютера.
7. За компьютером сидите прямо и ровно, на расстоянии вытянутой руки от экрана монитора. Не трогайте пальцами экран монитора — от этого на нём остаются грязные следы.

1.1.2. Санитарно-гигиенические нормы и эргономические требования

При работе с компьютером на пользователя воздействует целый ряд вредных факторов. Если не учитывать эти факторы и не соблюдать мер предосторожности и профилактики, то работа за компьютером может нанести ущерб здоровью пользователя. В числе таких факторов:

- повышенный уровень электромагнитных излучений и статического электричества, пониженная ионизация воздуха;
- перегрев портативного компьютера;
- специфические физические нагрузки на отдельные мышцы рук, спины и шеи при общей гиподинамии (недостатке движения);
- перенапряжение глаз.

Компьютер и его периферийные устройства представляют собой электрические приборы, при работе которых создаются хотя и достаточно слабые, но существенные при длительном пребывании вблизи них пользователя электромагнитные поля и излучения. Основными их источниками являются:

- блоки питания компьютера и периферийных устройств;
- блоки беспроводной связи (создание радиоизлучения);

- электрическая схема мониторов на базе ЭЛТ (электронно-лучевой трубки) и сама ЭЛТ;
- ЖК-экраны мониторов.

Основной вред от электромагнитных полей и излучений приносили мониторы на базе ЭЛТ, в составе которых имелись узлы высокого напряжения. В настоящее время вредное воздействие на пользователей компьютера существенно снижено за счёт почти повсеместной замены мониторов на базе ЭЛТ жидкокристаллическими.

Конструкторы компьютеров и периферийных устройств предпринимают специальные меры защиты пользователей от вредных излучений, а также стараются так размещать опасные узлы устройств, чтобы направлять вредные излучения не в сторону пользователя. Так, наибольшая мощность излучений обычно направлена от задней поверхности системного блока и монитора в сторону, противоположную месторасположению пользователя. Поэтому в компьютерном классе необходимо так размещать компьютеры, чтобы пользователи не оказывались в области воздействия вредных электромагнитных полей и излучений от другого компьютера (рис. 1.1).

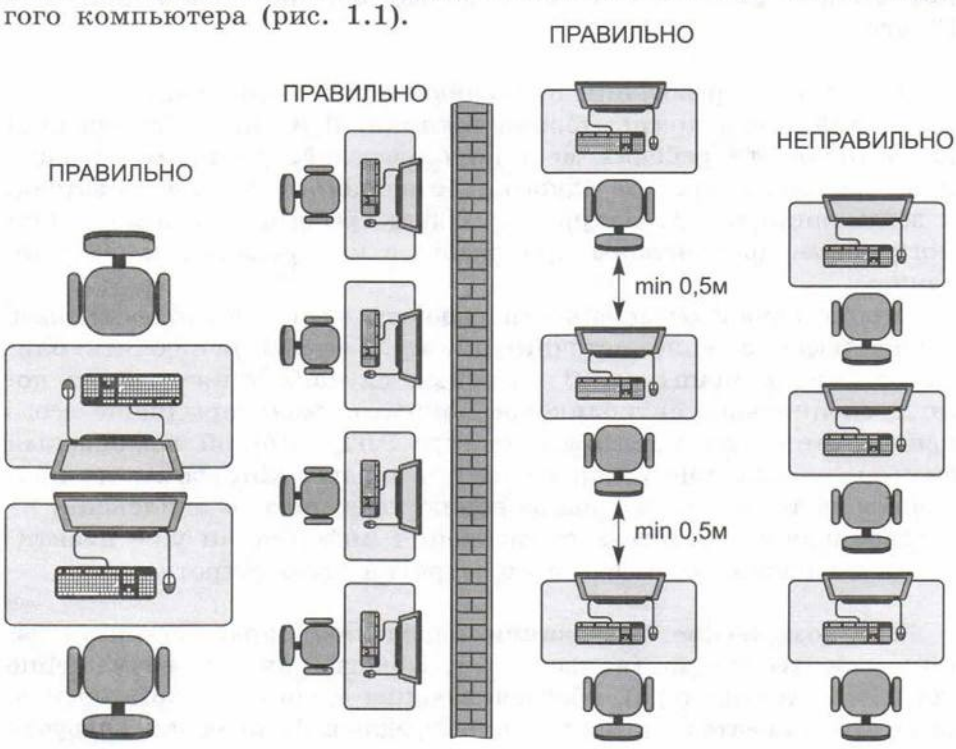


Рис. 1.1

У ноутбуков создаваемое ими электромагнитное излучение в основном направлено вниз, к поверхности стола. Поэтому крайне не рекомендуется при работе с ноутбуком размещать его на коленях, на бёдрах или на животе. В этом случае кроме электромагнитных излучений вредное воздействие на организм пользователя оказывает также нагрев нижней части корпуса ноутбука, где размещены вентиляционные щели для охлаждения видеокарты и процессора.

Из источников радиоизлучения вредное воздействие могут оказывать передатчики беспроводной связи Wi-Fi (в смартфонах, планшетах, ноутбуках, а также в беспроводных роутерах и точках доступа Wi-Fi) и Bluetooth, а также радиоизлучение при пользовании сотовой телефонной связью. В большинстве случаев это воздействие малоощутимо, но всё же следует ограничивать время разговоров по сотовому телефону (когда источник радиоизлучения располагается рядом с головой пользователя), а в некоторых случаях (при повышенной чувствительности к радиоизлучениям), возможно, придётся отказаться от использования беспроводной сети Wi-Fi в пользу обычной локальной сети Ethernet.

Отдельные требования предъявляются к помещению, в котором установлена компьютерная техника. В их числе требования к освещённости рабочих мест пользователей, к объёму помещения, приходящемуся на одного работающего в нём пользователя, к вентиляции и т. д. Например, особые требования к вентиляции могут быть предъявлены при работе с копировальным оборудованием.

Кроме общей освещённости рабочего места важно учитывать также расположение источников света, чтобы исключить блики от экрана монитора. В подобных случаях также может помочь специальное антибликовое покрытие монитора (чаще всего используется для мониторов на базе ЭЛТ). Другой же опасный фактор — мерцание изображения на экране, которое могло приводить не только к перенапряжению глаз, но и к эпилепсии, — в современных средствах отображения информации уже практически не проявляется и потому утратил свою остроту.

Эргономические требования определяют правила организации рабочего места (в частности, конструкцию и размещение устройств компьютера), обеспечивающие удобную и комфортную работу пользователя и исключение вредных физических нагрузок на его организм.

Основная часть эргономических требований касается правильной посадки за компьютером (рис. 1.2).

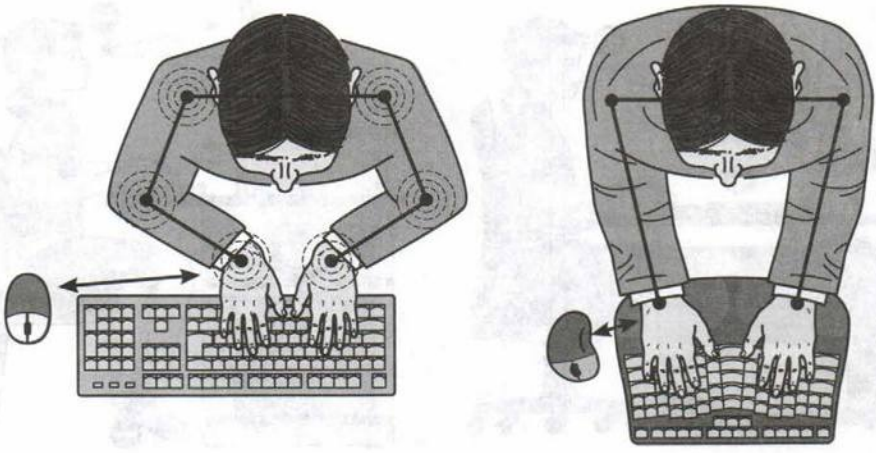
Рис. 1.2

При работе с компьютером нужно, чтобы спина и шея были ровными, спина опиралась на спинку стула, причём спинка стула должна иметь выступ в области поясничного изгиба. Локти, колени и ступни должны быть согнуты под прямым углом, руки — свободно лежать на столе, на клавиатуре и мыши. Монитор нужно расположить так, чтобы линия взгляда была перпендикулярна поверхности экрана и направлена под углом примерно в 30 градусов вниз от горизонтали, а расстояние от глаз до экрана составляло не менее 50 см.

Для обеспечения правильной посадки за компьютером необходимо использовать специальную мебель — столы (парты) с регулировкой по высоте (по росту учащихся), а также специальные стулья с регулируемой высотой сиденья и наклоном спинки.

Важным является и учёт эргономических требований к конструкции основных устройств ввода, используемых при работе с компьютером, — клавиатуры и мыши. Так, при работе с обычной клавиатурой пользователь вынужден постоянно изгибать кисти рук, что приводит к перенапряжению мышц и может при длительной работе с клавиатурой приводить к воспалению каналов

сухожилий (так называемый «туннельный синдром»). Чтобы избежать этого, для пользователей, которые в процессе работы активно занимаются вводом и редактированием текстов, разработаны специальные эргономичные клавиатуры (рис. 1.3).



Обычная клавиатура

Эргономичная клавиатура

Рис. 1.3

При работе с мышью важно правильно располагать руку на её корпусе (рис. 1.4), чтобы рука в запястье была прямой.

Следует учитывать, что длительная работа за компьютером приводит к усталости глаз. Особенно это касается компьютерных игр, в процессе которых играющий практически неотрывно следит за изображением на экране монитора. Перенапряжение глаз приводит к сухости поверхности роговицы (из-за более редких морганий), а также может приводить к близорукости.

Не менее важной с точки зрения эргономических требований является и организация рабочего распорядка. Время непрерывной работы за компьютером должно быть ограничено. Так, учащиеся начальных классов могут работать за компьютером не более 10 минут непрерывно. Ученики с 5 по 7 классы должны проводить за компьютером не дольше 15 минут, а с 7 по 9 класс — не дольше 20 минут. В 10–11 классах время работы за компьютером не должно превышать 30 минут на первом уроке и 20 минут — на втором.

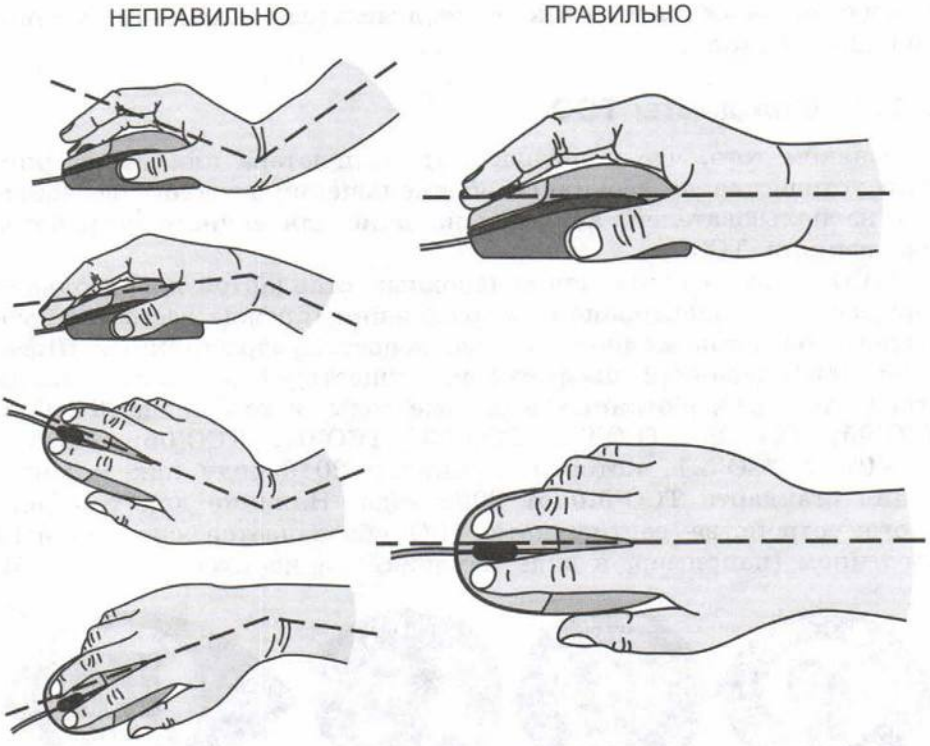


Рис. 1.4

Несоблюдение этих ограничений и длительное непрерывное сидение за компьютером может приводить к гиподинамии.

Для избежания вредного влияния на пользователя длительной работы за компьютером необходимо в паузах проводить разминки, выполняя простые физические упражнения, а также периодически (каждые 5–10 минут) выполнять простейшую зарядку для глаз:

- переводить несколько раз взгляд с ближних предметов вдаль и обратно;
- несколько раз перемещать направление взгляда влево-вправо, вверх-вниз и по кругу по часовой стрелке и против часовой стрелки;
- закрыть глаза и несколько минут посидеть зажмурившись.

Все указанные выше санитарно-гигиенические и эргономические требования к рабочему помещению, к организации рабочего места, к компьютерной технике и к регламентированию работы с ней оговорены в специальном документе — Санитарных правилах и нормах **(СанПиН)** 2.2.2/2.4.1340-03 «Гигиенические требования

к персональным электронно-вычислительным машинам и организации работы».

1.1.3. Стандарты TCO

Признаком того, что конструкторы компьютера или периферийного устройства позаботились об уменьшении вредного воздействия на пользователей, является наличие для данного устройства сертификата TCO.

TCO — это группа международных стандартов добровольной сертификации электронного оборудования (прежде всего компьютерного) на эргономичность и безопасность, разработанных Шведской конфедерацией профсоюзов. Существует несколько таких стандартов, разработанных в разные годы, в том числе: TCO'92, TCO'95, TCO'99, TCO'01, TCO'03, TCO'04, TCO'05, TCO'06, TCO'07 и TCO'5.1, который принят в 2010 году как модификация стандарта TCO'5.0 от 2009 года. Наличие для того или иного устройства сертификата TCO обозначается специальным логотипом (например, в виде наклейки на корпусе — рис. 1.5).

Рис. 1.5

1.1.4. Ресурсосбережение

Ещё один важный фактор, влияющий на правильный выбор компьютерного оборудования, — это ресурсосбережение. Прежде всего речь идёт об энергосбережении, т. е. об уменьшении суммарного расхода электроэнергии при работе с компьютером и его устройствами.

Одна из возможностей энергосбережения заключается в создании энергоэкономичных устройств, требующих меньше энергии для своей работы. Таковы, например, современные многоядерные процессоры, быстродействие которых достигается не за счёт роста тактовой частоты (что приводит к увеличению нагрева процессора и к росту расходов энергии на его работу и на его охлаждение), а за счёт параллелизации вычислительных процессов. Значительно меньший расход энергии требуется для работы современных жёстких дисков, а накопители на базе

флеш-памяти (в том числе современные диски SSD) потребляют минимум энергии.

Другая, ещё более существенная, возможность заключается в переводе устройств, которые временно не используются, в режим пониженного энергопотребления (в «спящий» режим). Например, для компьютера в этом режиме отключается отображение на экране монитора, выключаются двигатели накопителей (в том числе жёсткого диска), а процессор переводится на пониженную частоту работы. В таком режиме компьютер практически не потребляет электроэнергии. Аналогично, в «спящий» режим при достаточно длительной паузе в работе переводятся современные принтеры и копировальные аппараты.

Существует международный стандарт энергоэффективности электронных устройств — Energy Star (рис. 1.6). Наличие соответствующего логотипа позволяло утверждать, что при одной и той же функциональности устройство потребляет энергии на 20–30% меньше, чем его аналоги, не соответствующие стандартам Energy Star. Однако требования этого стандарта были определены недостаточно строго, поэтому в настоящее время требования энергоэффективности также включены в стандарты TCO.

Рис. 1.6

Вопросы и задания

1. В чём заключаются правила безопасной работы в компьютерном классе?
2. Какие вредные факторы могут действовать на пользователя при работе с компьютером?
3. Какие устройства компьютера порождают вредные электромагнитные излучения?
4. Как необходимо размещать компьютеры в компьютерном классе? Почему?
5. Какие правила нужно соблюдать при работе с портативными компьютерами (ноутбуками, планшетами, смартфонами) для защиты от электромагнитных излучений?
6. Какие санитарно-гигиенические требования предъявляются к помещению и рабочему месту?
7. Какие эргономические требования нужно соблюдать при работе с компьютером? К чему приводит их несоблюдение?
8. Что такое TCO? Что означает наклейка с логотипом TCO?
9. Какие меры ресурсосбережения чаще всего применяются для компьютеров и компьютерных устройств?

1.2. Информация. Измерение информации

Слово **«информация»** происходит от латинского слова *informatio*, что в переводе означает «сведения», «разъяснение», «ознакомление». Понятие «информация» является базовым в курсе информатики, невозможно дать его определение через другие понятия. В геометрии, например, невозможно выразить содержание базовых понятий «точка», «луч», «плоскость» через более простые понятия. Содержание основных, базовых понятий в любой науке должно быть пояснено на примерах или выявлено путём их сопоставления с содержанием других понятий.

В случае с понятием «информация» проблема его определения ещё более сложна, так как это понятие является общенаучным. Оно используется в различных науках (в информатике, кибернетике, биологии, физике и др.), при этом в каждой науке понятие «информация» связано с различными системами понятий.

Информация в неживой природе. В физике, которая изучает неживую природу, информация является мерой упорядоченности системы по шкале «хаос—порядок». Один из основных законов классической физики утверждает, что замкнутые системы, в которых отсутствует обмен веществом и энергией с окружающей средой, стремятся с течением времени перейти из менее вероятного упорядоченного состояния в наиболее вероятное хаотическое состояние.

В соответствии с такой точкой зрения физики в конце XIX века предсказывали, что нашу Вселенную ждёт «тепловая смерть», т. е. молекулы и атомы равномерно распределятся в пространстве и какие-либо изменения и развитие прекратятся.

Однако современная наука установила, что некоторые законы классической физики, справедливые для макротел, нельзя применять для микро- и мегамира. Согласно современным научным представлениям, наша Вселенная является динамически развивающейся системой, в которой постоянно происходят процессы усложнения структуры.

Таким образом, с одной стороны, в неживой природе в замкнутых системах идут процессы в направлении от порядка к хаосу (в них информация уменьшается). С другой стороны, в процессе эволюции Вселенной в микро- и мегамире возникают объекты со всё более сложной структурой и, следовательно, информация, являющаяся мерой упорядоченности элементов системы, возрастает.

Информация в живой природе. Живые системы в процессе своего развития способны повышать сложность своей структуры, т. е. увеличивать информацию, понимаемую как меру

упорядоченности элементов системы. Так, растения в процессе фотосинтеза потребляют энергию солнечного излучения и строят сложные органические молекулы из простых неорганических молекул.

Животные подхватывают эстафету увеличения сложности живых систем, поедают растения и используют растительные органические молекулы в качестве строительного материала при создании ещё более сложных молекул.

Биологи образно говорят, что «живое питается информацией», создавая, накапливая и активно используя информацию.

Целесообразное поведение живых организмов и выживание популяций животных во многом строятся на основе получения информационных сигналов. Такие сигналы могут иметь различную физическую или химическую природу: звук, свет, запах и др.

Генетическая информация представляет собой набор генов, каждый из которых отвечает за определённые особенности строения и функционирования организма. При этом дети не являются точными копиями своих родителей, так как каждый организм обладает уникальным набором генов, которые определяют различия в строении и функциональных возможностях.

Человек и информация. Человек существует в информационном пространстве, он постоянно получает информацию из окружающего мира с помощью органов чувств, хранит её в своей памяти, анализирует с помощью мышления и обменивается информацией с другими людьми.

Основные **информационные процессы** — это получение, хранение, обработка и передача информации.

Человек не может жить вне общества. В процессе общения с другими людьми он передаёт и получает информацию в форме сообщений. На заре человеческой истории для передачи информации сначала использовался язык жестов, а затем появилась устная речь. В настоящее время обмен сообщениями между людьми производится с помощью сотен естественных языков (русского, английского и пр.).

Для того чтобы человек мог правильно ориентироваться в окружающем мире, информация должна быть полной и точной. Получение полной и точной информации о природе, обществе и технике — это основная задача современной науки. Процесс систематического научного познания окружающего мира, в котором информация рассматривается как знания, начался с середины XV века после изобретения книгопечатания.

Информация в технике. Функционирование систем управления техническими устройствами связано с процессами получения (приёма), хранения, обработки и передачи информации. Системы управления встроены практически во всю современную бытовую технику, станки с числовым программным управлением, транспортные средства и т. д.

Системы управления могут обеспечивать функционирование технической системы по заданной программе. Например, системы программного управления обеспечивают выбор режимов стирки в стиральной машине, обработки детали на станке с программным управлением и т. д.

В некоторых случаях главную роль в процессе управления выполняет человек, в других управление осуществляет встроенный в техническое устройство микропроцессор или подключённый компьютер.

В современном информационном обществе главным ресурсом является информация, использование которой базируется на информационных и коммуникационных технологиях.

Информационные и коммуникационные технологии (ИКТ) являются совокупностью методов, устройств и производственных процессов, используемых обществом для сбора, хранения, обработки и распространения информации.

Количество информации как мера уменьшения неопределённости знания. Процесс познания окружающего мира приводит к накоплению информации в форме знаний (фактов, научных теорий и т. д.). Получение новой информации приводит к расширению знания или, ещё говорят, к уменьшению неопределённости знания. Если некоторое сообщение приводит к уменьшению неопределённости нашего знания, то можно говорить, что такое сообщение содержит информацию.

Чем более неопределённа первоначальная ситуация (больше количество возможных информационных сообщений), тем больше мы получим новой информации при получении информационного сообщения (в большее количество раз уменьшится неопределённость знания).

Количество информации можно рассматривать как меру уменьшения неопределённости знания при получении информационных сообщений.

Рассмотрим вопрос об определении количества информации более подробно на конкретных примерах.

Пусть у нас имеется монета, которую мы бросаем на ровную поверхность. С равной вероятностью произойдёт одно из двух возможных событий — монета окажется в одном из двух положений: «орёл» или «решка».

Можно говорить, что события равновероятны, если при возрастающем числе опытов количества выпадений «орла» и «решки» постепенно сближаются. Например, если мы бросим монету 10 раз, то «орёл» может выпасть 7 раз, а решка — 3 раза, если бросим монету 100 раз, то «орёл» может выпасть 60 раз, а «решка» — 40 раз, если бросим монету 1000 раз, то «орёл» выпадет 520 раз, а «решка» — 480 и т. д. В итоге при очень большой серии опытов количества выпаданий «орла» и «решки» практически сравняются.

Перед броском существует неопределённость нашего знания (возможны два события — и два возможных информационных сообщения об этих событиях), и как упадёт монета, предсказать невозможно. После броска наступает полная определённость, так как мы видим (получаем зрительное сообщение), что монета в данный момент находится в определённом положении (например, выпал «орёл»). Это сообщение приводит к уменьшению неопределённости нашего знания в два раза, так как из двух возможных равновероятных событий реализовалось одно (рис. 1.7).

Рис. 1.7

В окружающей действительности достаточно часто встречаются ситуации, когда может произойти некоторое количество равновероятных событий. Так, при бросании равносторонней четырёхгранной пирамиды существуют 4 равновероятных события, а при бросании шестигранного игрального кубика — 6 равновероятных событий.

Чем больше количество возможных событий (информационных сообщений), тем больше начальная неопределённость нашего знания и, соответственно, тем большее количество информации будет содержать сообщение о результатах опыта.

Рассмотренный выше подход к информации как мере уменьшения неопределённости знания позволяет количественно измерять информацию. Существует формула, которая связывает между собой количество возможных информационных сообщений N и количество информации i, которое несёт полученное сообщение:

$$N = 2^i. \qquad (1.1)$$

Для количественного выражения любой величины необходимо сначала определить единицу измерения.

За **единицу количества информации** принимается такое количество информации, которое содержится в информационном сообщении, уменьшающем неопределённость знания в два раза. Такая единица носит название **бит**.

Если вернуться к опыту с бросанием монеты, то здесь неопределённость уменьшается именно в два раза (из двух возможных событий реализуется одно) и, следовательно, количество полученной информации равно 1 биту.

Минимальной единицей измерения количества информации является бит, а следующей по величине единицей — **байт**, причём

$$1 \text{ байт} = 8 \text{ бит} = 2^3 \text{ бит}.$$

В информатике система образования кратных единиц измерения количества информации использует коэффициент 2^n. Кратные байту единицы измерения количества информации вводятся следующим образом:

1 Кбайт $= 2^{10}$ байт $= 1024$ байт;
1 Мбайт $= 2^{10}$ Кбайт $= 1024$ Кбайт;
1 Гбайт $= 2^{10}$ Мбайт $= 1024$ Мбайт.

Определение количества информации на основе уменьшения неопределённости нашего знания рассматривает информацию с точки зрения содержания: её понятности и новизны для человека. С этой точки зрения, в опыте по бросанию монеты одинаковое количество информации содержится и в зрительном образе упавшей монеты, и в коротком сообщении «Орёл», и в длинной фразе «Монета упала на поверхность земли той стороной вверх, на которой изображен орёл».

Алфавитный подход к определению количества информации. При алфавитном подходе к определению количества информации мы отвлекаемся от содержания информации и рассматриваем информационное сообщение как последовательность знаков определённой знаковой системы.

Формула (1.1) связывает между собой количество возможных информационных сообщений N и количество информации i, которое несёт полученное сообщение. Тогда в рассматриваемой ситуации N — это количество знаков в алфавите знаковой системы, а i — количество информации, которое несёт каждый знак:

$$N = 2^i.$$

С помощью данной формулы можно, например, определить количество информации, которое несёт знак в двоичной знаковой системе:

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

Таким образом, в двоичной знаковой системе знак несёт 1 бит информации. Интересно, что сама единица измерения количества информации бит (*bit*) получила свое название от английского словосочетания *BInary digiT* — «двоичная цифра».

Чем большее количество знаков содержит алфавит знаковой системы, тем большее количество информации несёт один знак.

Вопросы и задания

1. Приведите примеры перехода от хаоса к порядку (увеличения информации) в окружающем мире.
2. Приведите примеры перехода от порядка к хаосу (уменьшения информации) в окружающем мире.
3. Каковы должны быть свойства информации, представленной в форме знаний?
4. Приведите примеры систем управления техническими устройствами.
5. Приведите примеры информационных сообщений, которые несут 1 бит информации.

1.3. Передача информации

Важными отличиями человека от животных являются развитые абстрактное мышление и речь, в связи с чем информация является для человеческого общества важнейшим ресурсом. Не зря говорят: «Кто владеет информацией, тот владеет миром»[1]. Именно обладание своевременно полученной, достоверной информацией позволяет сделать свою деятельность успешной, а во многих случаях даже сохранить здоровье и спасти жизнь людей (например, благодаря своевременной эвакуации из зоны будущего стихийного бедствия). Не менее важна для человеческого общества эффективная обработка информации, а особенно возможность удобной и лёгкой коммуникации: обмена информацией между людьми, быстрого распространения информации (вещания) и т. д.

Ранее вы уже познакомились с понятием «информация» и с принципами измерения количества информации. Вы также знакомы с основными информационными процессами — получения (сбора), хранения и обработки информации. Теперь мы более подробно рассмотрим **процесс передачи информации**, его особенности и принципы реализации коммуникаций в современном обществе.

Процесс передачи информации всегда предполагает наличие по крайней мере трёх составляющих: источника информации, приёмника информации и канала передачи информации (рис. 1.8).

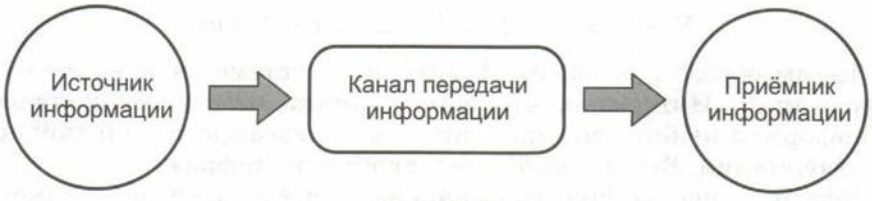


Рис. 1.8

Источник информации — это субъект, который предоставляет (сообщает) имеющуюся у него информацию другим субъектам, выступающим в роли **приёмников информации**. При этом информация передаётся в форме сигналов (знаков) при помощи некоторого материального **носителя** (например, при помощи радиоволн или бумажной поздравительной почтовой открытки). Используемый носитель, а также необходимое техническое оборудование представляют собой **канал передачи информации (информационный канал, канал связи)**.

[1] Цитата принадлежит Натану Ротшильду — английскому банкиру, бизнесмену и финансисту.

Передача информации может осуществляться:
- только в одном направлении (от источника к приёмнику);
- в обоих направлениях — такой процесс называют обменом информацией, а участвующие в нём субъекты поочерёдно или даже одновременно являются и источниками, и приёмниками информации;
- вещание — односторонняя передача информации от одного источника сразу множеству приёмников.

Пример 1. Разговор по телефону — процесс обмена информацией:
- источник и приёмник информации — разговаривающие абоненты (их роли взаимно меняются в зависимости от того, кто из них в данный момент говорит, а кто — слушает);
- канал передачи информации включает в себя телефонные аппараты, оборудование телефонных станций и используемые линии телефонной связи (проводные или беспроводные).

Пример 2. Телевизионная передача последних новостей — процесс вещания:
- источник информации — ведущий передачи или диктор;
- приёмники информации — телезрители;
- канал передачи информации включает в себя телекамеру и оборудование телестудии, радиосвязь и телевизионные приёмники (телевизоры).

1.3.1. Сигнал. Кодирование и декодирование

Чаще всего передача информации осуществляется в форме сигналов. **Сигнал** — это символ (знак), имеющий определённое смысловое значение согласно предварительной договорённости между источником и приёмником информации. Иными словами, обе стороны, участвующие в обмене информацией, должны понимать смысл используемых сигналов и располагать оборудованием для преобразования сигналов, доступных им для восприятия, в форму, требуемую для передачи при помощи выбранного носителя.

Пример 3 (рис. 1.9). При радиовещании происходят следующие преобразования сигналов:
- речь диктора (звуковые сигналы);
- электрические сигналы, полученные при преобразовании звуковых сигналов в микрофоне;
- радиосигналы, полученные при преобразовании электрических
- сигналов в радиопередатчике;

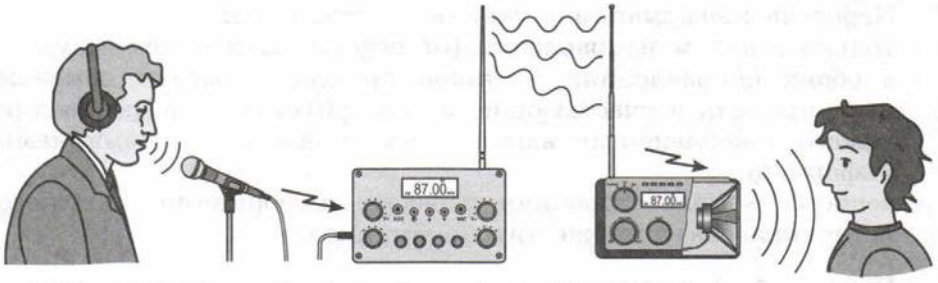


Рис. 1.9

- электрические сигналы после преобразования радиосигналов в радиоприёмнике;
- звуковые сигналы после преобразования электрических сигналов в динамике радиоприёмника.

Преобразование информации в форму сигналов называют **кодированием**. Кодирование осуществляется в соответствии с некоторым набором (алфавитом) символов (знаков). Например, речь представляет собой информацию, закодированную при помощи звуков; письменный текст — это информация, закодированная при помощи букв и других символов; цифровая информация в компьютере закодирована при помощи двоичных нулей и единиц и т. д.

Обратное преобразование информации называют **декодированием**. Например, декодирование информации происходит при чтении текста или прослушивании речи собеседника.

Часто понятия «кодирование» и «декодирование» рассматривают в более узком, техническом смысле, подразумевая под кодированием процесс преобразования информации, форма представления которой доступна для восприятия человеком без использования технических устройств, в другую форму, удобную для передачи по используемому информационному каналу. Например, под кодированием понимается преобразование человеческой речи в электрические, а далее — в радиосигналы. Соответственно, под декодированием в этом случае понимается обратное преобразование информации из формы, используемой при передаче по информационному каналу, в форму, доступную для восприятия человеком, — например, из радио- и электрических сигналов в звуки музыки или речи.

Отдельной разновидностью кодирования и декодирования является **шифрование** и **дешифрование**. Они используются для того, чтобы защитить информацию от несанкционированного доступа к ней посторонних лиц, сделать информацию секретной. При шифровании и дешифровании используются такие коды и

такие методы использования этих кодов, которые известны только источнику (отправителю) и приёмнику (получателю) информации. Например, при шифровании в качестве кодов могут использоваться обычные буквы русского алфавита, но переставленные местами по определённым правилам. Так, в *шифре Цезаря* каждая буква заменяется на другую, отстоящую от неё в алфавите на одно и то же фиксированное число позиций (рис. 1.10). Алфавит при этом считается замкнутым в кольцо.

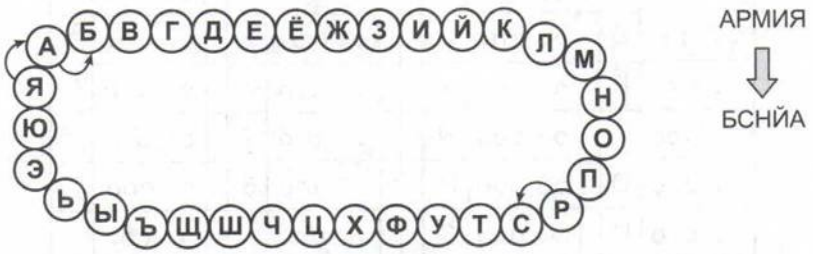


Рис. 1.10

1.3.2. Равномерные и неравномерные коды. Условие Фано

Кодирование информации может быть произведено так, что каждый получаемый код имеет одну и ту же длину. Например, французский инженер и изобретатель Эмиль Бодо в 1870 году разработал для своего телеграфа особый код — код Бодо (рис. 1.11), в котором каждая буква, цифра или другой знак кодировался при помощи комбинации из пяти сигналов. Такая телеграфная система могла, например, передавать информацию по пяти параллельным электрическим линиям, когда тот или иной символ кодировался наличием или отсутствием электрического тока в той или иной из этих пяти линий.

Очевидно, что каждому символу соответствует код одной и той же длины, а суммарный размер закодированной текстовой строки может быть вычислен умножением количества символов на длину одного кода. Такой код называют **равномерным кодом**, или **кодом постоянной длины**, а кодирование, соответственно, называют **равномерным кодированием**.

Равномерное кодирование реализовать достаточно просто. Но такой способ кодирования не всегда оптимален. Вспомним, что различные буквы и другие символы встречаются в тексте с разной частотой. Так, в текстах на русском языке буквы «А», «О»,

Управляющие символы	
o. ...	пробел, перейти к таблице букв
.o ...	пробел, перейти к таблице цифр
oo ...	удалить последний знак

Таблица букв						Таблица цифр				
.. o..	A		oo o..	K		.. o..	1		o. o..	.
.. oo.	É		oo oo.	L		.. .o.	2		o. .o.	9/
.. .o.	E		oo .o.	M		.. ..o	3		o. ..o	7/
.. .oo	I		oo .oo	N		.. o.o	4		o. o.o	2/
.. ooo	O		oo ooo	P		.. ooo	5		o. ooo	'
.. o.o	U		oo o.o	Q		.. oo.	1/		o. oo.	:
.. ..o	Y		oo ..o	R		.. .oo	3/		o. .oo	?
.o ..o	B		o. ..o	S		.o o..	6		oo o..	(
.o o.o	C		o. o.o	T		.o .o.	7		oo .o.	)
.o ooo	D		o. ooo	V		.o ..o	8		oo ..o	-
.o .oo	F		o. .oo	W		.o o.o	9		oo o.o	/
.o .o.	G		o. .o.	X		.o ooo	0		oo ooo	+
.o oo.	H		o. oo.	Z		.o oo.	4/		oo oo.	=
.o o..	J		o. o..	—		.o .oo	5/		oo .oo	£

Рис. 1.11

«И» встречаются гораздо чаще, чем, например буква «Ы» или «Ъ». При кодировании такого текста более выгодно назначать буквам коды различной длины в зависимости от частоты встречаемости этих букв. Для «А», «О», «И» и других часто используемых букв можно использовать более короткие коды, а для редко используемых символов — коды большей длины. Такие коды называют **неравномерными**, или **кодами переменной длины**, а кодирование — **неравномерным**.

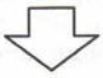
Примером неравномерного кода является **азбука Морзе** (рис. 1.12).

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

Рис. 1.12

В компьютере информация представлена в виде последовательностей нулей и единиц, поэтому в качестве кодов обычно используются последовательности именно этих цифр. Например, в азбуке Морзе можно заменить точки нулями, а тире — единицами. Тогда строка текста «КОД МОРЗЕ» может быть закодирована так — рис. 1.13.

К	О	Д		М	О	Р	З	Е
–.–	–––	–..		––	–––	.–.	––..	.
101	111	100		11	111	010	1100	0



10111110011111010110000

Рис. 1.13

При этом возникает вопрос: как обеспечить однозначность декодирования такого кода — так, чтобы полученную при кодировании «сплошную» последовательность нулей и единиц можно было разделить на коды, соответствующие «правильным» буквам. Для обеспечения однозначности декодирования закодированного сообщения необходимо выбирать неравномерные коды такими, что при «разборе» закодированной последовательности слева направо (или справа налево) каждый раз можно было отделять от неё очередной код единственным способом.

Правило, согласно которому необходимо формировать такой набор кодов, достаточно простое: **никакой код не должен совпадать с началом никакого другого кода**. Это правило называют **условием Фано** (в честь Роберта Фано — американского учёного-информатика итальянского происхождения).

Примеры

1. Набор кодов, соответствующий условию Фано:

А	Б	В
0	10	11

2. Набор кодов, не соответствующий условию Фано:

А	Б	В
0	10	100

В этом наборе кодов условие Фано нарушено для кодов, соответствующих буквам «Б» (10) и «В» (100), так как код буквы «Б» совпадает с началом кода буквы «В».

В результате, например, строка 100 может быть декодирована неоднозначно — или как одна буква «В», или как пара букв «БА».

1.3.3. Искажение информации

Процесс передачи информации очень редко осуществляется в идеальных условиях. Как правило, при этом возможны различные внешние помехи или неисправности оборудования информационного канала, которые могут приводить к **искажению** (повреждению) передаваемой информации. Чтобы защитить информацию от искажения, вызванного помехами, при её передаче принимают различные специальные меры. Например, при передаче цифровой

информации (представленной в виде последовательности чисел) в качестве таких мер могут выступать:

- подсчёт суммы всех передаваемых числовых значений (так называемой *контрольной суммы*) — её значение вычисляется источником информации и передаётся по каналу связи отдельно, а затем сравнивается с суммой, полученной последовательности чисел, независимо подсчитанной приёмником информации; если обе суммы совпали, то считается, что информация была передана без искажений;
- многократная передача одной и той же информации — поскольку обычно внешние помехи возникают случайно, достаточно маловероятно, что одно и то же искажение информации произойдёт в одном и том же её фрагменте; если передать одну и ту же последовательность чисел трижды и сравнить три принятых «образца» последовательностей, то в случае их несовпадения достаточно выбрать число, которое было одинаково принято дважды (считая единственное несовпадающее число ошибкой).

1.3.4. Скорость передачи информации

Важным количественным показателем является **скорость передачи информации**. Под этой величиной понимается количество информации, передаваемое по каналу связи за одну секунду.

Скорость передачи информации может быть выражена в битах в секунду (бит/с) или в кратных величинах: байтах в секунду (байт/с), Кбайт/с, Мбайт/с и т. д.

Взаимосвязь между количеством передаваемой информации, скоростью её передачи и затрачиваемым на это временем та же самая, что и между пройденным путём, скоростью и временем движения тела:

$$I = v \cdot t,$$

где I — количество информации, v — скорость передачи информации, t — время передачи.

Пример 4. Какова скорость передачи информации (в байтах в секунду), если файл объёмом 10 Кбайт был передан по каналу связи за 20 секунд?

Решение

Количество информации I составляет 10 Кбайт = 10 240 байт.

Время передачи t равно 20 секундам.

Следовательно, скорость передачи информации можно вычислить по формуле: $v = I/t = 10\,240/20 = 512$ байт/с.

Пример 5. За сколько минут Андрей сможет передать учителю текст своего реферата, если скорость передачи информации через компьютерный канал связи составляет 2048 байт/с? Количество минут нужно округлить до ближайшего целого значения. Реферат состоит из 25 страниц; на каждой странице имеется по 60 строк текста; в каждой строке — по 80 символов. Используется кодировка текста Unicode, при которой каждый символ кодируется 2 байтами.

Решение

Сначала определим количество информации в реферате.

Всего символов в нём: 25 (страниц) · 60 (строк) · 80 (символов в строке) = 120 000 (символов).

Пусть каждый символ кодируется двумя байтами (кодировка Юникод). Следовательно, объём информации составит 120 000 (символов) · 2 (байта) = 240 000 (байт).

Скорость передачи информации составляет 2048 байт/с.

Теперь можно вычислить время передачи информации по формуле: $t = I/v$ = 240 000 (байт)/2048 (байт/с) ≈ 117 (с) ≈ 2 (мин).

Важным понятием также является **пропускная способность** канала связи — скорость передачи информации по нему, достижимая в идеальных условиях (в частности, при отсутствии помех). Реальная скорость передачи информации по такому каналу может быть ниже, чем его пропускная способность (например, из-за многократной передачи информации для борьбы с её искажением при помехах).

Вопросы и задания

1. Укажите основные составляющие информационного процесса передачи информации.
2. Охарактеризуйте основные разновидности процесса передачи информации и укажите их различия.
3. Что такое кодирование и декодирование? Для чего требуются эти операции?
4. Каким способом можно бороться с искажением информации из-за помех?
5. Как и в каких величинах измеряется скорость передачи информации?
6. По первому каналу связи за 30 с можно передать файл объёмом 50 Кбайт. Скорость второго канала связи вдвое больше. Каков объём файла, который можно передать по второму каналу связи за то же самое время?

Практическая работа 1.1

Шифрование и дешифрование

Задание. Найти в Интернете информацию о различных кодах и шифрах. Освоить различные операции кодирования и декодирования, шифрования и дешифрования. Подготовить презентацию о выбранном методе шифрования (в рамках коллективного проекта).

Варианты выполнения работы:

- в операционной системе Windows или в операционной системе Linux;
- просмотр веб-страниц в различных браузерах (Mozilla Firefox, Opera, Chrome или др.);
- подготовка презентаций в программе Microsoft Power Point или OpenOffice Impress.

Работа с онлайновым кодировщиком Морзе

Выполним кодирование и декодирование информации при помощи онлайн-переводчика азбуки Морзе.

1. Запустить браузер в операционной системе Windows (например, Mozilla Firefox) или в операционной системе Linux и в поле *Адрес*: ввести **http://telegraphist.ru/book/morse-translator** (онлайн-переводчик азбуки Морзе).
2. Ознакомиться с содержимым окна онлайн-переводчика (рис. 1.14).

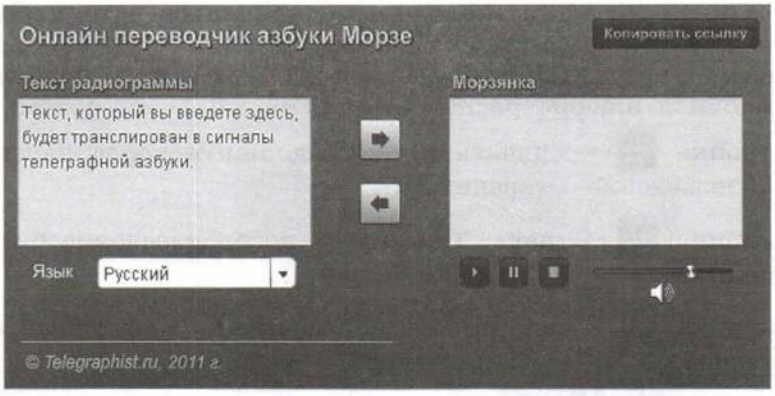


Рис. 1.14

3. Выбрав в раскрывающемся списке под левым полем язык сообщения (*Русский* или *Английский*), ввести в левое поле текст на выбранном языке, который нужно закодировать в виде знаков азбуки Морзе, например: «Россия — священная наша держава».
4. Для выполнения операции кодирования текста щёлкнуть мышью на кнопке ➡. Ознакомиться с появившейся в правом поле записью текста при помощи азбуки Морзе (рис. 1.15).

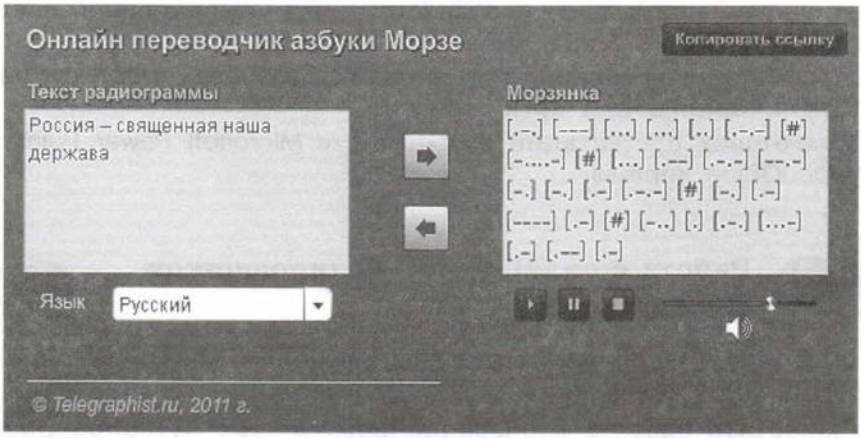


Рис. 1.15

Отличия генерируемой онлайн-переводчиком записи от традиционной: для наглядности знаки, соответствующие каждой букве, заключаются в квадратные скобки, а промежутки между словами обозначаются специальной записью [#].

5. При наличии на компьютере аудиоколонок или наушников прослушать закодированную информацию в виде звуковых сигналов азбуки Морзе («морзянки»). Для этого используется кнопочная панель, расположенная под правым полем:
   - кнопка ▶ — начать (или продолжить после паузы) воспроизведение «морзянки»;
   - кнопка ⏸ — пауза (повторное воспроизведение будет выполняться с места приостановки);
   - кнопка ■ — остановка воспроизведения (повторное воспроизведение будет выполняться с начала записи);
   - панель  — управление громкостью.

6. Выполнить ручное кодирование текста на русском языке в запись азбуки Морзе, используя таблицу (строчные и прописные буквы не различаются) — рис. 1.16.

Азбука Морзе				
А •—	К —•—	Ф ••—•	1 •————	. ••••••
Б —•••	Л •—••	Х ••••	2 ••———	. •—•—•—
В •——	М ——	Ц —•—•	3 •••——	; —•—•—•
Г ——•	Н —•	Ч ———•	4 ••••—	: ———•••
Д —••	О ———	Ш ————	5 •••••	? ••——••
Е •	П •——•	Щ ——•—	6 —••••	!—— ••——
Ж •••—	Р •—•	Ь,Ъ—••—	7 ——•••	–——•••• —
З ——••	С •••	Ы —•——	8 ———••	« •—••—•
И ••	Т —	Э ••—••	9 ————•	(—•—•—•—
Й •———	У ••—	Ю ••——	0 —————	/ —••—•
		Я •—•—		

Рис. 1.16

7. Полученный текст ввести в правое поле онлайн-переводчика с использованием принятых в нём обозначений (квадратных скобок и символа #), например:

   Россия — родина моя!

   [.-.] [---] [...] [...] [..] [.-.-] [#] [-....-] [#] [.-.] [---] [-..] [..] [-.] [.-] [#] [--] [---] [.-.-] [--..--]

8. Проверить правильность выполненного кодирования, щёлкнув мышью на кнопке ⬅ и прочитав в левом поле декодированное сообщение (пример — на рис. 1.17).

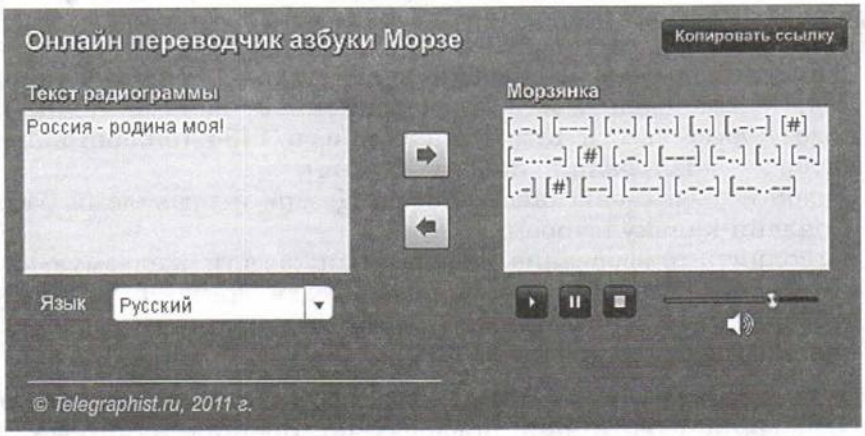


Рис. 1.17

9. При наличии на компьютере аудиоколонок или наушников прослушать закодированную информацию в виде звуковых сигналов «морзянки».
10. Оценить профессионализм радистов первой половины XX века, которые должны были с подобной скоростью по памяти выполнять кодирование текста в сигналы азбуки Морзе при помощи специального выключателя — телеграфного ключа (рис. 1.18), воспринимать полученные сигналы «морзянки» на слух и по памяти выполнять декодирование информации в обычный текст.

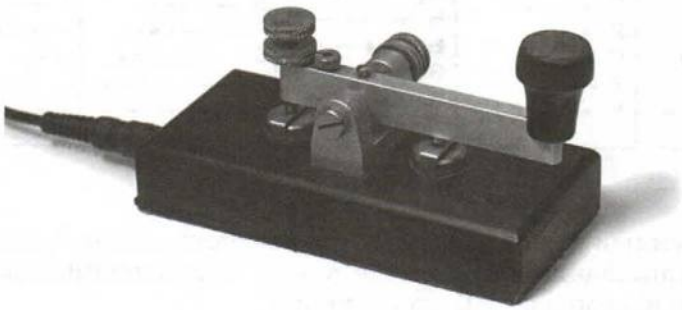


Рис. 1.18

Работа с онлайновым кодировщиком шифра Цезаря

Выполним шифрование информации при помощи онлайн-кодировщика шифра Цезаря.

1. Запустить браузер в операционной системе Windows (например, Mozilla Firefox) или в операционной системе Linux и в поле *Адрес*: ввести **http://planetcalc.ru/1434** (онлайн-калькулятор — кодировщик шифра Цезаря).
2. Ознакомиться с описанием шифра Цезаря и правилами работы с онлайн-калькулятором.
3. Выполнить шифрование информации: ввести желаемую строку текста (например: «Единожды предав, никогда не станешь другом!») в поле онлайн-калькулятора и щёлкнуть мышью на кнопке *Рассчитать* (рис. 1.19).

Ознакомиться с вариантами зашифрованного текста для разных значений сдвига при шифровании (например, строка для варианта ROT2 получена при сдвиге на 2 позиции в алфавите).

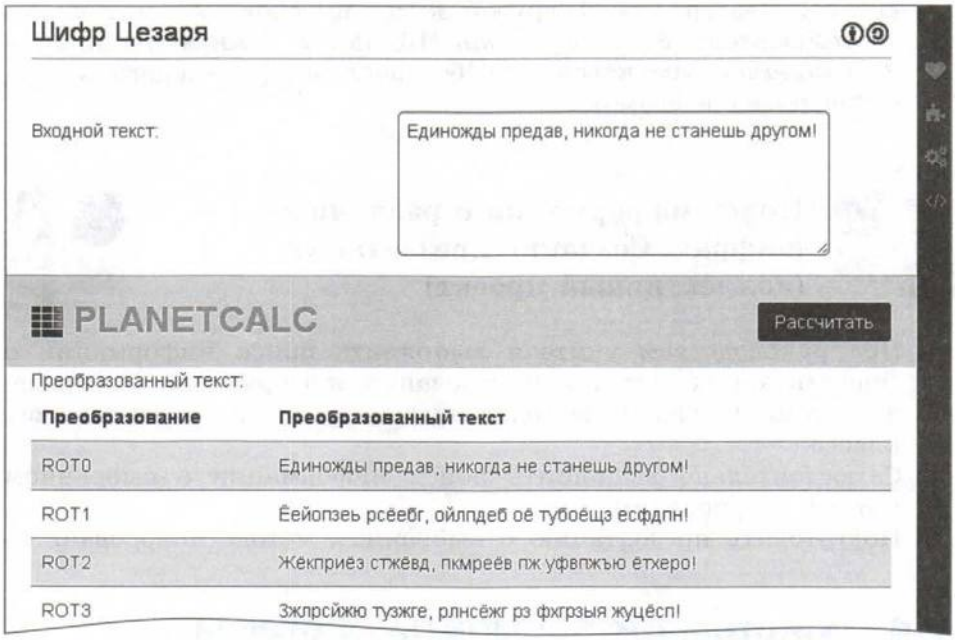


Рис. 1.19

Работа с онлайновым кодировщиком шифра Виженера

Выполним шифрование и дешифрование информации при помощи онлайн-кодировщика шифра Виженера.

1. Запустить браузер в операционной системе Windows (например, Mozilla Firefox) или в операционной системе Linux и в поле *Адрес*: ввести **http://planetcalc.ru/2468** (онлайн-калькулятор — кодировщик шифра Виженера).
2. Ознакомиться с описанием шифра Виженера и правилами работы с онлайн-калькулятором.
3. Выполнить шифрование информации: ввести желаемую строку текста в верхнем поле онлайн-калькулятора, ввести желаемый ключ шифрования — в нижнем поле калькулятора, выбрать переключатель *Зашифровать* и щёлкнуть мышью на кнопке *Рассчитать*.
4. Выполнить дешифрование текста при помощи онлайн-калькулятора. Для этого скопировать полученную зашифрованную строку в буфер обмена, вставить её в верхнее поле кальку-

лятора. Ввести любой другой ключ шифрования и выбрать переключатель *Расшифровать*. Щёлкнув мышью на кнопке *Рассчитать*, убедиться, что без знания правильного ключа дешифровка невозможна.

Поиск информации о различных шифрах. Создание презентации (коллективный проект)

1. Под руководством учителя выполнить поиск информации о существующих методах шифрования информации. Распределить темы (названия методов шифрования) между учащимися класса.
2. Самостоятельно выполнить поиск информации о выбранном методе шифрования.
3. Подготовить презентацию о выбранном методе шифрования.

1.4. Система и элементы системы

Система — это целостная совокупность взаимосвязанных элементов (компонентов), взаимодействующих друг с другом и с окружающей средой и имеющих общую цель функционирования.

В этом определении указаны основные отличия системы от некоего множества («кучи») элементов, просто собранных в одном и том же месте:

- между элементами системы имеются существенные взаимосвязи, обусловленные определённым назначением каждого элемента в составе системы как единого целого;
- система имеет определённую структуру, т. е. её элементы взаимосвязаны согласно определённой схеме, обеспечивающей функционирование системы в целом;
- указанная совокупность взаимосвязанных элементов создана с определённой целью, выполнение которой обеспечивается функционированием системы.

Кроме того, объединение элементов в систему позволяет получить новые свойства, которыми не обладает ни один её элемент в отдельности и которые проявляются только при взаимодействии этих элементов. Данный факт называют **системным эффектом**.

Примером системы является электрический фонарик, собранный из батарейки, лампочки, выключателя и электрических проводов:

- все элементы взаимосвязаны в соответствии с их назначением;
- взаимосвязь указанных элементов выполнена в соответствии с определённой электрической схемой;
- имеется определённая цель, которой подчинено существование данной системы (включение/выключение освещения);
- вся система в целом (фонарик) обладает свойством создавать свет, которым не обладают её элементы в отдельности (например, лампочка не может излучать свет сама по себе, без источника электропитания).

Элемент системы в свою очередь тоже можно рассматривать как систему более низкого уровня. В этом случае подобный компонент системы называют её **подсистемой**. Точно так же любая система может быть элементом (подсистемой) какой-либо более сложной системы. Например, электрическая лампочка (элемент системы «электрический фонарик») является системой, состоящей из элементов «цоколь», «колба», «нить накаливания» и т. д., а сам фонарик может быть одним из компонентов более сложной системы — велосипеда (использован в качестве фары).

Существует достаточно много различных способов классификации систем. Их можно классифицировать по сферам реализации (социальные, биологические, механические, экономические, физические и т. д.), по степени сложности (в зависимости от количества составляющих системху элементов), по происхождению (естественные — природные и искусственные — созданные человеком) и т. д. В данном учебнике вопросы классификации систем подробно не рассматриваются.

1.4.1. Состояние и взаимодействие компонентов системы

Для любой системы можно определить её **состояние** — как бы мгновенный «фотоснимок» состояния всех её элементов, определяющий свойства системы в этот момент. При этом можно отдельно определить:

- внутреннее состояние (как множество значений свойств её элементов и взаимосвязей между ними),
- состояние входов системы (фактически — значения параметров окружающей среды, которые влияют на систему),
- состояние выходов системы (значения параметров элементов системы, влияющих на окружающую среду).

Например, для электрического фонарика можно определить его текущее состояние как совокупность внутреннего состояния его элементов (скажем, насколько разряжена его батарейка), состояния входа — нажата или не нажата кнопка выключателя и состояния выхода — горит или не горит лампочка.

Если система может с течением времени изменять свое состояние, то она (в отличие от неизменной — **статической**) называется **динамической**, а процесс изменения её состояний, определяемый взаимосвязями её элементов и определяющий функционирование системы, называют **поведением** системы.

1.4.2. Информационное взаимодействие в системе и вне её. Управление. Обратная связь

Система предполагает кроме физического (механического, электрического и т. п.) также информационное взаимодействие её компонентов как друг с другом, так и с внешней средой. При этом система может быть как объектом анализа, когда рассмотрению подлежит именно внутреннее взаимодействие её компонентов, так и «чёрным ящиком», когда всё внимание конструктора или исследователя сосредоточено только на взаимодействии системы в целом с окружающей средой. В последнем случае ключевыми характеристиками системы являются состояния её входов и выходов и поведение системы, выражающееся в изменении этих состояний.

Как уже было сказано, любая система предполагает наличие определённой цели, которой подчинено её существование и функционирование. Для достижения этой цели может быть необходимо внешнее управляющее воздействие на систему, вызывающее требуемое изменение её состояния или поведения.

Воздействие одной системы на другую, осуществляемое с определённой целью, называется **управлением**. Система, которая осуществляет такое воздействие, называется **управляющей системой**. Система, на которую направлено управляющее воздействие, называется **управляемой системой** или **объектом управления**.

Возможны различные стратегии управления системой.

Разомкнутое управление (рис. 1.20) — односторонне направленное: воздействие на управляемую систему осуществляется по заранее выбранному алгоритму без учёта происходящих измене-

ний состояния системы (т. е. предполагается, что система реагирует на управляющие воздействия однозначным, заранее предсказуемым образом).

Рис. 1.20

Замкнутое управление — воздействие на управляемую систему с контролем её состояния и поведения с возможной коррекцией управляющего воздействия. Подобный контроль состояния управляемой системы называют **обратной связью** (рис. 1.21).

Рис. 1.21

Положительная обратная связь организована так, что любые происходящие в системе отклонения от некоторого исходного состояния приводят к ещё большему её отклонению от этого состояния. Обычно такая система является неустойчивой.

Отрицательная обратная связь, наоборот, приводит к уменьшению отклонения системы от исходного состояния. Чем больше система отклоняется от исходного состояния, тем управляющее воздействие сильнее стремится возвратить систему в исходное состояние. Системы с отрицательной обратной связью являются наиболее устойчивыми.

Примером системы с отрицательной обратной связью является обычный холодильник. Если температура внутри него по каким-то причинам повышается, срабатывает температурный датчик, включает компрессор и начинается дополнительное охлаждение.

Когда же температура окажется ниже заданной, то температурный датчик отключает компрессор, и температура в холодильнике снова повышается до нормативной.

Вопросы и задания

1. Приведите примеры систем из различных областей знаний. Обоснуйте свой ответ.
2. В чём различие систем с замкнутым и разомкнутым управлением?

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

- Единицы измерения информации
- Информация и информационные процессы
- Информация, информационные процессы в обществе, природе и технике

Глава 2
ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

При изучении данной главы рекомендуется установить следующее программное обеспечение для операционных систем Windows и Linux:

- браузер Mozilla FireFox;
- калькулятор NumLock Calculator;
- текстовый редактор OpenOffice Writer;
- электронные таблицы OpenOffice Calc;
- векторный графический редактор OpenOffice Draw;
- средство разработки презентаций OpenOffice Impress;
- растровый графический редактор GIMP;
- систему компьютерного черчения Компас;
- систему онлайновых словарей и переводчиков ABBYY;
- онлайновый переводчик ПРОМТ;

- электронные таблицы Microsoft Excel;

- программу разработки презентаций Microsoft PowerPoint;

- растровый графический редактор Microsoft Paint;

- систему оптического распознавания документов ABBYY FineReader;

- стандартную программу Microsoft Звукозапись;

- редактор звуковых файлов Wave Editor.

2.1. Кодирование и обработка текстовой информации

2.1.1. Кодирование текстовой информации

Информация, выраженная с помощью естественных и формальных языков в письменной форме, обычно называется текстовой информацией. Начиная с конца 1960-х годов, компьютеры всё больше стали использоваться для обработки текстовой информации.

Кодирование и декодирование текстовой информации. Для кодирования прописных и строчных букв русского и латинского алфавитов, цифр и ряда специальных знаков (знаки арифметических операций, знаки препинания и пр.) достаточно использовать 256 различных символов.

Кодирование для компьютера заключается в том, что каждому символу ставится в соответствие уникальный десятичный код от 0 до 255 или соответствующий ему двоичный код от 00000000 до 11111111. Получается, что для кодирования одного из 256 символов достаточно 8 двоичных цифр (бит), т. е. 1 байта.

Этот же результат можно получить по формуле, связывающей количество сообщений (здесь — знаков в алфавите знаковой

системы) N и количество информации i, необходимой, чтобы закодировать каждый из 256 знаков:

$$N = 2^i = 256 = 2^i = 2^8 = 2^i = i = 8 \text{ бит} = 1 \text{ байт}.$$

Таким образом, человек различает символы по их начертанию, а компьютер — по их коду.

При вводе в компьютер текстовой информации происходит её двоичное кодирование, изображение символа преобразуется в его двоичный код. Пользователь нажимает на клавиатуре клавишу с символом, и в компьютер поступает определённая последовательность из восьми электрических импульсов (двоичный код символа). Код символа хранится в оперативной памяти компьютера, где занимает одну байтовую ячейку.

В процессе вывода символа на экран компьютера производится обратный процесс — декодирование, т. е. преобразование кода символа в его изображение.

Кодировки русского алфавита. Важно, что присваивание символу конкретного кода — это вопрос соглашения, которое фиксируется в кодовой таблице. Первые 33 кода (с 0 по 32) этой таблицы соответствуют не символам, а операциям (перевод строки, ввод пробела и т. д.).

Коды с 33 по 127 являются интернациональными и соответствуют символам латинского алфавита, цифрам, знакам арифметических операций и знакам препинания.

Коды с 128 по 255 являются национальными, т. е. в национальных кодировках одному и тому же коду соответствуют различные символы. Для кодирования текстов на русском языке (букв кириллицы) применяются следующие кодовые страницы:

- CP1251, или Windows-1251 — в системах Windows;
- семейство кодовых страниц KOI8 — основная русская кодировка в Unix-совместимых ОС и в почтовых клиентах;
- CP866 (IBM code page 866), или Альтернативная кодировка — в системах DOS, а также в ней записываются имена файлов в системе FAT;
- MacCyrillic — на компьютерах Macintosh;
- ISO 8859-5 — восьмибитовая таблица ASCII.

Поэтому тексты, созданные в одной кодировке, не будут правильно отображаться в другой.

В настоящее время широкое распространение получил международный стандарт **Unicode**, который поддерживает одно-, двух- и четырёхбайтовые кодировки UTF-8, UTF-16 и UTF-32 соответственно. С помощью UTF-16 можно закодировать не 256 символов, а $N = 2^{16} = 65\,536$ различных символов. Формы записи в UTF-32

позволяют кодировать до 231 (2 147 483 648) кодовых позиций. Такого количества символов достаточно, чтобы закодировать не только русский и латинский алфавиты, цифры, знаки и математические символы, но и греческий, арабский, иврит и другие алфавиты.

Вопросы и задания

Как вы думаете, почему для кодирования текстовой информации в компьютере перешли от однобайтовых кодировок к двухбайтовой?

Практическая работа 2.1

Кодировки русских букв

Задание. Используя заготовки из электронного приложения, выяснить с помощью браузера, в каких кодовых страницах созданы веб-страницы.

Варианты выполнения работы:

- использование веб-страниц с разным содержанием;
- просмотр веб-страниц в различных браузерах (Mozilla FireFox, Chrome, Internet Explorer или др.).

Просмотр пяти веб-страниц в различных кодировках в браузере

Электронное приложение к главе 2.

В операционной системе Windows или Linux последовательно откроем в браузере пять веб-страниц из электронного приложения и выясним их кодировку.

1. Используя главное меню браузера Mozilla FireFox (появляется при нажатии клавиши F10), для каждой страницы определить нужную кодировку командой [*Вид—Кодировка текста*] (рис. 2.1).

Обратите внимание, что в Mozilla FireFox кодировки обозначены:

- CP1251 — Кириллица (Windows);
- CP866 — Кириллица (DOS);
- KOI8 — Кириллица (КОИ8);
- ISO 8859-5 — Кириллица (ISO);
- UTF-8 — Юникод.

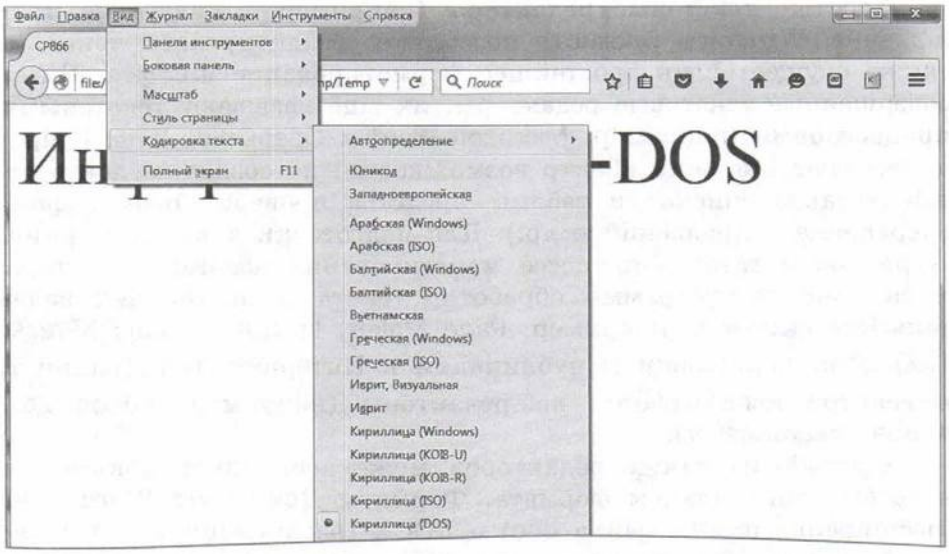


Рис. 2.1

2. Осуществить просмотр веб-страниц (рис 2.2).

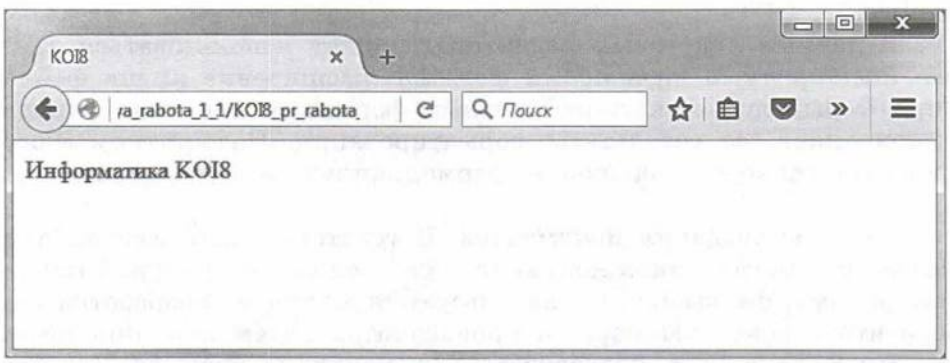


Рис. 2.2

2.1.2. Создание и редактирование документов в текстовых редакторах

Текстовые редакторы. Для обработки текстовой информации на компьютере используются текстовые редакторы, которые позволяют создавать, редактировать, форматировать, сохранять и распечатывать документы.

Простые текстовые редакторы (например, стандартное приложение Windows Блокнот) позволяют редактировать текст, а также осуществлять простейшее форматирование шрифта. Более совершенные текстовые редакторы, их ещё называют **текстовыми процессорами** (например, Microsoft Word и OpenOffice Writer), предоставляют широкий спектр возможностей по созданию документов (вставка списков и таблиц, средства проверки орфографии, сохранение исправлений и др.). Для подготовки к изданию книг, журналов и газет в процессе макетирования издания используются мощные программы обработки текста — **настольные издательские системы** (например, Page Maker, InDesign, Quark XPress, TEX). Для подготовки к публикации в Интернете веб-страниц и веб-сайтов используются **веб-редакторы** (например, Konpo Zer, Adobe Dreamweaver).

Каждый из таких редакторов может сохранять документы в своём оригинальном формате. Форматы *Документ Word* DOC (расширение имени файла doc) и DOCX (расширение docx) являются оригинальными форматами текстового редактора Microsoft Word, в котором полностью сохраняется всё форматирование.

В интегрированном офисном приложении OpenOffice используется открытый формат документов для офисных приложений ODF (*OpenDocument Format*), в том числе для текстовых документов — ODT.

В данных текстовых редакторах может использоваться также *Расширенный текстовый формат*, расширение имени файла rtf, который является универсальным форматом текстовых файлов и сохраняет все результаты форматирования. Недостатком этого формата является большой информационный объём файлов.

Способы создания документов. В текстовых процессорах для создания многих типов документов со сложной структурой (письма, резюме, факсы и т. д.) используются **мастеры**. Разработка документа с помощью мастера производится путём внесения необходимых данных в последовательно появляющиеся диалоговые окна.

Создание документов можно производить с помощью **шаблонов**, т. е. пустых заготовок документов определённого назначения. Шаблон задаёт структуру документа, которую пользователь заполняет определённым содержанием. Текстовые процессоры имеют обширные библиотеки шаблонов для создания документов различного назначения (визитная карточка, реферат и др.).

Выбрать требуемый шаблон можно при создании нового документа. Например, в текстовом процессоре Microsoft Word

версии 2007/2010/2013 при выборе на вкладке *Файл* пункта *Создать* предлагается выбрать шаблон из имеющегося набора (рис. 2.3). В том числе можно «извлечь» шаблон из существующего документа, выбрав иконку *Из существующего документа*.

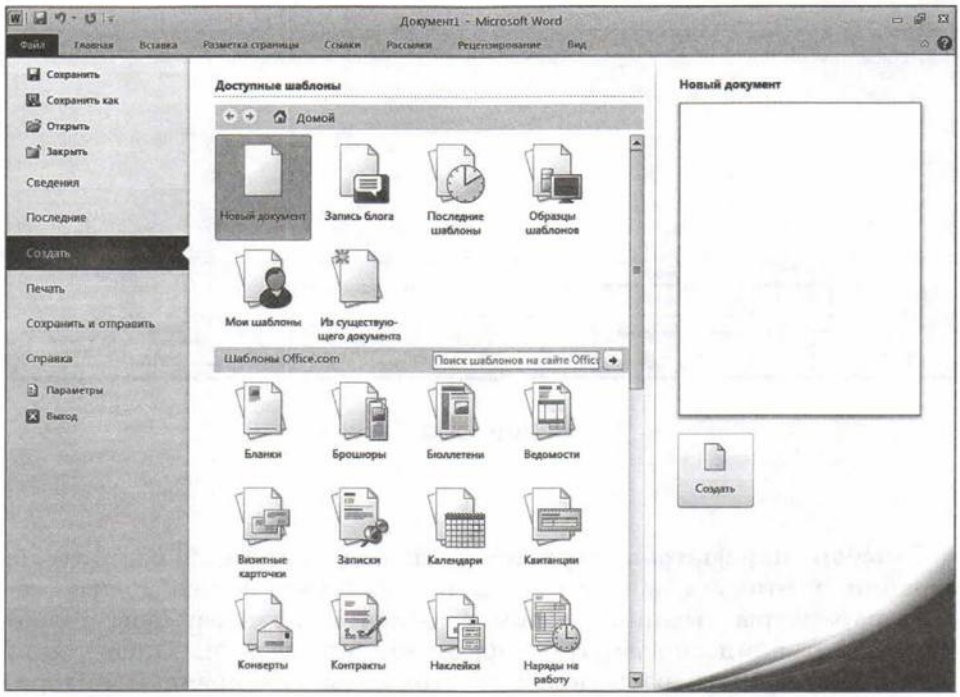


Рис. 2.3

В большинстве случаев для создания документов используется пустой шаблон *Новый документ*, который пользователь заполняет содержанием по своему усмотрению.

Созданный новый документ (либо изменённый документ, созданный по готовому шаблону) пользователь может сохранить как новый шаблон для последующего использования. Для этого достаточно выбрать на вкладке *Файл* пункт *Сохранить как* и в окне сохранения выбрать тип файла *Шаблон Word* (рис. 2.4).

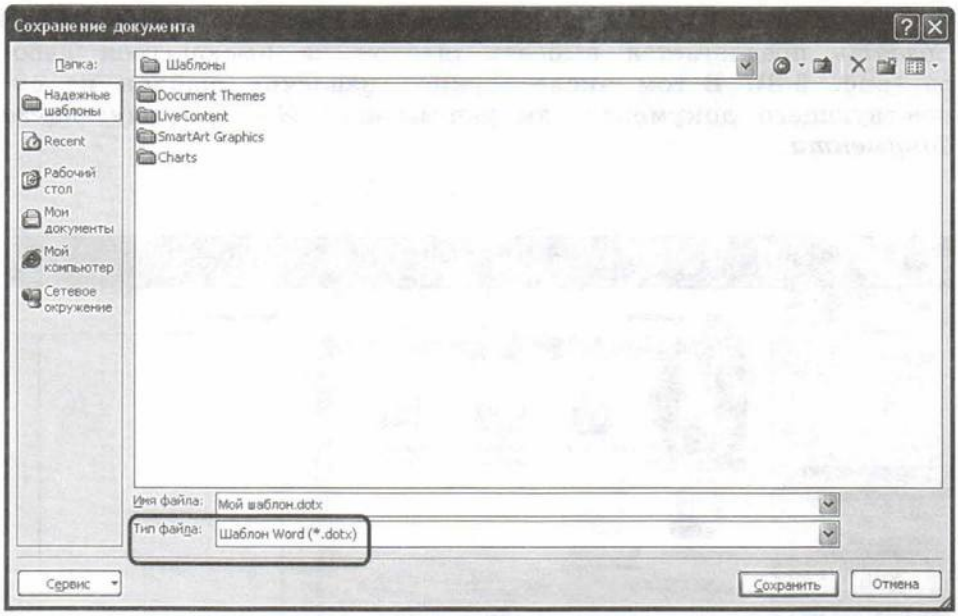


Рис. 2.4

Выбор параметров страницы. Любой документ состоит из страниц, поэтому в начале работы над документом необходимо задать параметры страницы: формат, ориентацию и размеры полей. Формат страниц документа определяет их размер. Ориентация позволяет выбрать расположение страницы на экране монитора. Существуют две возможные ориентации страницы — книжная и альбомная. На странице можно установить требуемые размеры полей (верхнего и нижнего, правого и левого), которые определяют расстояния от краёв страницы до границы текста.

Колонтитулы и номера страниц. Для вывода на каждой странице документа одинакового либо однотипного текста (например, имени автора, названия документа, номера страницы и др.) удобно использовать верхний или нижний колонтитул. Расстояния от края страницы до колонтитула можно изменять.

Страницы документа рекомендуется нумеровать, причём номера можно размещать в колонтитуле вверху или внизу страницы по центру, справа или слева.

Сноски. В конце страницы или документа можно разместить сноски, которые вносят необходимые разъяснения в текст документа. В тексте документа номера сносок оформляются в виде верхних индексов.

Ввод текста. Для представления текстов может использоваться 256 или 65 536 символов, однако ряд символов невозможно ввести с клавиатуры компьютера. Для ввода некоторых знаков математических операций, букв греческого алфавита, денежных знаков и многих других символов используются таблицы символов.

Вставка изображений, формул и других объектов в документ. Большинство современных документов содержат не только текст, но и другие объекты (изображения, формулы, таблицы, диаграммы и т. д.). Текстовые редакторы позволяют вставлять в документ изображения, созданные в графических редакторах, таблицы и диаграммы, созданные в электронных таблицах, и даже звуковые и видеофайлы, созданные в соответствующих приложениях.

Копирование, перемещение и удаление фрагментов документа. Редактирование документа производится путём копирования, перемещения или удаления выделенных символов или фрагментов документа. Копирование позволяет размножить выделенный фрагмент документа, т. е. вставить его копии в указанные места документа. Перемещение позволяет вставить копии выделенного фрагмента документа в указанные места документа, но удаляет сам выделенный фрагмент. Удаление позволяет удалить выделенный фрагмент.

Поиск и замена. В процессе работы над документом иногда бывает необходимо заменить одно многократно использованное слово на другое. Если делать это вручную, то процесс замены отнимет много времени и сил. В большинстве текстовых редакторов существует операция *Найти и заменить*, которая обеспечивает автоматический поиск и замену слов во всем документе.

Проверка правописания. В процессе создания документа могут быть допущены ошибки в написании слов и в построении предложений. Ошибки можно исправить, если запустить встроенную во многие текстовые редакторы систему проверки правописания, которая содержит орфографические словари и грамматические правила для нескольких языков (это позволяет исправлять ошибки в многоязычных документах).

Автозамена частых опечаток. В процессе ввода текста иногда допускаются опечатки (например, в начале слова случайно вводятся ДВе прописные буквы). В этом случае срабатывает функция *Автозамена*, которая автоматически исправляет такие опечатки.

Сохранение исправлений. В процессе работы над документом могут участвовать несколько пользователей. Исправления, вносимые каждым из них, запоминаются и могут быть просмотрены и распечатаны (вставленные фрагменты обычно отображаются подчёркнутым шрифтом синего цвета, а удалённые фрагменты текста — зачёркнутым шрифтом красного цвета).

Сохранение документов. В процессе сохранения документа необходимо в иерархической файловой системе компьютера выбрать диск и папку, в которой необходимо сохранить файл документа.

Кроме того, необходимо выбрать формат файла, который определяет способ хранения текста в файле. Существуют **универсальные форматы** текстовых файлов (например, TXT, RTF и HTML), которые могут быть прочитаны большинством текстовых редакторов, и **оригинальные форматы** (например, ODT), которые используются только определённым текстовым редактором (OpenOffice Writer).

Печать документов. Перед выводом документа на печать полезно выполнить предварительный просмотр документа. Это позволяет увидеть, как будет выглядеть документ, напечатанный на бумаге с использованием подключённого к компьютеру принтера.

При выводе документа на печать необходимо установить параметры печати: задать номера выводимых на печать страниц, количество копий документа и др.

Кроме того, целесообразно проверить установки самого принтера: ориентацию бумаги, качество бумаги, качество печати и др.

Вопросы и задания

1. Какие параметры страницы документа необходимо установить?
2. В каком формате нужно сохранить файл, чтобы он мог быть прочитан в других приложениях с сохранением форматирования? Без сохранения форматирования?
3. Перечислите операции редактирования документа.
4. Какие элементы кроме текста можно вставлять в текстовый документ?

2.1.3. Форматирование документов в текстовых редакторах

Для представления содержания документа в более понятной и выразительной форме применяется форматирование. Символы являются основными объектами, из которых состоит текстовый документ, поэтому прежде всего необходимо правильно установить основные параметры, определяющие их внешний вид: шрифт, размер, начертание и цвет.

Форматирование символов. Шрифт — это полный набор символов (букв, цифр, знаков пунктуации, математических знаков, а также специальных символов) определённого рисунка. Для каждого исторического периода и каждой страны характерен свой шрифт. Шрифты имеют названия, например: Times New Roman, Arial, Courier New и др.

По способу представления в компьютере различаются шрифты растровые и векторные. Для представления растровых шрифтов используются методы растровой графики, когда символы шрифта представляют собой группы пикселей. Растровые шрифты допускают масштабирование только с определёнными коэффициентами (например, MS Sans Serif 8, 10, 12 и т. д.). В векторных шрифтах символы описываются математическими формулами и допускают произвольное масштабирование.

Обычно различные символы шрифта имеют и различную ширину, например буква Ш шире, чем буква А. Однако имеются и моноширинные шрифты, в которых ширина всех символов одинакова. Примером такого шрифта является шрифт Courier New.

Шрифты также разделяют на две большие группы: шрифты с засечками (например, Times New Roman) и рубленые (например, Arial). Считается, что шрифты с засечками легче воспринимаются глазом, поэтому в большинстве печатных текстов используются именно они. Рубленые шрифты применяют обычно для заголовков, выделений в тексте и подписей к рисункам.

Единицей измерения **размера шрифта** является **пункт** (1 пт = 0,376 мм). Размеры шрифтов можно изменять в больших пределах (обычно от 1 до 1638 пунктов), причём в большинстве редакторов по умолчанию используется шрифт размером 10 пт.

Кроме обычного **начертания символов** может применяться **полужирное**, *курсивное*, ***полужирное курсивное*** и подчёркнутое.

Можно установить дополнительные параметры форматирования символов: подчёркивание символов различными типами линий, верхний индекс, нижний индекс, зачёркивание, изменение

расстояния между символами (разреженный, уплотнённый шрифт) и др.

Если планируется многоцветная печать документа, то для разных групп символов можно задать разные цвета, выбранные из предлагаемой текстовым редактором палитры.

Рис. 2.5

Буквица (капитель). Буквица широко применялась в старинных книгах в начале абзаца или главы (рис. 2.5). Можно установить врезанную в текст буквицу (она называется капителью) высотой в любое количество строк.

Форматирование абзацев. В компьютерных текстовых документах **абзацем** называется часть текста между двумя управляющими символами конца абзаца. Ввод конца абзаца обеспечивается нажатием клавиши *Enter* и отображается символом ¶, если включён режим отображения непечатаемых символов.

Абзац может состоять из любого набора символов, рисунков и объектов других приложений. Форматирование абзацев позволяет подготовить правильно и красиво оформленный документ.

Выравнивание абзаца отражает расположение текста относительно границ полей страницы. Чаще всего абзац начинается отступом первой строки. Весь абзац целиком может иметь отступы слева и справа, которые отмеряются от границ полей страницы.

Расстояние между строками документа можно изменять, задавая различные значения междустрочных интервалов. Для визуального отделения абзацев друг от друга можно устанавливать увеличенные интервалы до и после абзаца.

Нумерованные и маркированные списки. Списки являются удобным вариантом форматирования абзацев и применяются для размещения в документе различных перечней.

В нумерованных списках элементы списка последовательно обозначаются с помощью чисел (арабских или римских) и букв (русского или латинского алфавита). При создании, удалении или перемещении элементов нумерованного списка автоматически меняется вся нумерация. Пользователь может установить свою систему нумерации, например начать список с любого номера, пропустить номер и т. д.

В маркированных списках элементы списка обозначаются с помощью маркеров (специальных значков): ●, ■, ⇨ и др. Пользователь может выбрать тип маркера, изменить его размер и цвет, а также выбрать в качестве маркера любой символ из таблицы символов.

В многоуровневых списках в пункты списка более высокого уровня вставляются списки более низкого уровня (вложенные списки). Вложенные списки могут совпадать по типу с основным списком, но могут и отличаться от него. Многоуровневые списки можно использовать для отображения иерархических перечней (например, иерархической файловой структуры).

Стили форматирования. При создании многостраничных документов удобнее для каждого типа абзацев использовать определённый стиль форматирования (например, один стиль для текста параграфа, а другой — для вопросов к параграфу). Каждому стилю форматирования присваивается название и устанавливаются все необходимые параметры форматирования шрифта, абзаца или списка.

После этого для изменения параметров форматирования абзацев одного типа достаточно изменить параметры соответствующего стиля форматирования. Все абзацы данного стиля форматирования автоматически получат во всём документе новые параметры форматирования.

Оглавление документа. В процессе создания документа в нём создаются заголовки, для которых используются различные стили форматирования.

После создания объёмного документа целесообразно вставить в документ оглавление, которое позволит читателю лучше ориентироваться в его содержании. Оглавление представляет собой список заголовков, содержащихся в документе, с указанием страниц.

Таблицы. Таблицы состоят из строк и столбцов, на пересечении которых находятся ячейки. В ячейках таблиц могут быть размещены данные различных типов (текст, числа или изображения). При размещении в таблице чисел можно производить над ними вычисления по формулам.

В документ можно вставить пустую таблицу, указав необходимое количество строк и столбцов, а также их ширину и высоту. В таблицу можно также преобразовать уже имеющийся текст, при этом требуется указать разделитель текста (например, символ конца абзаца), который позволит текстовому редактору автоматически распределить выделенный текст по ячейкам создаваемой таблицы.

Можно подобрать подходящий внешний вид таблицы, изменив тип, ширину и цвет границ ячеек, а также цвет фона ячеек. Изменение внешнего вида таблицы можно провести автоматически, используя готовые форматы, или настроить вручную.

Работа с иллюстрациями. Для форматирования иллюстраций используются инструменты: *Настройка размера*, *Положение* (*за текстом*, *перед текстом*), *Обтекание текстом* и т. д. Инструмент *Группировка* позволяет работать с несколькими иллюстративными объектами в тексте одновременно. Сгруппированные объекты воспринимаются в текстовом редакторе как один, это означает, что с ним можно осуществлять те же действия, что описаны выше. Список инструментов для *Работы с рисунками* в Microsoft Word находится в меню *Формат* (рис. 2.6).

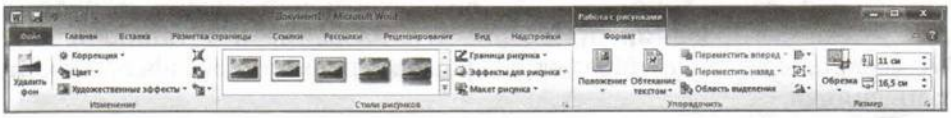


Рис. 2.6

Вопросы и задания

1. Какие существуют параметры форматирования символов?
2. Какие существуют параметры форматирования абзацев?
3. Чем различаются между собой отступ первой строки абзаца и отступ абзаца?
4. Может ли многоуровневый список включать как нумерованные, так и маркированные списки?
5. Данные каких типов могут храниться в ячейках таблицы?

2.1.4 Деловая переписка

В повседневной жизни мы стараемся соблюдать общественный этикет (нормы и правила поведения в обществе). Его частью является этикет делового общения, требующий в том числе грамотного составления документов, осуществления переписки с партнерами и коллегами. В настоящее время в обществе при ведении деловой переписки укоренились специальные устойчивые речевые формы.

Деловая переписка — это набор правил и средств, которые нужно знать, чтобы грамотно составлять любые документы.

Рекомендации при составлении делового письма

1. Для деловой переписки характерен нейтральный тон, не должно быть излишней эмоциональности в изложении.
2. Необходимо применять логические средства оценки ситуации.
3. Высказывания, суждения в документе необходимо подкреплять существенными достоверными фактами.
4. В официальных текстах при обращении к человеку используются формы слова «уважаемый».
5. При обращении к лицам одной профессии используется выражение «уважаемый коллега», если речь идет о группе людей, то «уважаемые коллеги».
6. В нейтральном обращении используется слово «коллеги».
7. При написании текста личностного характера к человеку предпочтительно обращаться по имени и отчеству.
8. При необходимости написания документа, содержащего решение по какому либо вопросу, стоит обратить внимание на обоснование решения.

Примером деловой переписки является благодарственное письмо.

БЛАГОДАРСТВЕННОЕ ПИСЬМО

Уважаемый Пётр Александрович!

Издательство «БИНОМ. Лаборатория знаний» выражает слова искренней благодарности за Ваш высокий профессионализм, компетентность, педагогический талант, за взаимное сотрудничество, распространение передового педагогического опыта.

Желаем Вам неиссякаемой творческой энергии, свершения всех начинаний и планов, успехов, крепкого здоровья!

Методическая служба
издательства «БИНОМ. Лаборатория знаний»

Вопросы и задания

1. Почему о деловой переписке говорят, что это баланс вежливости и взаимных интересов?
2. Почему этикет деловой переписки предполагает, что на письма нужно давать ответы?
3. Создайте средствами текстового редактора деловое письмо, придерживаясь принятых правил.

2.1.5. Библиографическое описание. Стандарты, правила оформления

В современном мире невозможно представить ни одно издательство, книготорговую организацию или библиотеку, не имеющих своих каталогов, в которых по определённым правилам структурирована информация. Для организации обмена информацией созданы системы, обеспечивающие извлечение, интеграцию и обработку знаний из различных информационных источников. Для обеспечения свободного поиска и обнаружения знаний отрасль информационной деятельности «Библиография» располагает методами структурирования и упорядочения информации (название произошло от древнегреческого βιβλιογραφία, βιβλίον (biblion) — книга, γράφω (grapho) — пишу).

Библиография — это отрасль информационной деятельности, обеспечивающая подготовку, распространение и использование библиографической информации.

Понятие «библиографическая информация» разработал профессор Олег Павлович Коршунов (1926–2013) — российский учёный, доктор педагогических наук, работающий в Московском государственном университете культуры и искусств на кафедре общей библиографии и на кафедре электронных библиотек, информационных технологий и систем.

Библиографическая информация — это по определённым правилам организованная информация о документах, содействующая реализации соответствий между документами и их потребителями.

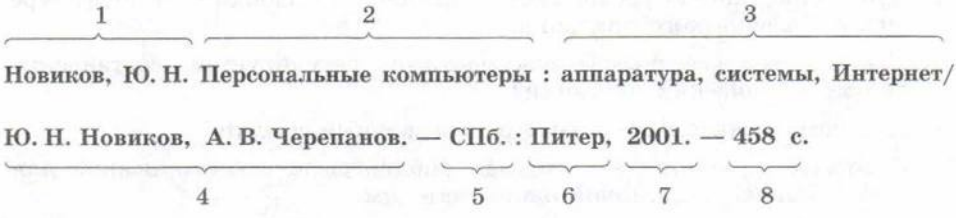
Результат исследований в различных научных областях фиксируется в виде документа, структурированного по определённым

правилам. Одним из разделов этого документа является список используемой литературы, или библиографический список, в котором перечисляют источники (книги, журналы, отдельные статьи, материалы, размещенные в Интернете), регламентированные правилами библиографического описания.

Библиографическое описание — совокупность норм и правил описания документов для составления библиографического списка использованных источников.

В нашей стране существует **система стандартов** по информации, библиотечному и издательскому делу. Правила оформления библиографических описаний регулируются государственным стандартом ГОСТ 7.1-2003 «Библиографическая запись. Библиографическое описание. Общие требования и правила составления», где устанавливаются «...общие требования и правила составления библиографического описания документа, его части или группы документов: набор областей и элементов библиографического описания, последовательность их расположения, наполнение и способ представления элементов, применение предписанной пунктуации и сокращений». Стандарты предоставляют возможность в тексте документа корректно указывать источник используемой информации.

Приведём пример библиографического описания для книги двух авторов.



Здесь 1 — фамилия автора, 2 — основное заглавие, 3 — сведения, относящиеся к заглавию, 4 — сведения об ответственности, 5 — место издания, 6 — издательство, 7 — объём и примечания.

Затекстовые ссылки и подстрочные ссылки

При написании реферата (статьи, научной работы и т. д.) в тексте работы после упоминания произведения (после цитаты из него), необходимо проставлять затекстовые ссылки или, при необходимости, подстрочные ссылки.

Затекстовые ссылки. В квадратных скобках проставляют номер библиографического описания, включённого в библиографический список, а в необходимых случаях, дополняют сведениями о страницах.

Пример затекстовой ссылки:

...«В капиталистическом обществе вождем и руководителем «общественного целого» является буржуазии. В переходном обществе — пролетариат» [3, С. 21].

Подстрочные ссылки размещаются внизу страницы, под строками основного текста.

Пример подстрочной ссылки:

...«В капиталистическом обществе вождем и руководителем «общественного целого» является буржуазии. В переходном обществе — пролетариат»[1].

Вопросы и задания

1. Подумайте, почему любой документ следует описывать по определённым правилам.
2. Что такое библиографическое описание? Расскажите о структуре библиографического описания.
3. Какими нормативными документами регулируется составление библиографических описаний?
4. Для чего нужны затекстовые и подстрочные ссылки?
5. Составьте и запишите в тетрадь библиографическое описание для любой книги, написанной одним автором.
6. Создайте средствами текстового процессора список источников и расставьте в тексте ссылки на них.
7. Подготовьте реферат по теме «Библиографическое описание интернет-источников».

---

[1] Капелюшников Р. М. Экономическая теория прав собственности / Р. М. Капелюшников. — М.: ИМЭМО, 1990. — С. 67.

Практическая работа 2.2

Создание и форматирование документа

Задание. Ввести и отформатировать текст по образцу:

Абзац с выравниванием по ширине, отступ слева 6 см, шрифт Times New Roman, размер 10 пт, начертание обычное, цвет символов синий.

Абзац с выравниванием по центру, отступ слева 0 см, шрифт Arial, размер 14 пт, начертание полужирное, цвет символов зелёный.

Абзац с выравниванием по левому краю, отступ первой строки 1 см, шрифт Courier New, размер 12 пт, начертание курсив, цвет символов красный.

Варианты выполнения работы:

- создание документов с разными параметрами форматирования шрифта и абзацев;
- форматирование документа в различных текстовых редакторах (Microsoft Word, OpenOffice Writer или др.).

Создание и форматирование документа в OpenOffice Writer

Электронное приложение к главе 2.

1. В операционной системе Windows запустить текстовый редактор OpenOffice Writer, предварительно установив его с веб-ресурса. В операционной системе Linux запустить OpenOffice Writer командой [*Офис—OpenOffice Writer (Текстовый процессор)*].

Для каждого абзаца (рис. 2.7) установим параметры форматирования символов и абзаца.

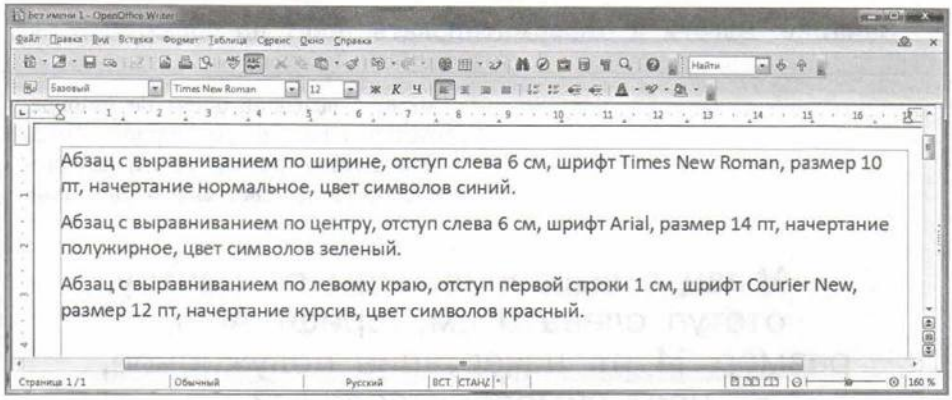


Рис. 2.7

2. Для форматирования символов выделить текст и ввести команду [*Формат—Символы...*], откроется диалоговое окно *Символы*.
На вкладке *Шрифт* установить параметры форматирования символов: гарнитуру, начертание и кегль (т. е. размер шрифта) — рис. 2.8.

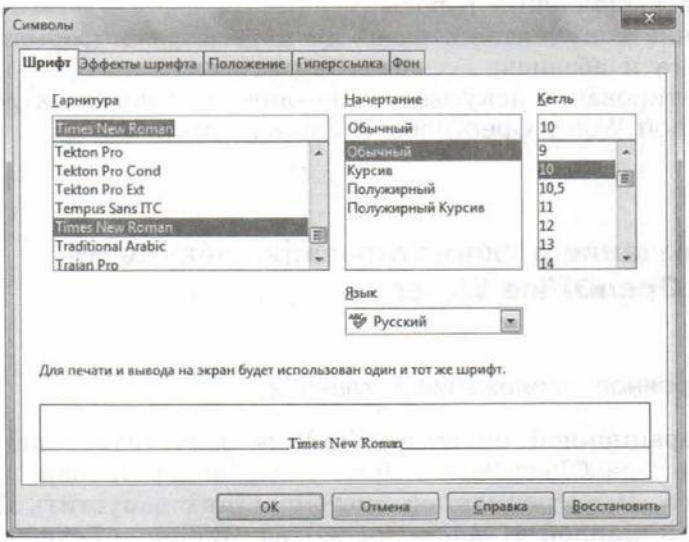


Рис. 2.8

3. На вкладке *Эффекты шрифта* с помощью раскрывающегося списка установить цвет шрифта (рис. 2.9).

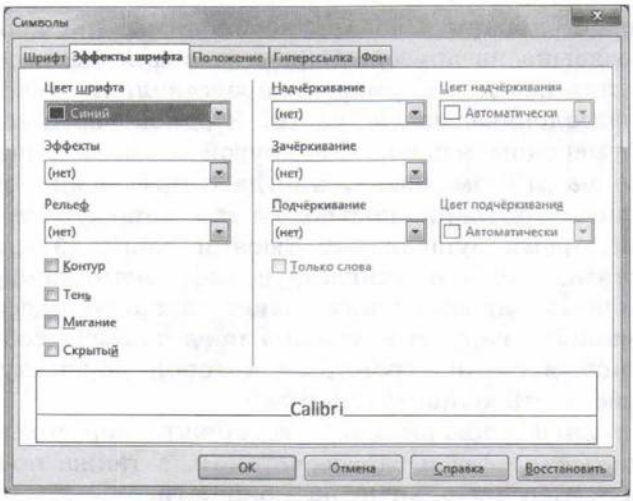


Рис. 2.9

4. Для форматирования абзаца выделить текст и ввести команду [*Формат—Абзац...*].

   В появившемся диалоговом окне *Абзац* на вкладке *Отступы и интервалы* установить отступы абзаца и красной строки (рис. 2.10).

   На вкладке *Выравнивание* установить тип выравнивания абзаца.

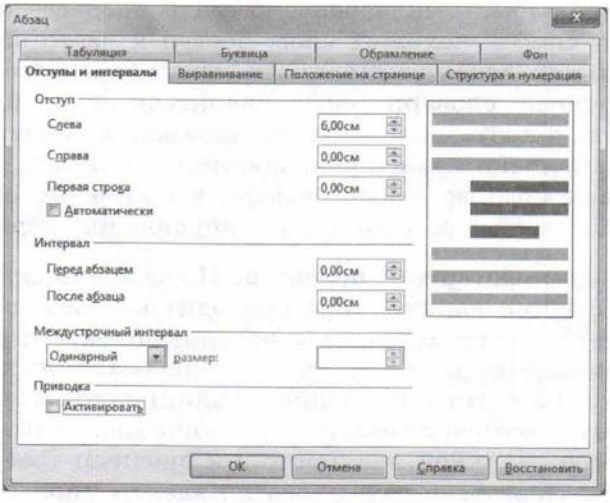


Рис. 2.10

2.1.6. Компьютерные словари и системы компьютерного перевода текстов

Компьютерные словари. Словари необходимы для перевода текстов с одного языка на другой. Первые словари были созданы около 5 тысяч лет назад в Шумере и представляли собой глиняные таблички, разделённые на две части. В одной части записывалось слово на шумерском языке, а в другой — аналогичное по значению слово на другом языке, иногда с краткими пояснениями.

Современные словари построены по такому же принципу. В настоящее время существуют тысячи словарей для перевода между сотнями языков (англо-русский, немецко-французский и т. д.), причём каждый из них может содержать десятки тысяч слов. В бумажном варианте словарь представляет собой толстую книгу объёмом в сотни страниц, в которой поиск нужного слова — процесс достаточно трудоёмкий.

Компьютерные словари могут содержать переводы на разные языки сотен тысяч слов и словосочетаний, а также предоставляют пользователю дополнительные возможности:

- существуют многоязычные компьютерные словари, позволяющие пользователю выбрать языки и направление перевода (например, англо-русский, испано-русский и т. д.);
- компьютерные словари могут кроме основного словаря общеупотребительных слов содержать десятки специализированных словарей по областям знаний (техника, медицина, информатика и др.);
- компьютерные словари обеспечивают быстрый поиск словарных статей: «быстрый набор», когда в процессе набора слова возникает список похожих слов; доступ к часто используемым словам по закладкам; возможность ввода словосочетаний и др.;
- компьютерные словари могут являться мультимедийными, т. е. предоставлять пользователю возможность прослушивания слов в исполнении дикторов, носителей языка;
- онлайновые компьютерные словари в Интернете обеспечивают выбор тематического словаря и направления перевода.

Системы компьютерного перевода. Происходящая в настоящее время глобализация нашего мира приводит к необходимости обмена документами между людьми и организациями, находящимися в разных странах мира и говорящими на различных языках.

В этих условиях использование традиционной технологии перевода вручную тормозит развитие межнациональных контактов. Перевод многостраничной документации вручную требует длительного времени. Перевод полученного по электронной почте письма или просматриваемой в браузере веб-страницы необходимо осуще-

ствить «здесь и сейчас», и нет возможности и времени пригласить переводчика.

Системы компьютерного перевода позволяют решить эти проблемы. Они способны переводить многостраничные документы с высокой скоростью (одна страница в секунду) и могут переводить веб-страницы «на лету», в режиме реального времени.

Системы компьютерного перевода осуществляют перевод текстов, основываясь на формальном «знании» языка: синтаксиса (правил построения предложений), правил словообразования и использовании словарей. Программа-переводчик сначала анализирует текст на одном языке, а затем конструирует этот текст на другом языке.

Онлайновые компьютерные переводчики в Интернете обеспечивают выбор тематического словаря и направления перевода. Они позволяют переводить любые тексты, набранные в окне перевода или скопированные из буфера обмена, целые веб-страницы, включая гиперссылки и с сохранением исходного форматирования, а также электронные письма.

Современные системы компьютерного перевода позволяют с приемлемым качеством переводить техническую документацию, деловую переписку и другие специализированные тексты. Но на эти системы нельзя полностью полагаться. Они допускают смысловые и стилистические ошибки и неприменимы, например, для перевода художественных произведений, так как неспособны адекватно переводить метафоры, аллегории и другие элементы художественного творчества человека.

Вопросы и задания

1. Какими преимуществами обладают компьютерные словари по сравнению с традиционными бумажными словарями?
2. Какие документы целесообразно переводить с помощью систем компьютерного перевода?

Практическая работа 2.3

Перевод с помощью онлайновых словаря и переводчика

Задание 1. В Интернете с помощью онлайнового компьютерного словаря перевести с русского языка на английский язык какое-либо одно слово, например «словарь».

Варианты выполнения работы:

- использование различных направлений перевода;
- использование различных слов для перевода.

Задание 2. В Интернете с помощью онлайнового компьютерного переводчика перевести с английского языка на русский язык

целое предложение, например: «The teacher's computer is placed on the table in the corner of the classroom».

Варианты выполнения работы:

- использование различных направлений перевода;
- использование различных текстов для перевода.

Перевод в Интернете с помощью онлайнового компьютерного словаря Lingvo

1. В операционной системе Windows или Linux запустить браузер и открыть в Интернете компьютерный словарь по адресу **http://www.lingvo.ru**.
2. Ввести русское слово (в данном случае «словарь») в текстовое поле (рис. 2.11).

   Выбрать из списка направление перевода (в данном случае *Русский<>Английский*).

   Щёлкнуть по кнопке *Перевести*.

   Появятся варианты перевода (в данном случае «dictionary; vocabulary; glossary; lexicon») — см. рис. 2.11.

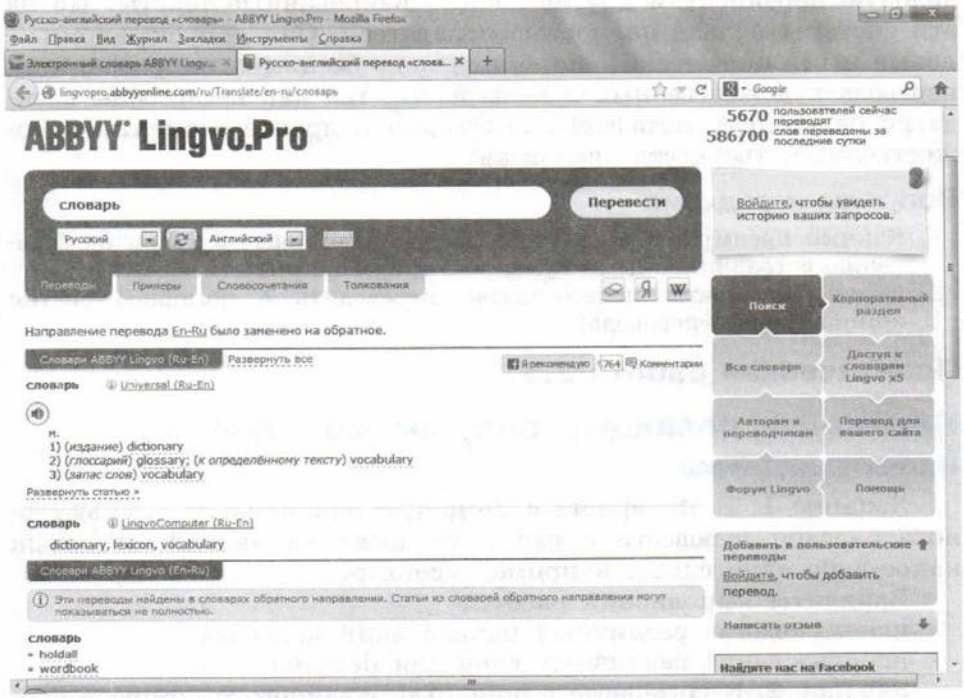


Рис. 2.11

Перевод в Интернете с помощью онлайнового компьютерного переводчика ПРОМТ

1. В операционной системе Windows или Linux запустить браузер и открыть в Интернете компьютерный переводчик по адресу **http://www.translate.ru**.
2. Ввести английское предложение в текстовое поле.
   Выбрать из списка направление перевода (в данном случае *Английский<>Русский*).
   Щелкнуть по кнопке *Перевести*.
   В текстовом поле появится перевод (в данном случае «Компьютер учителя положен на стол в углу классной комнаты») — рис. 2.12.

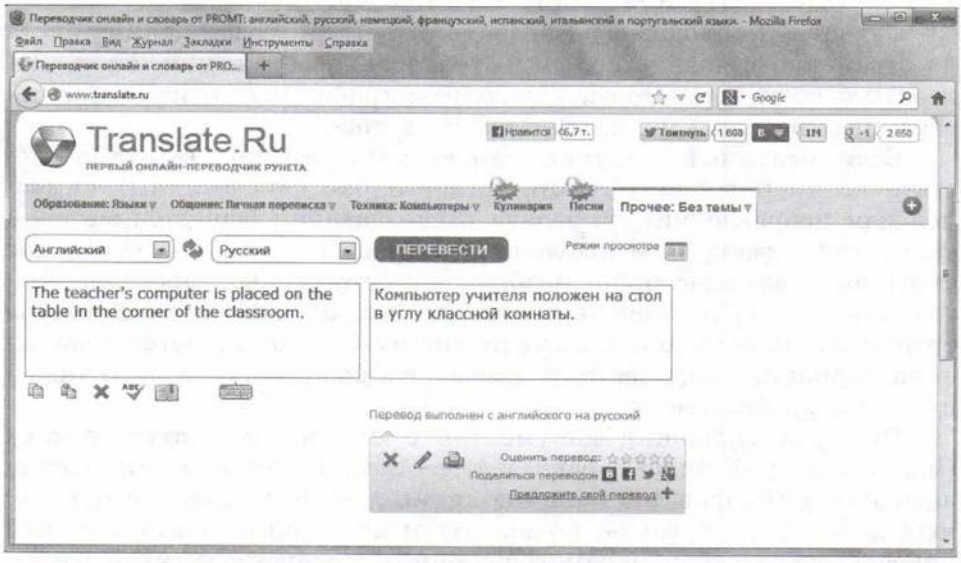


Рис. 2.12

Заметим, что перевод содержит ошибку. Правильный перевод: «Компьютер учителя находится на столе в углу классной комнаты».

2.1.7. Системы оптического распознавания документов

Системы оптического распознавания символов. При создании электронных библиотек и архивов путём перевода книг и доку-

ментов в цифровой компьютерный формат, при переходе предприятий от бумажного к электронному документообороту, при необходимости отредактировать полученный по факсу документ используются системы оптического распознавания символов.

С помощью сканера несложно получить изображение страницы текста в графическом файле. Однако для получения документа в формате текстового файла необходимо провести распознавание текста, т. е. преобразовать элементы графического изображения в последовательности текстовых символов.

Текст, преобразованный из графической формы в символьную (текстовую), можно далее обрабатывать любыми текстовыми редакторами. Системы оптического распознавания символов экспортируют результаты распознавания в популярные офисные приложения (Microsoft Office, OpenOffice и др.), причём распознанный текст можно сохранить в различных текстовых форматах: DOCX, DOC, ODT, RTF, TXT, HTML и др.

Сначала необходимо распознать структуру размещения текста на странице: выделить колонки, таблицы, изображения и т. д. Далее выделенные текстовые фрагменты графического изображения страницы необходимо преобразовать в текст.

Если исходный документ имеет типографское качество (достаточно крупный шрифт, отсутствие плохо напечатанных символов или исправлений), то задача распознавания решается методом сравнения с растровым шаблоном. Сначала растровое изображение страницы разделяется на изображения отдельных символов. Затем каждый из них последовательно накладывается на шаблоны символов, имеющихся в памяти системы, и выбирается шаблон с наименьшим количеством точек, не совпадающих с точками входного изображения.

При распознавании документов с низким качеством печати (машинописный текст, факс и т. д.) используется метод распознавания символов по наличию в них определённых структурных элементов (отрезков, колец, дуг и др.). Любой символ можно описать через набор параметров, определяющих взаимное расположение его элементов. Например, буква «Н» и буква «И» состоят из трёх отрезков, два из которых расположены параллельно друг другу, а третий соединяет эти отрезки. Различие между буквами — в величине углов, которые составляет третий отрезок с двумя другими. При распознавании структурным методом в искажённом символьном изображении выделяются характерные детали и сравниваются со структурными шаблонами символов. В результате выбирается тот символ, для которого совокупность всех структурных элементов и их расположение больше всего соответствуют распознаваемому символу.

Наиболее распространённые системы оптического распознавания символов используют как растровый, так и структурный метод распознавания. Кроме того, эти системы являются «самообучающимися» (для каждого конкретного документа они создают соответствующий набор шаблонов символов), поэтому скорость и качество распознавания многостраничного документа постепенно возрастают.

Системы оптического распознавания форм. При проведении Единого государственного экзамена, при заполнении налоговых деклараций и т. д. используются различного вида бланки с полями. Рукопечатные тексты (данные вводятся в поля печатными буквами от руки) распознаются с помощью систем оптического распознавания форм и вносятся в компьютерные базы данных.

Сложность состоит в том, что необходимо распознавать символы, написанные от руки, а они довольно сильно различаются у разных людей. Кроме того, система должна определить, к какому полю относится распознаваемый текст.

Оптическое распознавание документов. Интеллектуальные системы оптического распознавания позволяют быстро и точно переводить бумажные документы, цифровые фотографии документов и PDF-файлы в электронный вид. При распознавании они полностью сохраняют оформление документа: иллюстрации, картинки, списки, таблицы и т. д. Полученные результаты можно исправлять в текстовых редакторах, сохранять в разных форматах, отправлять по электронной почте и публиковать в Интернете.

Анализ и обработка документа целиком, а не постранично, позволяют распознать такие элементы его внутренней структуры, как верхние и нижние колонтитулы, сноски, гиперссылки, подписи к картинкам и диаграммам, стили, шрифты и т. д. Таким образом, система оптического распознавания точно распознаёт и максимально полно сохраняет исходное оформление любого документа.

Оптическое распознавание изображений. Системы оптического распознавания символов работают со всеми популярными моделями сканеров, но для распознавания необязательно оснащать компьютер сканером, так как современные системы позволяют распознавать фотографии документов, сделанные цифровой камерой. Существует множество случаев, когда для получения изображения удобнее использовать фотоаппарат, нежели сканер. Например, во время деловой встречи вне офиса, при распознавании вывесок или объявлений, в библиотеке, особенно при работе

с толстыми или старинными книгами. Не говоря уже о том, что цифровой фотоаппарат работает в несколько раз быстрее любого сканера.

Системы оптического распознавания символов работают с большим количеством графических файлов распространённых форматов: BMP, JPEG, TIFF, PNG и других. Для сканирования большого количества страниц в программах предусмотрен специальный режим, позволяющий работать как с автоподатчиком сканера, так и без него.

Системы оптического распознавания символов позволяют даже предварительно обработать изображения, чтобы повысить качество распознавания и упростить дальнейшую работу с документом. Программы могут очистить изображение от «мусора», устранить перекосы и искажение строк, инвертировать изображение, повернуть или зеркально отразить его, обрезать края или стереть часть изображения.

Системы распознавания рукописного текста. С появлением первого карманного компьютера в 1990 году начали создаваться системы распознавания рукописного текста. Такие системы преобразуют текст, написанный на экране карманного компьютера специальной ручкой (стилусом), в текстовый компьютерный документ.

Вопросы и задания

В чём состоят различия в технологиях распознавания документов типографского качества и с низким качеством печати?

Практическая работа 2.4

Сканирование бумажного и распознавание электронного текстового документа

Задание. Отсканировать и преобразовать в электронный текстовый документ страницу учебника.

Варианты выполнения работы:

- использование различных «бумажных» документов для сканирования;
- использование различных параметров сканирования;
- распознавание изображения с помощью сканера или электронного фотоаппарата.

Сканирование бумажного и распознавание текстового документа с использованием ABBYY FineReader

Электронное приложение к главе 2.

1. Положить раскрытый на нужной странице учебник на стекло сканера.
2. Запустить программу распознавания текста ABBYY FineReader.
3. В окне системы сканирования и оптического распознавания ввести команду [*Файл—Сканировать страницы...*].
   В появившемся окне *Сканирование* выбрать тип области сканирования — в данном случае *Чёрно-белое изображение (оттенки серого)*.
   В окне системы оптического распознавания появится отсканированное изображение текстовой страницы.
4. В окне *Изображение* выделить текстовую область для распознавания (рис. 2.13).

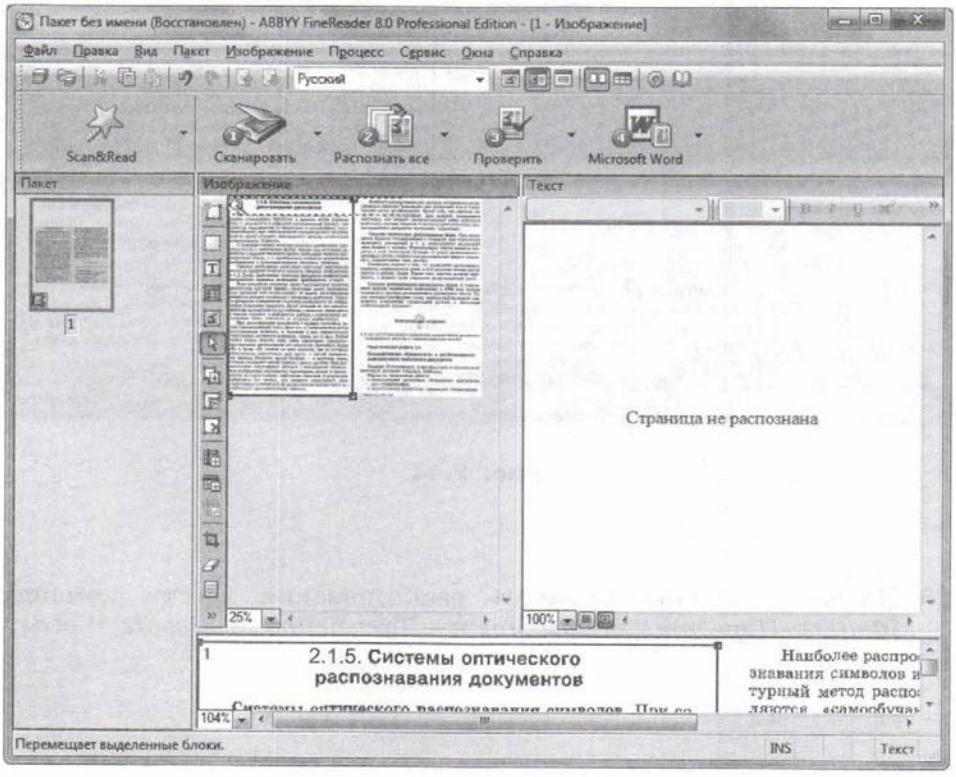


Рис. 2.13

5. Для преобразования графического изображения страницы в текстовый файл ввести команду [*Сервис—Распознать текст...*]. В окне *Текст* появится распознанный текст (рис. 2.14).

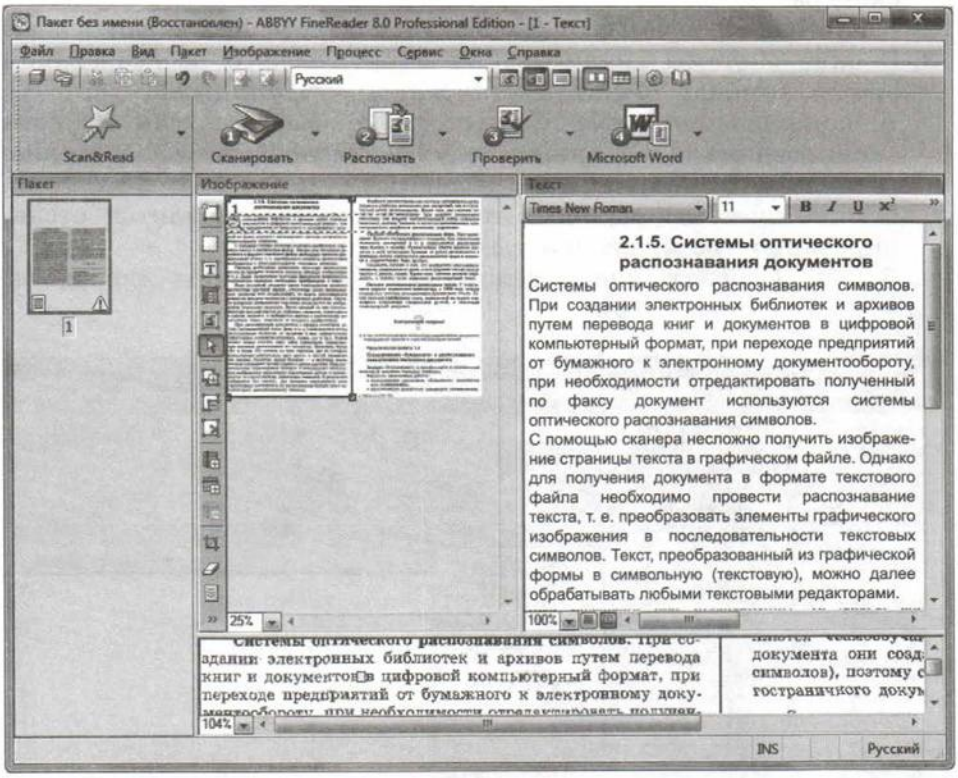


Рис. 2.14

6. После окончания процесса распознавания ввести команду [*Файл—Передать документ в—Документ Microsoft Word*].

Документ откроется в текстовом редакторе (рис. 2.15).

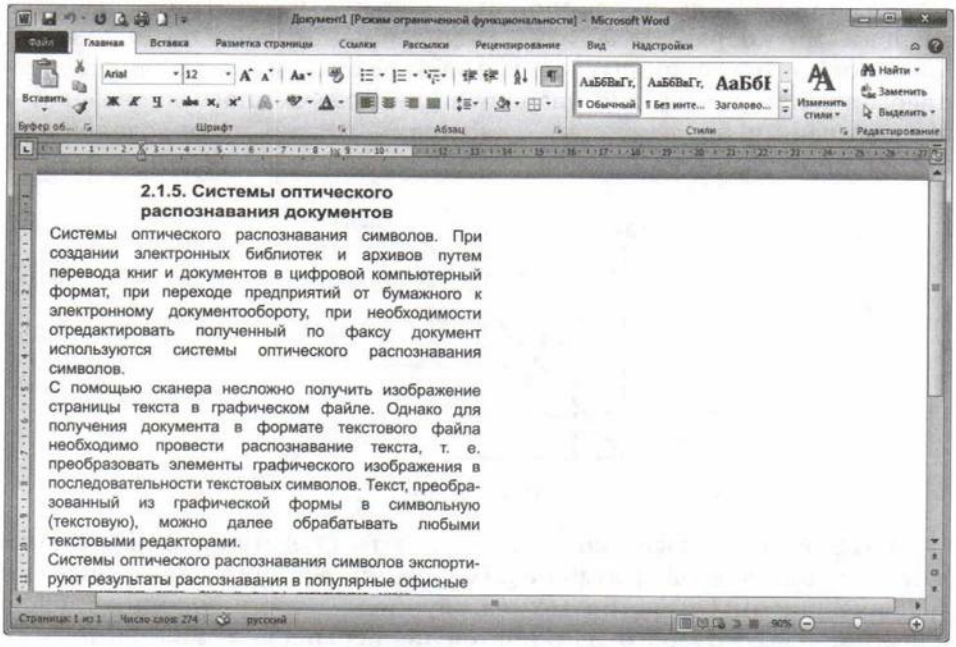


Рис. 2.15

7. После исправления ошибок, допущенных в процессе распознавания, в текстовом редакторе ввести команду [*Файл—Сохранить как...*] и выбрать место сохранения, имя и тип полученного текстового файла.

2.2. Кодирование и обработка графической информации

2.2.1. Кодирование графической информации

Аналоговый и дискретный способы представления графической информации. При аналоговом представлении физическая величина принимает бесконечное множество значений, причём её значения изменяются непрерывно. При дискретном представлении физическая величина принимает конечное множество значений, причём её значения изменяются скачкообразно.

В качестве примера аналогового и дискретного представления информации можно привести наклонную плоскость и лестницу. Положение тела на наклонной плоскости и на лестнице задаётся значениями координат X и Y. При движении тела по наклонной плоскости его координаты принимают бесконечное множество непрерывно изменяющихся значений из определённого диапазона, а при движении по лестнице — значения только из определённого набора, причём меняющиеся скачкообразно (рис. 2.16).

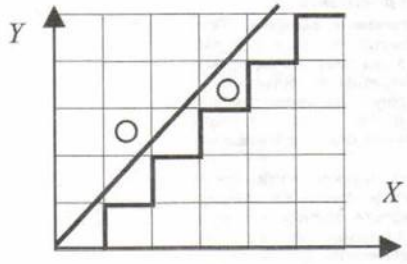


Рис. 2.16

Графическая информация может быть представлена в аналоговой или дискретной форме. Примером аналогового представления графической информации может служить, например, живописное полотно, цвет которого изменяется непрерывно, а дискретного — изображение, напечатанное с помощью струйного принтера и состоящее из отдельных точек разного цвета.

Пространственная дискретизация. Преобразование графической информации из аналоговой формы в дискретную производится путём пространственной дискретизации, т. е. разбиения непрерывного графического изображения на отдельные элементы. В результате пространственной дискретизации графическая информация представляется в виде растрового изображения, которое формируется из определённого количества строк, которые, в свою очередь, содержат определённое количество точек (пикселей).

Пиксель — минимальный участок изображения, для которого независимым образом можно задать цвет.

Важнейшей характеристикой качества растрового изображения является разрешение.

Разрешение растрового изображения определяется количеством пикселей по горизонтали и вертикали на единицу длины изображения.

Устройства вывода растровых изображений характеризуются разрешающей способностью. Разрешающая способность монитора обычно выражается в виде двух целых чисел, например 1600 × 200, и измеряется в ppi (англ. *pixel per inch* — пикселей на дюйм). Эти числа означают размеры изображения в пикселях по горизонтали и вертикали. Разрешающая способность принтеров и сканеров обычно указывается в dpi (англ. *dots per inch* — точек на дюйм) — количестве пикселей по горизонтали и вертикали на дюйм (например, 2400 × 1200 dpi).

Качество растрового изображения тем выше, чем больше его разрешение, т. е. чем меньше размер точки и, соответственно, чем большее количество точек составляет изображение.

Кодирование цвета точки. В процессе пространственной дискретизации производится кодирование, т. е. присваивание каждой точке конкретного числового значения цвета.

Качество дискретного изображения тем выше, чем больше для него используется цветов. Совокупность используемых цветов образует палитру цветов. Количество цветов N в палитре и количество информации i, необходимое для кодирования цвета каждой точки (**глубина цвета**), связаны между собой и могут быть вычислены по формуле (1.1):

$$N = 2^i.$$

Наиболее распространёнными значениями глубины цвета при кодировании цветных изображений являются 8, 16 или 24 бита на точку. Зная глубину цвета, по формуле (1.1) можно вычислить количество цветов в палитре.

Системы цветопередачи. С экрана монитора человек воспринимает цвет как сумму излучения трёх базовых цветов: красного, зелёного и синего. Такая система цветопередачи называется **RGB**, по первым буквам английских названий цветов (*Red* — красный, *Green* — зелёный, *Blue* — синий).

Цвета в палитре RGB формируются путём **сложения базовых цветов**, которые могут иметь различную интенсивность. Цвет палитры *Color* можно определить с помощью формулы (2.1). При этом надо учитывать глубину цвета — количество бит, отводимое в компьютере для кодирования цвета. Глубина цвета 24 бита означает, что на кодирование каждого из трёх базовых цветов выделяется по 8 бит. В этом случае для каждого из цветов возможны $N = 2^8 = 256$ уровней интенсивности. Уровни интенсивности задаются десятичными (от минимального — 0 до максимального — 255), двоичными (от 00000000 до 11111111) или шестнадцатеричными (от 00 до FF) кодами.

$$Color = R + G + B, \quad (2.1)$$

для глубины цвета 24 бита:

$0 \le R < 256,\ 0 \le G < 256,\ 0 \le B < 256$

При минимальных интенсивностях всех базовых цветов получается чёрный цвет, при максимальных интенсивностях — белый цвет. При максимальной интенсивности одного цвета и минимальной двух других получаются чистые красный, зелёный и синий цвета. Наложение зелёного и синего цветов образует голубой цвет (*Cyan*), красного и зелёного — жёлтый цвет (*Yellow*), красного и синего — пурпурный цвет (*Magenta*).

В **системе цветопередачи RGB** палитра цветов формируется путём сложения *красного*, *зелёного* и *синего* цветов с учётом их интенсивностей.

Напечатанное на бумаге изображение человек воспринимает в отражённом свете. Если на бумагу краски не нанесены, то падающий белый свет полностью отражается и мы видим белый цвет листа бумаги. Если краски нанесены, то они поглощают определённые цвета. При печати изображений на принтерах используется палитра цветов в системе **CMY** (*Cyan* — голубой, *Magenta* — пурпурный и *Yellow* — жёлтый), цвета в которой формируются путём **вычитания из белого цвета определённых цветов**. Нанесённая на бумагу голубая краска поглощает красный свет и отражает зелёный и синий свет, и мы видим голубой цвет. Нанесённая на бумагу пурпурная краска поглощает зелёный свет и отражает красный и синий свет, и мы видим пурпурный цвет. Нанесённая на бумагу жёлтая краска поглощает синий свет и отражает красный и зелёный свет, и мы видим жёлтый цвет.

Итак, цвета в палитре CMY создаются с помощью **наложения красок базовых цветов**. Цвет палитры *Color* можно определить с помощью формулы (2.2), в которой интенсивность каждой краски задаётся в процентах.

$$Color = C + M + Y, \quad (2.2)$$

где $0\% \le C \le 100\%,\ 0\% \le M \le 100\%,\ 0\% \le Y \le 100\%$

Смешивая попарно краски системы CMY, мы получим базовые цвета в системе цветопередачи RGB. Если нанести на бумагу пурпурную и жёлтую краски, то будет поглощаться зелёный и

синий свет, и мы увидим красный цвет. Если нанести на бумагу голубую и жёлтую краски, то будет поглощаться красный и синий свет, и мы увидим зелёный цвет. Если нанести на бумагу пурпурную и голубую краски, то будет поглощаться зелёный и красный свет, и мы увидим синий цвет.

Смешение трёх красок — голубой, жёлтой и пурпурной — должно приводить к полному поглощению света, и мы должны увидеть чёрный цвет. Однако на практике вместо чёрного цвета получается грязно-бурый цвет. Поэтому в цветовую модель CMY добавляют ещё один, истинно чёрный, цвет. Так как буква B уже используется для обозначения синего цвета, для обозначения чёрного цвета принята последняя буква в английском названии чёрного цвета blacK, т. е. K. Расширенная палитра получила название **CMYK**.

В струйных принтерах для получения изображений высокого качества используются четыре картриджа, содержащие базовые краски системы цветопередачи CMYK.

В системе цветопередачи CMYK палитра цветов формируется путём наложения голубой, пурпурной, жёлтой и чёрной красок.

Вопросы и задания

1. Приведите примеры аналогового и дискретного способов представления графической информации.
2. Как связаны между собой количество цветов в палитре и глубина цвета?
3. В каких единицах выражается разрешающая способность различных устройств ввода/вывода?
4. Чем различаются между собой системы цветопередачи RGB и CMY?

Практическая работа 2.5

Кодирование графической информации

Задание. Определить установленную на вашем компьютере разрешающую способность экрана монитора, измеренную в ppi.

Варианты выполнения работы:

- использование мониторов различного размера;
- использование различных разрешающих способностей экрана монитора.

Определение разрешающей способности экрана монитора в ppi

1. В операционной системе Windows щёлкнуть правой кнопкой мыши по *Рабочему столу*. Появится контекстное меню, в котором надо выбрать пункт *Разрешение экрана* — рис. 2.17.

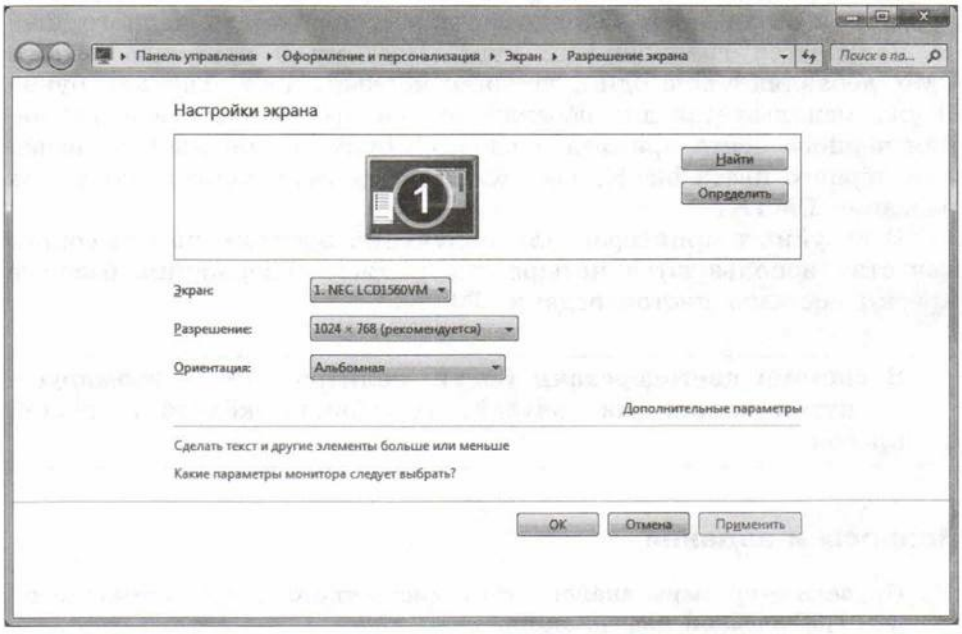


Рис. 2.17

С помощью раскрывающегося списка *Разрешение* узнать установленную разрешающую способность экрана монитора. Разрешение по горизонтали = 1024 пикселя.

2. Измерить с помощью линейки размер изображения на экране монитора по горизонтали (например, для 17"-монитора $L = 31{,}5$ см).
3. Определить, чему равен горизонтальный размер изображения на экране монитора в дюймах:
$L = 31{,}5$ см $= 31{,}5$ см$/2{,}54$ см/дюйм $\approx 12{,}4$ дюйма.
4. Определить разрешающую способность экрана монитора в ppi. Разрешение по горизонтали в ppi = 1024 точки/12,4 дюйма $\approx$ $\approx 82{,}5$ ppi.
5. Выполнить аналогичные измерения и расчёты для разрешения по вертикали.

2.2.2. Растровая графика

Растровые графические изображения формируются в процессе сканирования существующих на бумаге или фотоплёнке изображений и фотографий, а также при использовании цифровых фото- и видеокамер. Можно создать растровое графическое изображение и непосредственно на компьютере с использованием графического редактора.

Растровые изображения очень чувствительны к масштабированию (увеличению или уменьшению). При уменьшении растрового изображения несколько соседних точек преобразуются в одну, поэтому теряется чёткость мелких деталей изображения. При увеличении изображения увеличивается размер каждой точки и появляется ступенчатый эффект, который можно увидеть невооружённым глазом.

Растровые графические редакторы. Растровые графические редакторы являются средством обработки цифровых фотографий и отсканированных изображений, поскольку позволяют повышать их качество путём изменения цветовой палитры изображения и даже цвета каждого отдельного пикселя. Можно повысить яркость и контрастность старых или некачественных фотографий, удалить мелкие дефекты изображения (например, царапины), преобразовать чёрно-белое изображение в цветное и т. д. Растровые редакторы также позволяют выполнять редактирование растровых изображений по отдельным пикселям, выделение и последующее вырезание, копирование, вставку или стирание фрагментов, а также рисовать на компьютере.

Кроме того, растровые графические редакторы можно использовать для художественного творчества путем использования различных эффектов преобразования изображения. Обычную фотографию можно превратить в мозаичное панно, картину, рельефное изображение и др.

В состав операционной системы Windows входит простой растровый графический редактор Paint, широкие возможности по обработке растровых изображений имеют профессиональный графический редактор Adobe Photoshop и его бесплатный аналог GIMP.

Инструменты рисования растровых графических редакторов. Для создания изображения традиционными методами необходимо выбрать инструмент рисования (это могут быть фломастеры, кисть с красками, карандаши и многое другое). В растровых графических редакторах существуют аналогичные

инструменты, позволяющие изменять цвет определённых групп пикселей:

- *Карандаш* позволяет рисовать произвольные тонкие линии;
- *Кисть* позволяет рисовать произвольные линии различной толщины с использованием «кисти» выбранной формы;
- *Ластик* («кисть», рисующая цветом фона) позволяет стирать произвольные пиксели изображения, при этом размер *Ластика* можно менять;
- *Распылитель* позволяет разбрызгивать «краску» (закрашивать пиксели случайным образом) и таким образом закрашивать произвольные области;
- *Заливка* позволяет закрашивать замкнутые области целиком;
- *Надпись* позволяет создавать текстовые области на пиксельных изображениях. Установив курсор внутри текстовой области, можно произвести ввод текста, который становится частью пиксельного изображения.

Рисование графических примитивов. Растровые графические редакторы позволяют рисовать в поле рисования графические примитивы (прямая линия, кривая линия, прямоугольник, многоугольник и окружность). В растровом графическом редакторе нарисованный объект перестаёт существовать как самостоятельный элемент после окончания рисования и становится группой пикселей на изображении. Инструменты рисования примитивов:

- *Линия* позволяет нарисовать прямую линию; существует возможность перед рисованием задать тип линии (сплошная, пунктирная и т. д.), её толщину и цвет с помощью дополнительных меню;
- *Кривая* позволяет нарисовать произвольную линию и перетаскиванием мышью придать ей требуемую форму;
- *Прямоугольник* позволяет нарисовать прямоугольник: щелчком зафиксировать положение первой вершины, перетащить указатель по диагонали и зафиксировать положение второй вершины. Если в процессе рисования держать нажатой клавишу *Shift*, то будет нарисован квадрат;
- *Многоугольник* позволяет нарисовать многоугольник: последовательно щелчками зафиксировать положение вершин и двойным щелчком зафиксировать положение последней вершины;
- *Овал* позволяет нарисовать овал (эллипс): щелчком зафиксировать положение точки овала, перетащить указатель по диагонали и зафиксировать положение точки, противоположной относительно центра овала. Если в процессе рисования держать нажатой клавишу *Shift*, то будет нарисована окружность.

Операции копирования, перемещения и удаления. Редактирование изображения может производиться с использованием трёх основных операций: копирования, перемещения и удаления. При выполнении операции копирования выделенный фрагмент сохраняется в изображении и может быть многократно в него вставлен. При выполнении операции перемещения выделенный фрагмент удаляется из изображения, но может быть многократно в него вставлен. Операция удаления приводит к удалению выделенного фрагмента из изображения.

Перед выполнением каждой операции редактирования необходимо **выделить** область изображения (группу пикселей) в растровом редакторе. Обычно при этом возможны следующие выделения:

- *выделение прямоугольной области*;
- *выделение произвольной области*.

Палитра цветов. Различают основной цвет, которым рисуются контуры фигур, и цвет фона, которым фигуры закрашиваются. В меню палитры цветов обычно размещаются индикаторы основного цвета и цвета фона, которые отображают текущие выбранные оттенки.

Выбор цвета с использованием меню палитры ограничен, так как оно содержит только несколько десятков цветов. Однако графические редакторы позволяют использовать расширенную палитру цветов, в которой можно осуществлять выбор среди набора из десятков миллионов цветов.

Принцип формирования цветов в расширенной палитре базируется на том, что любой оттенок цвета можно получить, смешивая в определённой пропорции три базовых цвета: красный, зелёный и синий. Это можно сделать как с помощью мыши, перемещая указатель по цветовому полю, так и вводя величины интенсивностей каждого базового цвета (в интервале от 0 до 255) с клавиатуры в соответствующие текстовые поля.

В большинстве графических редакторов для копирования цветов можно использовать инструмент *Пипетка*. Щелчок левой кнопкой мыши в области с выбранным цветом задаёт его в качестве основного цвета, а щелчок правой кнопкой — в качестве цвета фона.

Геометрические преобразования. Растровые изображения могут быть подвергнуты геометрическим преобразованиям:

- изменению размера по горизонтали и вертикали;
- поворотам по часовой стрелке или против часовой стрелки;
- наклонам на различные углы;
- отражениям в различных плоскостях.

В растровых редакторах имеется масштабирующий инструмент, который позволяет увеличивать или уменьшать масштаб представления изображения или рисунка на экране, но не влияет при этом на его реальные размеры. Обычно такой инструмент называется *Лупа*.

Форматы растровых графических файлов. Форматы графических файлов определяют способ хранения информации в файле, а также используемый алгоритм сжатия.

Растровые графические файлы имеют обычно большой информационный объём, так как в них хранятся коды цветов всех точек изображения. Для растровых графических файлов обычно применяется **сжатие**, которое отличается от архивирования с помощью программ-архиваторов тем, что алгоритм сжатия включается непосредственно в формат графического файла (форматы BMP, TIFF, GIF, PNG и др.).

Для сжатия изображений, содержащих большие области однотонной закраски, наиболее эффективно применение алгоритма сжатия, который заменяет последовательность повторяющихся величин (пикселей одинакового цвета) на две величины (пиксель и количество его повторений). Для рисунков с мелкими деталями изображения целесообразно применение другого метода сжатия, который использует поиск повторяющихся в рисунке «узоров».

Для сжатия отсканированных фотографий и иллюстраций используется метод сжатия JPEG, который отбрасывает избыточную для человеческого восприятия информацию (компьютер обеспечивает воспроизведение более 16 млн различных цветов, тогда как человек вряд ли способен различить более сотни цветов и оттенков). Применение метода JPEG позволяет сжимать файлы в десятки раз, однако приводит к необратимой потере информации (файлы не могут быть восстановлены в первоначальном виде).

GIF-анимация. GIF-анимация является последовательностью растровых графических изображений (кадров), которые хранятся в одном растровом графическом файле в формате GIF. Для создания последовательности растровых изображений и для их преобразования в GIF-анимацию можно использовать многофункциональные растровые графические редакторы или специальные редакторы GIF-анимаций.

В процессе просмотра такого GIF-файла растровые графические изображения последовательно появляются на экране монитора, что создаёт иллюзию движения. При создании GIF-анимации

можно задать величину задержки появления каждого кадра: чем она меньше, тем лучше качество анимации. Кроме того, можно установить количество повторений (от одного до бесконечности) последовательности кадров, хранящихся в GIF-файле.

Большое количество кадров ведёт к лучшему качеству анимации, но увеличивает размер GIF-файла. Для уменьшения его информационного объёма можно анимировать только некоторые части изображения.

Вопросы и задания

1. Почему при уменьшении и увеличении растрового изображения ухудшается его качество?
2. В чём состоят основные различия форматов растровых графических файлов?

Практическая работа 2.6

Работа с растровой графикой

Задание 1. Осуществить геометрические преобразования изображения в растровом графическом редакторе (например, отразить и растянуть по вертикали и наклонить по горизонтали слово «информатика»).

Варианты выполнения работы:

- использование различных графических редакторов;
- использование различных изображений и геометрических преобразований.

Задание 2. В растровом графическом редакторе осуществить преобразование растрового фотографического изображения в мозаику и барельеф.

Варианты выполнения работы:

- использование различных графических редакторов;
- использование различных фильтров для преобразования изображений.

Задание 3. В растровом графическом редакторе растровое изображение в формате BMP сохранить в различных графических форматах (GIF, JPEG, PNG и TIFF).

Варианты выполнения работы:

- использование различных графических редакторов;
- использование различных растровых изображений.

Задание 1. Геометрические преобразования изображения в растровом редакторе Paint

1. В операционной системе Windows запустить редактор Paint командой [*Пуск—Все программы—Стандартные—Paint*].
   На ленте перейти на вкладку *Главная*, щёлкнуть по кнопке *Текст* и ввести слово «информатика» (рис. 2.18).
   На появившейся вкладке *Текст* на ленте выбрать параметры шрифта.

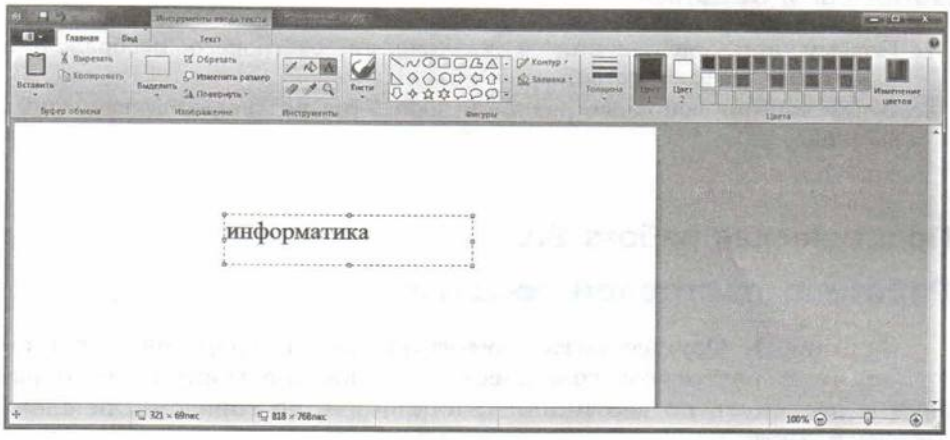


Рис. 2.18

2. На вкладке *Главная* щёлкнуть по кнопке *Повернуть или отразить*.
   В появившемся раскрывающемся списке кнопки *Повернуть или отразить* выбрать параметры действия (например, *Отразить по вертикали*) — рис. 2.19.

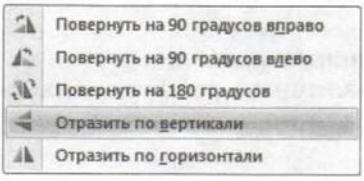


Рис. 2.19

3. На вкладке *Главная* щёлкнуть по кнопке *Изменить размер и наклонить*.
   В появившемся диалоговом окне *Изменение размеров и наклона* выбрать параметры действия (например, *Изменить*, *По вертикали* и *Наклон*, *По горизонтали*) — рис. 2.20.

Рис. 2.20

4. В результате будет получена отражённая по вертикали, увеличенная по вертикали и наклонённая по горизонтали надпись (рис. 2.21).

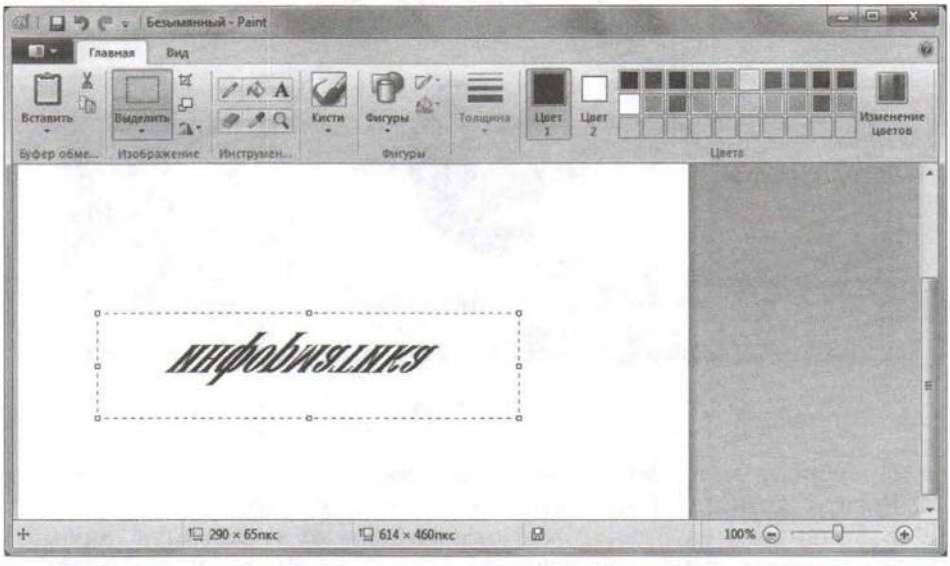


Рис. 2.21

Задание 2. Преобразование растрового изображения в графическом редакторе GIMP

Электронное приложение к главе 2.

1. В операционной системе Windows запустить растровый графический редактор GIMP, предварительно установив его с веб-ресурса. В операционной системе Linux выполнить команду [*Графика—GIMP* (*Редактор изображений*)].
2. В появившемся диалоговом окне графического редактора открыть растровый графический файл в формате BMP (например, rastr.bmp) командой [*Файл—Открыть...*] — рис. 2.22.

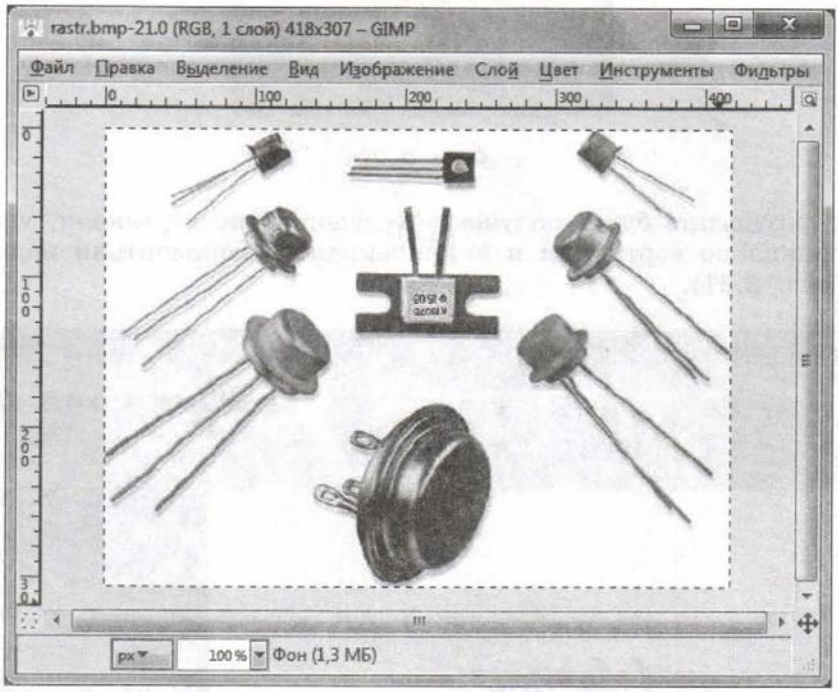


Рис. 2.22

Преобразуем изображение в мозаику.

3. Ввести команду [*Фильтры—Искажения—Мозаика...*].
   В появившемся диалоговом окне *Мозаика* установить параметры преобразования фотографического изображения в мозаику (рис. 2.23, 2.24).

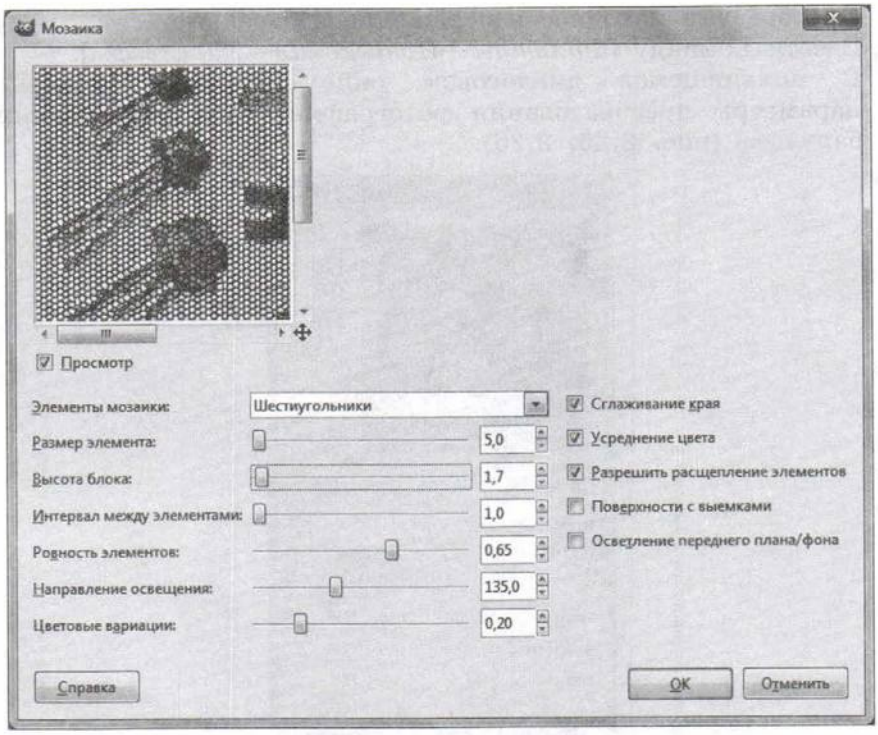


Рис. 2.23

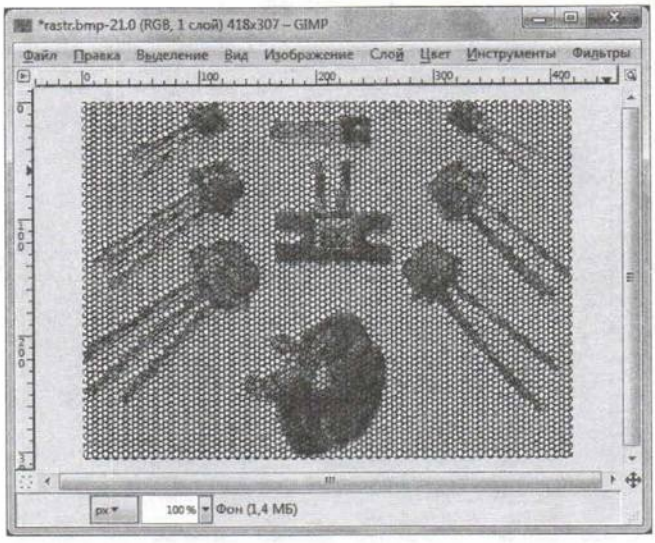


Рис. 2.24

Преобразуем исходное изображение в барельеф.

4. Ввести команду [*Фильтры—Искажения—Барельеф...*].
   В появившемся диалоговом окне *Барельеф* установить параметры преобразования фотографического изображения в барельеф (рис. 2.25, 2.26).

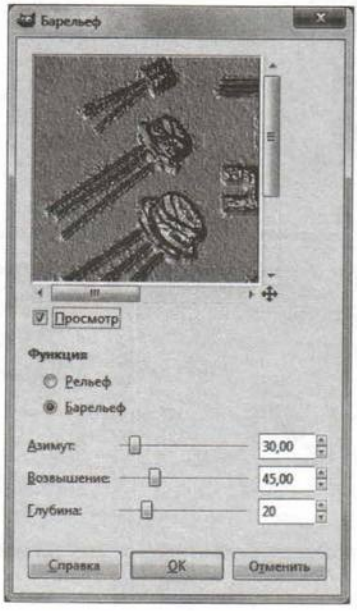


Рис. 2.25

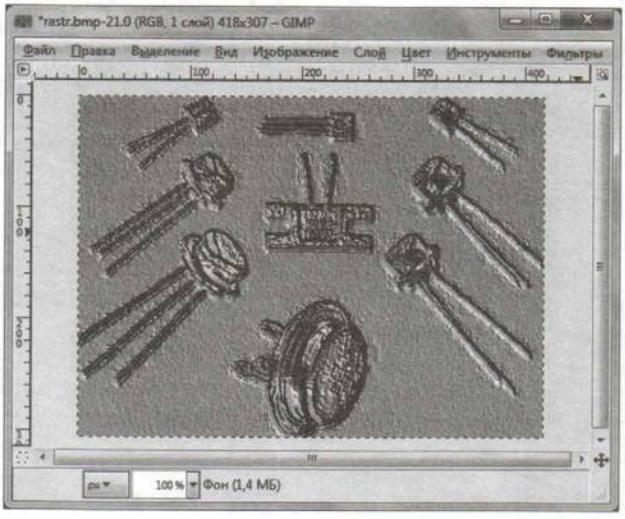


Рис. 2.26

Задание 3. Сохранение растрового изображения в различных графических форматах в графическом редакторе GIMP

Электронное приложение к главе 2.

1. Запустить растровый графический редактор GIMP.
2. В появившемся диалоговом окне графического редактора открыть растровый графический файл в формате BMP (например, rastr.bmp) командой [*Файл—Открыть...*].

Сохраним это изображение в различных графических форматах: GIF, JPEG, PNG и TIFF, установив для каждого формата запрашиваемые параметры сохранения.

3. Ввести команду [*Файл—Экспортировать...*].

В появившемся окне *Экспорт изображения* щёлкнуть по ссылке *Выберите тип файла* (*По расширению*).

В списке выбрать тип формата файла *Изображение GIF*.

Щёлкнуть по кнопке *Эскпортировать*.

В появившемся диалоговом окне *Экспортировать изображение как GIF* (рис. 2.27) оставить настройки без изменений и щёлкнуть по кнопке *Экспорт*.

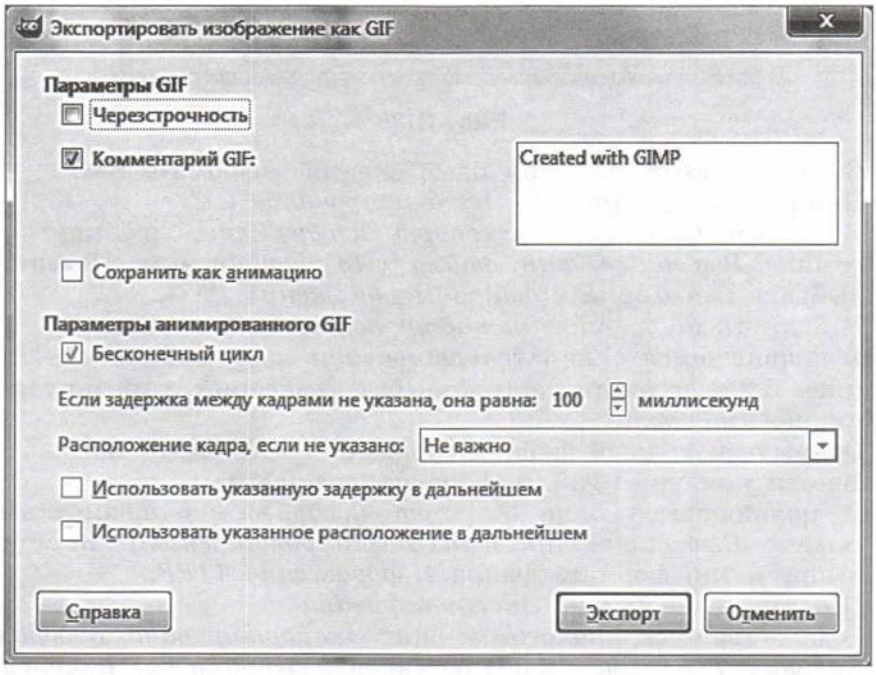


Рис. 2.27

4. Открыть исходный файл изображения в формате BMP.
   Ввести команду [*Файл—Экспортировать*].
   В появившемся окне *Экспорт изображения* щёлкнуть по ссылке *Выберите тип файла* (*По расширению*). В списке выбрать тип формата файла *Изображение JPEG*.
   Щёлкнуть по кнопке *Экспортировать*.
   В появившемся диалоговом окне *Экспортировать изображение как JPEG* (рис. 2.28) оставить настройки без изменений и щёлкнуть по кнопке *Экспорт*.

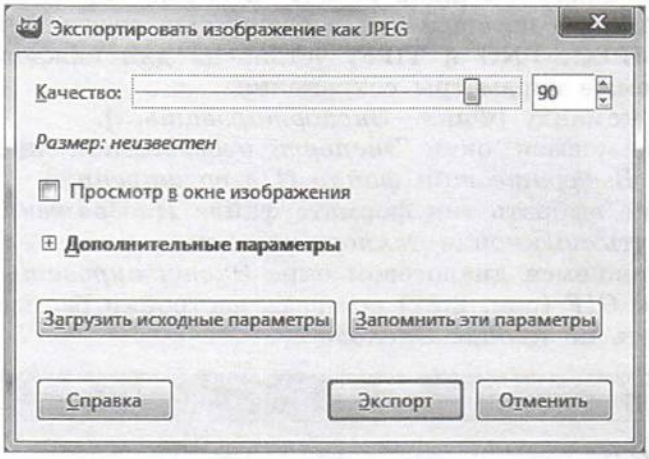


Рис. 2.28

5. Открыть исходный файл изображения в формате BMP.
   Ввести команду [*Файл—Экспортировать...*].
   В появившемся окне *Экспорт изображения* щёлкнуть по ссылке *Выберите тип файла* (*По расширению*). В списке выбрать тип формата файла *Изображение PNG*.
   Щёлкнуть по кнопке *Экспортировать*.
   В появившемся окне *Экспортировать изображение как PNG* (рис. 2.29) оставить настройки без изменений и щёлкнуть по кнопке *Экспорт*.
6. Открыть исходный файл изображения в формате BMP.
   Ввести команду [*Файл—Экспортировать...*].
   В появившемся окне *Экспорт изображения* щёлкнуть по ссылке *Выберите тип файла* (*По расширению*). В списке выбрать тип формата файла *Изображение TIFF*.
   Щёлкнуть по кнопке *Экспортировать*.
   В появившемся диалоговом окне *Экспортировать изображение как TIFF* (рис. 2.30) оставить настройки без изменений и щёлкнуть по кнопке *Экспорт*.

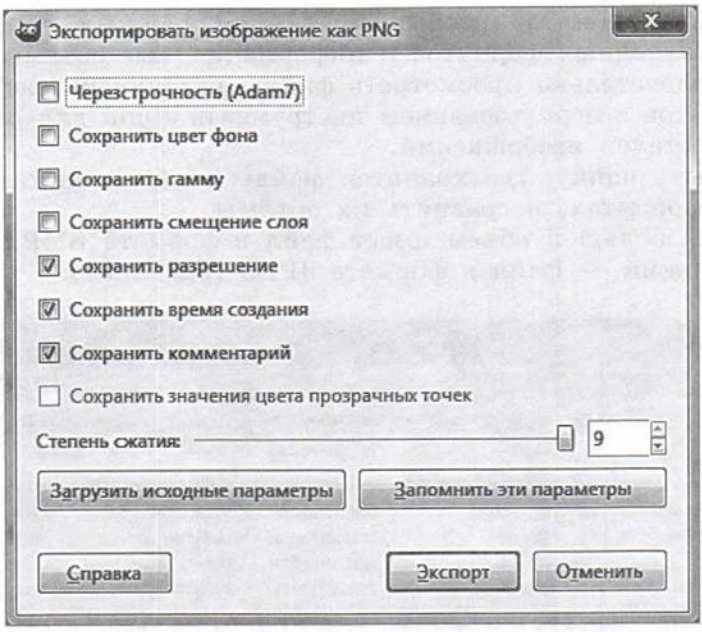


Рис. 2.29

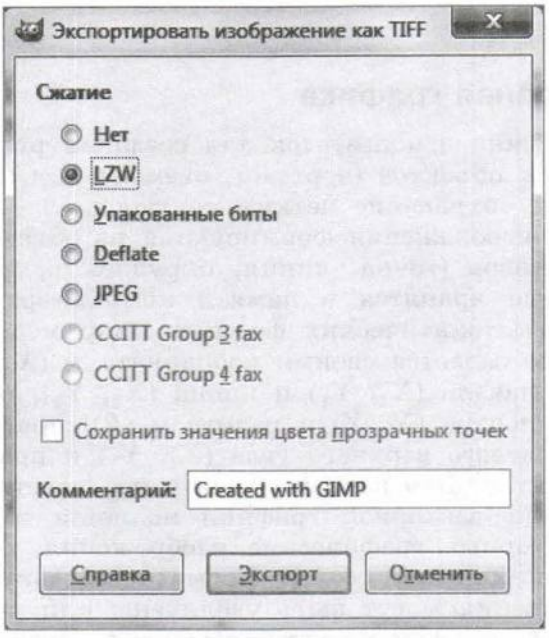


Рис. 2.30

Сравним качество изображений в полученных графических файлах различных форматов и информационные объёмы файлов.

7. Последовательно просмотреть файлы различных графических форматов с использованием инструмента *Лупа* для рассмотрения деталей изображений.

   Открыть папку, где хранятся файлы изображений в различных форматах, и сравнить их объёмы.

   Самый большой объём имеет файл в формате BMP, а самый маленький — файл в формате JPEG (рис. 2.31).

Рис. 2.31

2.2.3. Векторная графика

Векторная графика используется для создания рисунков, а также графических объектов (чертежи, схемы и т. д.), для которых имеет значение сохранение чётких контуров.

Векторные изображения формируются из объектов — графических примитивов (точка, линия, окружность, прямоугольник и т. д.), которые хранятся в памяти компьютера в виде описывающих их математических формул. Например, графический примитив точка задается своими координатами (X, Y), линия — координатами начала (X_1, Y_1) и конца (X_2, Y_2), окружность — координатами центра (X, Y) и радиусом (R), прямоугольник — координатами левого верхнего угла (X_1, Y_1) и правого нижнего угла (X_2, Y_2) и т. д. Для каждого примитива задаётся также цвет.

Достоинством векторной графики является то, что файлы, хранящие векторные графические изображения, обычно имеют сравнительно небольшой объём. Кроме того, векторные графические изображения могут быть увеличены или уменьшены без потери качества, так как масштабирование изображений производится с помощью простых математических операций (умножения

параметров графических примитивов на коэффициент масштабирования).

Рисование с использованием векторных графических редакторов. Векторный графический редактор можно рассматривать как графический конструктор, который позволяет создавать рисунки из отдельных объектов (линий, прямоугольников, многоугольников, окружностей и др.). Объекты могут быть и трёхмерными (шары, кубы и параллелепипеды, пирамиды и др.).

В векторных редакторах можно создавать текстовые области, в которых вводится и форматируется текст. Кроме того, для ввода надписей к рисункам можно использовать выноски различных форм.

Векторный рисунок легко редактировать, так как каждый графический примитив может существовать как самостоятельный объект, который можно без потери качества изображения перемещать, изменять его размеры, цвет и прозрачность.

В векторном редакторе **выделение** объектов осуществляется с помощью специального инструмента (на панели инструментов изображается стрелкой). Для выделения объекта достаточно выбрать этот инструмент и щёлкнуть по нужному объекту на рисунке. Вокруг выделенного объекта появятся восемь меток в виде маленьких квадратиков по его периметру.

Если поместить на такую метку указатель мыши, то он примет вид стрелки, направленной в две противоположные стороны. Перетаскивая метку, можно изменять размер объекта.

Для перемещения объекта необходимо установить указатель мыши внутри выделенной области (он примет вид стрелки, указывающей «на все четыре стороны») и перетащить объект.

Простой векторный графический редактор OpenOffice Draw входит в состав интегрированного офисного приложения OpenOffice; удобный векторный редактор встроен в текстовый редактор Microsoft Word.

Видимость объектов. Каждый графический примитив рисуется в своём слое, поэтому рисунки состоят из множества слоёв. Графические примитивы можно накладывать друг на друга, при этом одни объекты могут заслонять другие. Например, если сначала был нарисован прямоугольник, а затем поверх него — окружность, то слой окружности будет располагаться поверх слоя прямоугольника, и окружность заслонит прямоугольник.

Существует возможность изменения видимости объектов путём изменения порядка размещения их слоев на рисунке. Для этого используются операции изменения порядка, которые позволяют перемещать выделенный объект на передний план (самый верх-

ний слой рисунка) или на задний план (самый нижний слой рисунка), а также на один слой вперёд или назад.

Заливка объектов. В векторных редакторах существует возможность осуществлять заливку объектов выбранным цветом (в том числе градиентную). При градиентной заливке интенсивность закраски может изменяться по горизонтали, по вертикали, диагонально или от центра объекта. Кроме того, объекты могут быть заштрихованы различными способами (линиями, клетками и т. д.).

Прозрачность объектов. Для каждого объекта (слоя рисунка) можно задать степень прозрачности (в процентах от 0 до 100). При нулевой прозрачности объект, нарисованный на нижерасположенном слое, виден не будет. Наоборот, при стопроцентной прозрачности он будет виден полностью.

Группировка объектов. Отдельные графические примитивы можно преобразовать в единый объект (сгруппировать). С этим новым объектом можно производить те же действия, что и с отдельными графическими примитивами, т. е. перемещать, изменять размеры, цвет и другие параметры. Можно и наоборот, разбить объект, состоящий из нескольких объектов, на самостоятельные объекты (разгруппировать объект).

Выравнивание объектов. Для большей точности рисования объектов в окне редактора по горизонтали и по вертикали размещаются линейки с делениями. Для выравнивания нарисованных объектов по горизонтали и вертикали используется сетка, к которой привязываются объекты. Точность привязки объектов можно менять, изменяя размер ячейки сетки.

Системы компьютерного черчения. Системы компьютерного черчения являются векторными графическими редакторами, предназначенными для создания чертежей. При классическом черчении с помощью карандаша, линейки и циркуля производится построение элементов чертежа (отрезков, окружностей и прямоугольников) с точностью, которую предоставляют чертёжные инструменты. Использование же систем компьютерного черчения позволяет создавать чертежи с гораздо большей точностью. Кроме того, системы компьютерного черчения дают возможность измерять расстояния, углы, периметры и площади начерченных объектов.

Пространственные соотношения между реальными объектами (положение и ориентация объектов в пространстве и их размеры) изучаются в курсе геометрии. Важное место в школьном курсе геометрии занимают геометрические построения с использованием линейки и циркуля. Для создания геометрических

моделей на компьютере удобно использовать системы компьютерного черчения.

Системы компьютерного черчения могут использоваться в школьном курсе технологии, так как позволяют создавать чертежи деталей, в том числе трёхмерные. Такие системы дают возможность грамотно оформить чертёж: обозначить на нём размеры деталей и сделать надписи в соответствии с существующими стандартами.

Системы компьютерного черчения используются в качестве инструмента автоматического проектирования на производстве, так как обеспечивают возможность реализации сквозной технологии проектирования и изготовления деталей. На основе компьютерных чертежей генерируются управляющие программы для станков с числовым программным управлением (ЧПУ), в результате по компьютерным чертежам могут изготавливаться высокоточные детали из металла, пластмассы, дерева и других материалов.

Система компьютерного черчения КОМПАС специально предназначена для обучения компьютерному черчению в школах. КОМПАС можно использовать для выполнения геометрических построений с помощью циркуля и линейки, а также при создании чертежей деталей.

Форматы векторных графических файлов. Наиболее широко распространённым форматом векторных графических файлов является формат WMF, который используется для хранения коллекции графических изображений Microsoft Clip Gallery. Некоторые программы обработки изображений используют оригинальные форматы, которые распознаются только самой создающей программой. Например, векторный редактор OpenOffice Draw сохраняет файлы в собственном формате ODG, система компьютерного черчения КОМПАС — в формате FRM, а система векторной флеш-графики Adobe Animate — в специализированном формате FLA.

Флеш-анимация. Компьютерная анимация использует быструю смену кадров (как это делается в кино), которую глаз человека воспринимает как непрерывное движение. Чем большее количество кадров меняется за одну секунду (в кино в секунду сменяется 24 кадра), тем более полная иллюзия движения возникает у человека.

Флеш-анимация базируется на использовании векторной графики и представляет собой последовательность векторных рисунков (кадров). Кадр строится с использованием набора векторных графических объектов (прямых и произвольных линий, окружностей и прямоугольников), для каждого из которых можно задать размер, цвет линий и заливки и другие параметры.

Достоинством флеш-анимации является то, что нет необходимости прорисовывать каждый кадр. Достаточно нарисовать ключевые кадры и задать тип перехода между ними (свободная трансформация, трансформация с вращением, трансформация с отражением и т. д.). Редактор флеш-анимации автоматически построит промежуточные кадры. Если промежуточных кадров много, то анимация получается плавной, а если мало, то резкой.

В процессе просмотра флеш-анимации векторные кадры последовательно появляются на экране монитора, что и создает иллюзию движения. При создании флеш-анимации можно задать количество кадров в секунду: чем оно больше, тем лучше качество анимации.

Достоинством флеш-анимации является небольшой информационный объём файлов, и поэтому она широко используется на веб-сайтах в Интернете.

Для разработки флеш-анимации используется система векторной флеш-графики Adobe Animate.

Вопросы и задания

1. Что в векторных графических редакторах позволяет изменять видимость объектов, образующих рисунок?
2. В каких случаях полезно воспользоваться операцией группировки объектов?

Практическая работа 2.7

Работа с трёхмерной векторной графикой

Задание. Нарисовать различные трехмёрные тела (шар, конус и т. д.).

Варианты выполнения работы:

- рисование различных трёхмерных геометрических фигур;
- установка различных параметров (освещённость, материал, цвет и др.).

Создание трёхмерной графики в векторном редакторе OpenOffice Draw

Электронное приложение к главе 2.

1. В операционной системе Windows или Linux запустить интегрированное офисное приложение OpenOffice и ввести команду [*Файл—Создать—Рисунок*].

2. Ввести команду [*Вид—Панель инструментов—3D-объекты*] (рис. 2.3.2).
Появится панель *3D-объекты*.

Рис. 2.32

3. Последовательно выбрать на панели и нарисовать в поле рисования *Сферу*, *Пирамиду*, *Тор*, *Конус*, *Куб* и *Цилиндр* (рис. 2.3.3).

Рис. 2.33

При рисовании трёхмерных тел можно устанавливать различные параметры (режим освещённости, цвет и текстуру поверхности и др.).

4. Выделить одну из трёхмерных фигур (например, сферу) и в контекстном меню выбрать пункт *Трёхмерные эффекты*.
В появившемся диалоговом окне *Трёхмерные эффекты* (рис. 2.34) щёлкнуть по кнопке *Освещение* и выбрать источник света.
Установить цвет источника и цвет рассеянного света.
Перемещая положение источника с помощью мыши или полос прокрутки, в окне *Просмотр* пронаблюдать изменение освещённости трёхмерного тела.
Для присваивания установленных свойств щёлкнуть по кнопке *Применить* (кнопка с зелёной «галочкой»).

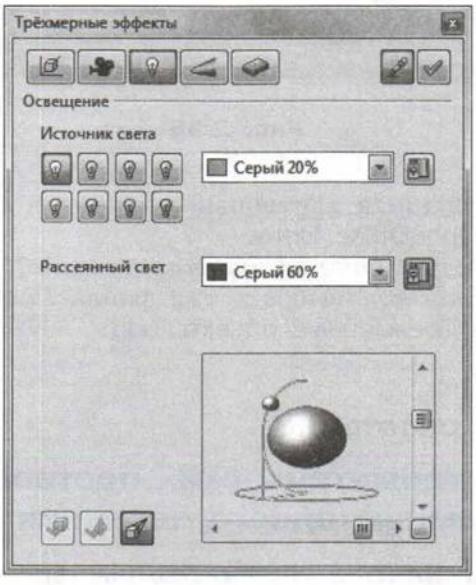


Рис. 2.34

5. Щёлкнуть по кнопке *Материал* и выбрать тип материала, цвет объекта и цвет освещения (рис. 2.35).
   Выбрать цвет и интенсивность для точки блика.
   В окне *Просмотр* наблюдать результат применения выбранных установок к объекту.
   Щёлкнуть по кнопке *Применить*.

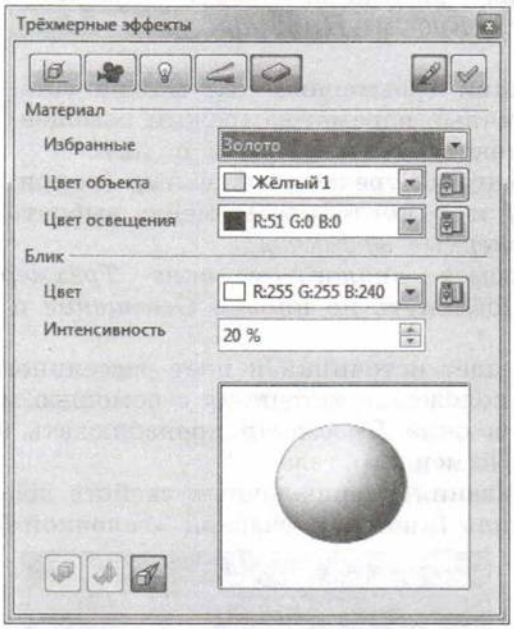


Рис. 2.35

Сохраним созданный векторный рисунок в собственном формате редактора OpenOffice Draw.

6. Ввести команду [*Файл—Сохранить как...*] и в диалоговом окне *Сохранить как* выбрать тип файла *Рисунок ODF (.odg)* и имя файла Трёхмерные объекты.odg.

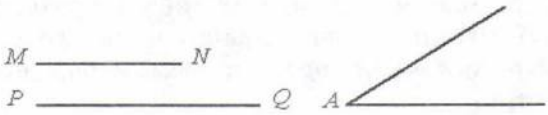
Практическая работа 2.8

Выполнение геометрических построений в системе компьютерного черчения КОМПАС

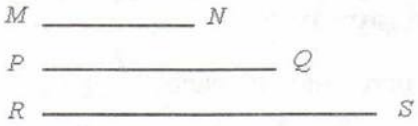
Задания. Выполнить в системе компьютерного черчения следующие геометрические построения:

1.8.1. Отложить от луча OM угол, равный заданному углу A:

1.8.2. Построить треугольник по двум сторонам и углу между ними:

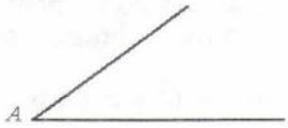


1.8.3. Построить треугольник по трём сторонам:



1.8.4. Даны прямая и точка на ней. Построить прямую, проходящую через заданную точку и перпендикулярную к заданной прямой:

1.8.5. Дан неразвёрнутый угол A. Построить его биссектрису:



Варианты выполнения работы:

- выполнение различных геометрических построений.

Пояснение к рисованию объектов. Можно рисовать объект (отрезок, окружность, прямоугольник и др.) с помощью мыши или вводить данные об объекте вручную в соответствующих полях на *Панели свойств* (рис. 2.36), которая находится внизу системы компьютерного черчения. Например, для отрезка можно задать координаты его концов, длину, угол наклона и стиль линии.

Рис. 2.36

Можно снять значения параметров непосредственно с чертежа. Для подобного снятия параметров используется *Геометрический калькулятор*. Если в процессе рисования объекта установить указатель мыши над каким-либо из полей и щёлкнуть правой кнопкой мыши, то на экране появляется меню команд *Геометрического калькулятора*, причём набор команд зависит от типа параметра.

2.8.1. Геометрическое построение угла, равного заданному

Электронное приложение к главе 2.

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма:

1. Построить угол A и луч OM.
2. Построить окружность произвольного радиуса с центром в точке A, обозначить точки пересечения окружности со сторонами угла буквами B и C.
3. Построить окружность такого же радиуса с центром в точке O, обозначить точку пересечения окружности с лучом OM буквой D.
4. Построить окружность с радиусом, равным отрезку BC с центром в точке D, обозначить точку пересечения окружностей буквой E.
5. Соединить отрезком точки O и E. Угол EOD, равный углу A, построен.

Начертим геометрические объекты, заданные в условии задачи: произвольный угол и отрезок.

1. С помощью *Компактной панели* вызвать панель *Геометрия*. Выбрать объект *Отрезок* и построить сначала произвольный угол A (начертить два отрезка, выходящих из одной точки), а затем построить произвольный луч OM (начертить отрезок).

Введём обозначение точек на чертеже с помощью панели *Обозначения*.

2. С помощью *Компактной панели* вызвать панель *Обозначения*. Щёлкнуть по кнопке *Ввод текста* и последовательно ввести обозначения угла и концов отрезка (рис. 2.37).

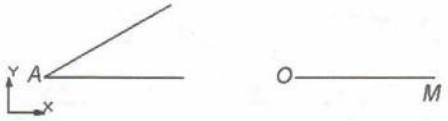


Рис. 2.37

Построим окружность произвольного радиуса с центром в вершине заданного угла A, которая пересечёт стороны угла в точках B и C.

3. На панели *Геометрия* выбрать объект *Окружность* и построить окружность с центром в точке A.
На панели *Обозначения* щёлкнуть по кнопке *Ввод текста* и обозначить точки пересечения окружности со сторонами угла буквами B и C (рис. 2.38).

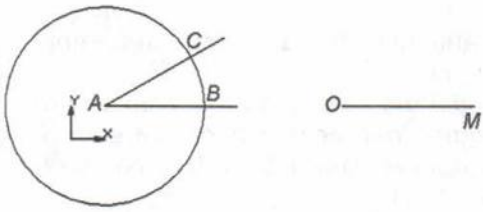


Рис. 2.38

Построим окружность того же радиуса с центром в начале заданного луча OM, которая пересечёт отрезок OM в точке D.

4. На панели *Геометрия* выбрать объект *Окружность*.
На *Панели свойств* (рис. 2.39) щёлкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
На чертеже навести курсор сначала на точку A, а затем на точку B.
Центр появившейся окружности заданного радиуса переместить в точку O.

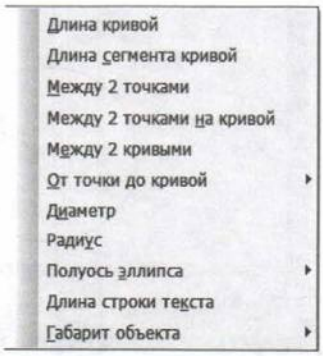


Рис. 2.39

5. На панели *Обозначения* щёлкнуть по кнопке *Ввод текста* и обозначить точку пересечения окружности с отрезком *OM* буквой *D* (рис. 2.40).

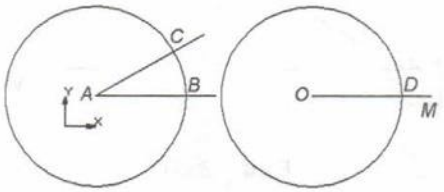


Рис. 2.40

Построим окружность с центром в точке *D* заданного радиуса *BC*.

6. На панели *Геометрия* выбрать объект *Окружность*.
   На *Панели свойств* щёлкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
   На чертеже навести курсор сначала на точку *C*, а затем на точку *B*.
   Центр появившейся окружности заданного радиуса переместить в точку *D*.
7. На панели *Обозначения* щёлкнуть по кнопке *Ввод текста* и обозначить точку пересечения окружностей буквой *E*.
8. Соединить отрезком точки *O* и *E*. Угол *EOD*, равный углу *A*, построен (рис. 2.41).

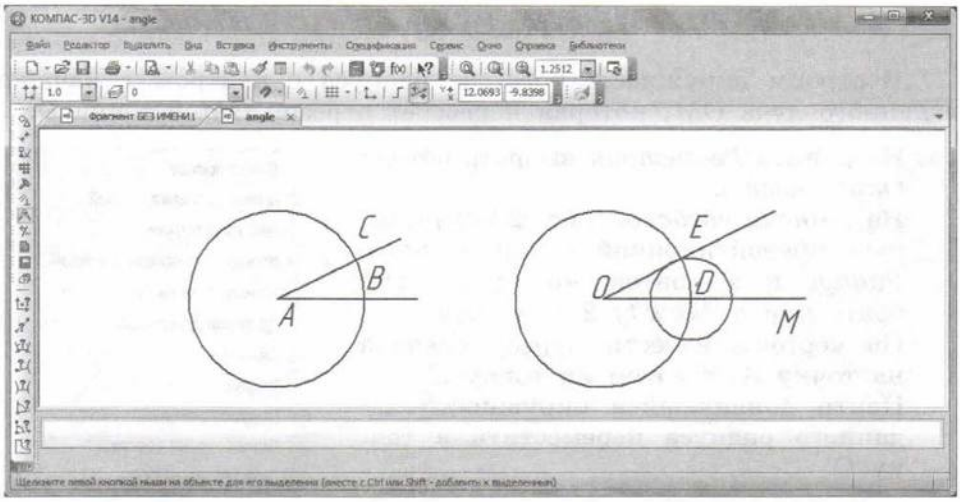


Рис. 2.41

2.8.2. Построение треугольника по двум сторонам и углу между ними

Электронное приложение к главе 2.

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма:

1. Построить угол A и два отрезка MN и PQ.
2. Построить угол K, равный заданному углу A (см. практическую работу 2.8.1).
3. Отложить на сторонах угла К отрезки, длины которых равны длинам заданных отрезков MN и PQ, путём построения двух окружностей соответствующих радиусов с центром в точке K. Обозначить точки пересечения окружностей со сторонами угла буквами B и C.
4. Соединить отрезком точки B и C. Треугольник KBC построен.

Начертим геометрические объекты, заданные в условии задачи: произвольный угол и два отрезка.

1. Построить произвольный угол A (начертить два отрезка, выходящих из одной точки). Построить два отрезка MN и PQ. Ввести обозначение точек на чертеже.

Построим угол, равный заданному.

2. Построить угол K, равный заданному углу A (см. практическую работу 2.8.1).

Отложим на сторонах угла отрезки, длины которых равны длинам заданных отрезков MN и PQ.

3. С помощью *Компактной панели* вызвать панель *Геометрия* и выбрать объект *Окружность*.
   На *Панели свойств* щёлкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
   На чертеже навести курсор сначала на точку M, а затем на точку N.
   Центр появившейся окружности заданного радиуса переместить в точку K.
4. Аналогично построить окружность, радиус которой равен длине отрезка PQ.
5. С помощью *Компактной панели* вызвать панель *Обозначения*. Щёлкнуть по кнопке *Ввод текста* и обозначить точки пересечения сторон угла и окружностей B и C.

6. Соединить отрезком точки B и C. Треугольник KBC построен (рис. 2.42).

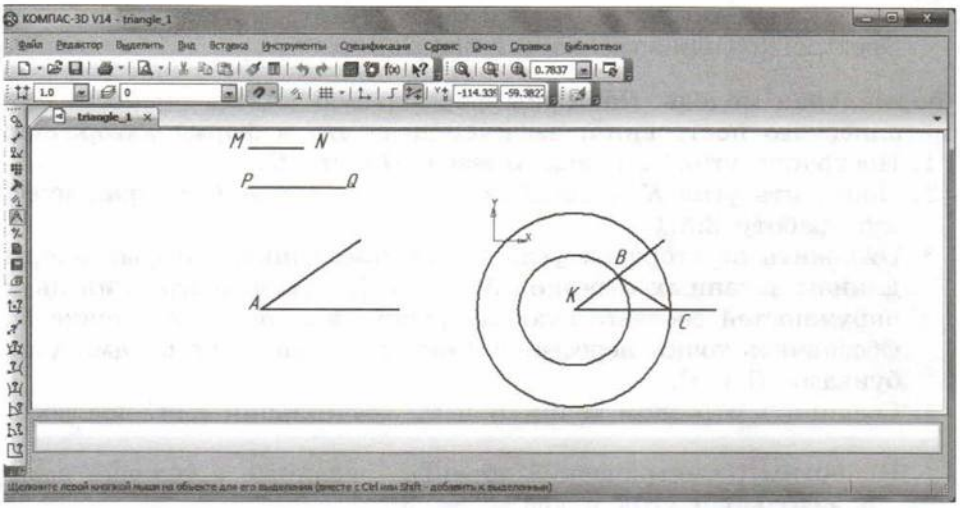


Рис. 2.42

2.8.3. Построение треугольника по трём сторонам

Электронное приложение к главе 2.

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма:

1. Построить три отрезка MN, PQ и RS, причём длина ни одного из них не должна превышать сумму длин двух других.
2. Провести прямую и отложить на ней отрезок, длина которого равна длине заданного отрезка RS, обозначить его концы буквами A и B.
3. Построить две окружности, радиусы которых равны длинам заданных отрезков MN и PQ, с центрами в точках A и B. Обозначить точку пересечения окружностей буквой C.
4. Построить отрезки AC и BC. Треугольник ABC построен.

Начертим геометрические объекты, заданные в условии задачи: три отрезка (длина ни одного из них не должна превышать сумму длин двух других).

1. Построить три отрезка MN, PQ и RS и ввести обозначение точек на чертеже.

Проведём прямую и отложим на ней отрезок, длина которого равна длине заданного отрезка RS.

2. На панели *Геометрия* выбрать объект *Вспомогательная прямая* и начертить горизонтальную прямую.
3. Скопировать отрезок RS командой [*Редактор—Копия—Указанием*] и поместить его на горизонтальную прямую щелчком мышью. Обозначить концы отрезка буквами A и B.

Построим две другие стороны треугольника, длины которых равны длинам заданных отрезков MN и PQ.

4. На панели *Геометрия* выбрать объект *Окружность*.

На *Панели свойств* щёлкнуть правой кнопкой мыши по полю *Радиус*, и в контекстном меню выбрать пункт *Между 2 точками*.

На чертеже навести курсор сначала на точку M, а затем на точку N.

Центр появившейся окружности заданного радиуса переместить в точку A.

5. Аналогично построить окружность, радиус которой равен длине отрезка PQ, и поместить ее центр в точку B.
6. Пересечение окружностей обозначить буквой C.
7. Построить отрезки AC и BC. Треугольник ABC построен (рис. 2.43).

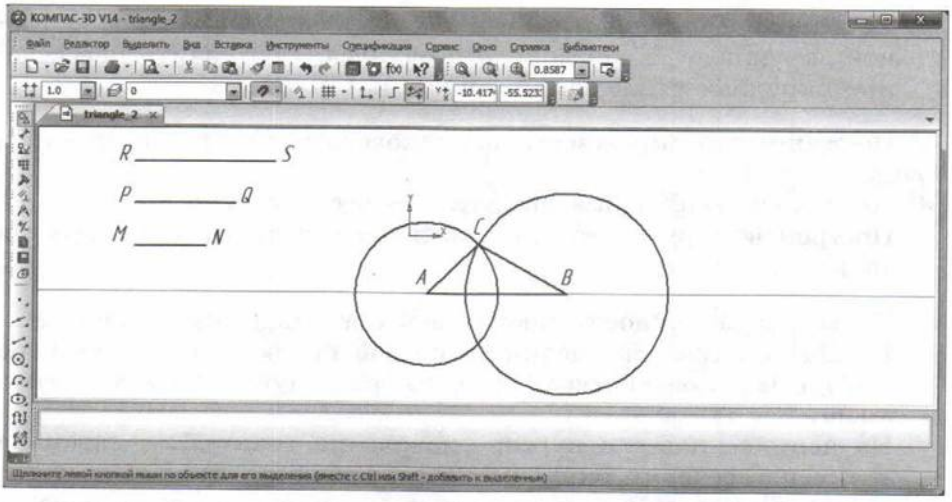


Рис. 2.43

2.8.4. Построение перпендикуляра к заданной прямой

Электронное приложение к главе 2.

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма:

1. Построить прямую a и точку M на ней.
2. На равных расстояниях от точки M построить на прямой точки A и B путём рисования окружности с центром в точке M.
3. Построить две окружности равного радиуса с центрами в точках A и B.
4. Через точки пересечения окружностей P и Q провести прямую. Данная прямая пройдёт через точку M и будет являться перпендикуляром к прямой a.

Начертим геометрические объекты, заданные в условии задачи: проведём прямую a и установим точку M на ней.

1. На панели *Геометрия* выбрать объект *Вспомогательная прямая* и начертить горизонтальную прямую. Ввести обозначение a.
2. На панели *Геометрия* выбрать объект *Точка* и установить точку M на прямой a.

На прямой на равных расстояниях от точки M построим точки A и B.

3. На панели *Геометрия* выбрать объект *Окружность* и построить окружность произвольного радиуса. Обозначить пересечение окружности с прямой точками A и B.

Построим две окружности одинакового радиуса с центрами в точках A и B.

4. На панели *Геометрия* выбрать объект *Окружность*.
   Построить окружность произвольного радиуса с центром в точке A.

Построим окружность такого радиуса с центром в точке B.

5. На *Панели свойств* щёлкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
   На чертеже навести курсор сначала на точку A, а затем на точку пересечения окружности с прямой a.
   Центр появившейся окружности заданного радиуса переместить в точку B.

Через точки пересечения окружностей P и Q проведём прямую.

6. Обозначить точки пересечения окружностей буквами P и Q. Соединить точки пересечения окружностей отрезком.
7. Данная прямая пройдёт через точку M и будет являться перпендикуляром к прямой a (рис. 2.44).

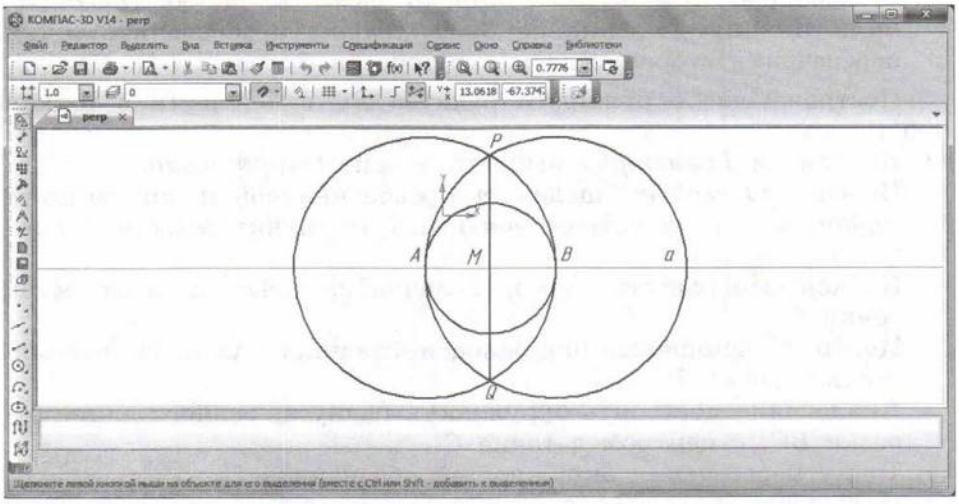


Рис. 2.44

2.8.5. Построение биссектрисы неразвёрнутого угла

Электронное приложение к главе 2.

Формальная модель. Построим формальную модель процесса геометрического построения, зафиксировав его в форме алгоритма:

1. Построить окружность произвольного радиуса с центром в вершине заданного угла A, которая пересечёт стороны угла в точках B и C.
2. Построить две окружности радиуса BC с центрами в точках B и C. Точку пересечения окружностей внутри угла обозначить буквой E.
3. Через вершину угла A и точку пересечения окружностей E провести прямую. Отрезок AE — биссектриса заданного угла.

Начертим геометрические объекты, заданные в условии задачи: два отрезка, исходящие из одной точки под произвольным неразвёрнутым углом.

1. Построить два отрезка, исходящие из одной точки. Обозначить угол на чертеже буквой A.

Построим окружность произвольного радиуса с центром в вершине заданного угла A.

2. Построить окружность произвольного радиуса с центром в точке A.
3. С помощью *Компактной панели* вызвать панель *Обозначения*. Щёлкнуть по кнопке *Ввод текста* и обозначить точки пересечения сторон угла и окружности буквами B и C.

Построим две окружности радиуса BC с центрами в точках B и C.

4. На панели *Геометрия* выбрать объект *Окружность*.
   На *Панели свойств* щёлкнуть правой кнопкой мыши по полю *Радиус* и в контекстном меню выбрать пункт *Между 2 точками*.
   На чертеже навести курсор сначала на точку B, а затем на точку C.
   Центр появившейся окружности заданного радиуса переместить в точку B.
   Аналогично построить окружность радиуса, равного длине отрезка BC, с центром в точке C.
5. Точку пересечения окружностей обозначить E.

Через вершину угла A и точку пересечения окружностей E проведём прямую.

6. Начертить отрезок с концами в точках A и E. Отрезок AE — биссектриса заданного угла (рис. 2.45).

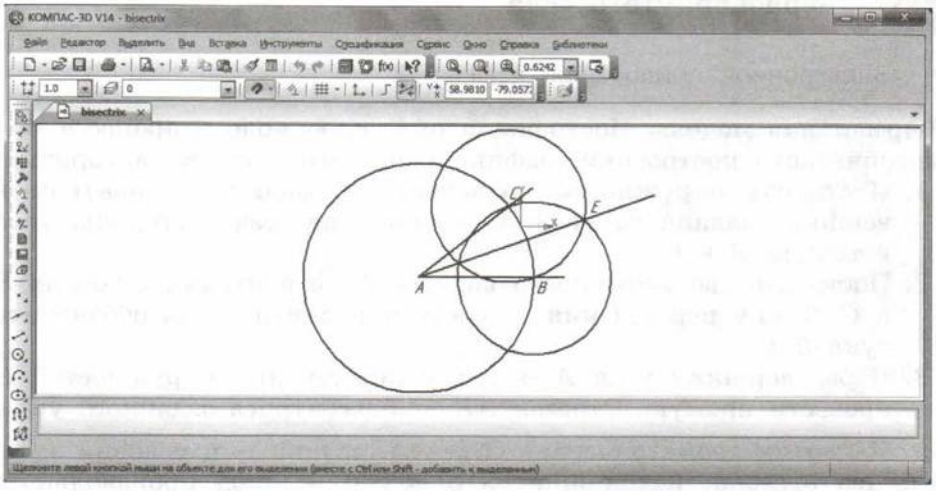


Рис. 2.45

2.3. Кодирование звуковой информации

Временна́я дискретизация звука. Звук представляет собой звуковую волну с непрерывно меняющейся амплитудой и частотой. Чем больше амплитуда сигнала, тем он громче для человека; чем больше частота сигнала, тем выше тон. Для того чтобы компьютер мог обрабатывать звук, непрерывный звуковой сигнал должен быть превращён в последовательность электрических импульсов (двоичных нулей и единиц).

Для записи аналогового звука и его преобразования в цифровую форму используется микрофон, подключённый к звуковой плате.

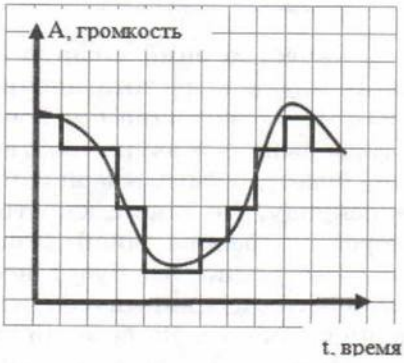


Рис. 2.46

В процессе кодирования непрерывного звукового сигнала производится его временна́я дискретизация. Непрерывная звуковая волна разбивается на отдельные маленькие временны́е участки (рис. 2.46), причём для каждого такого участка устанавливается определённая величина амплитуды. Непрерывная зависимость амплитуды сигнала от времени $A(t)$ заменяется на дискретную последовательность уровней громкости. На графике это выглядит как замена гладкой кривой на последовательность «ступенек».

Глубина кодирования. Каждой «ступеньке» присваивается определённое значение уровня громкости звука. Уровни громкости звука можно рассматривать как набор возможных состояний N, для кодирования которых необходимо определённое количество информации i, которое называется глубиной кодирования звука.

Глубина кодирования звука — это количество информации, которое необходимо для кодирования дискретных уровней громкости цифрового звука.

Если известна глубина кодирования, то количество уровней громкости цифрового звука можно рассчитать по формуле (1.1). Пусть глубина кодирования звука составляет 16 бит, тогда количество уровней громкости звука равно:

$$N = 2^i = 2^{16} = 65\,536.$$

Частота дискретизации. Качество цифрового звука зависит от количества измерений уровня громкости звука в единицу времени, т. е. от частоты дискретизации. Чем большее количество измерений производится за одну секунду (чем больше частота дискретизации), тем точнее «лесенка» цифрового звукового сигнала повторяет кривую аналогового сигнала.

Частота дискретизации звука — это количество измерений громкости звука за одну секунду.

Качество оцифрованного звука. Чем больше глубина кодирования звука и частота дискретизации звука, тем более качественным будет звучание оцифрованного звука. Самое низкое качество оцифрованного звука, соответствующее качеству телефонной связи, будет достигаться при частоте дискретизации 8000 измерений в секунду, глубине дискретизации 8 бит и записи одной звуковой дорожки (режим моно). Качество оцифрованного звука, соответствующее аудио-CD, будет достигаться при частоте дискретизации 48 000 измерений в секунду, глубине дискретизации 16 бит и записи двух звуковых дорожек (режим стерео).

Необходимо помнить, что чем выше качество цифрового звука, тем больше информационный объём высококачественного звукового файла. Можно оценить информационный объём цифрового стереозвукового файла длительностью звучания 1 секунда при среднем качестве звука (16 бит, 48 000 измерений в секунду). Для этого глубину кодирования необходимо умножить на количество измерений в 1 секунду и умножить на 2 (стереозвук):

$$16 \text{ бит} \cdot 48\,000 \cdot 2 = 1\,536\,000 \text{ бит} = 192\,000 \text{ байт} = 187{,}5 \text{ Кбайт}.$$

Звуковые редакторы. Звуковые редакторы позволяют не только записывать и воспроизводить звук, но и редактировать его. Оцифрованный звук представляется в звуковых редакторах в наглядной форме, поэтому операции копирования, перемещения и удаления частей звуковой дорожки можно легко осуществлять с помощью мыши. Кроме того, можно накладывать звуковые дорожки друг на друга (микшировать звуки) и применять различные акустические эффекты (эхо, воспроизведение в обратном направлении и др.).

Звуковые редакторы позволяют изменять качество цифрового звука и объём звукового файла путём изменения частоты дискретизации и глубины кодирования. Оцифрованный звук можно сохранять без сжатия в звуковых файлах в универсальном формате WAV, в формате со сжатием MP3.

Вопросы и задания

1. В чём состоит принцип двоичного кодирования звука?
2. От каких параметров зависит качество двоичного кодирования звука?

Практическая работа 2.9

Создание и редактирование оцифрованного звука

Задание

1. Записать звук.
2. Отредактировать и воспроизвести оцифрованный звук.

Варианты выполнения работы:

- запись цифрового звука с различными глубиной кодирования и частотой дискретизации;
- редактирование звукового файла различными способами.

Запись звука с использованием программы Звукозапись

Стандартное приложение операционной системы Windows Звукозапись играет роль цифрового диктофона и позволяет записывать цифровой звук.

1. В операционной системе Windows запустить приложение Звукозапись командой [*Пуск—Все программы—Стандартные—Звукозапись*].
2. Для начала записи оцифрованного звука щёлкнуть по кнопке *Начать запись* (рис. 2.47).

Рис. 2.47

3. Для остановки записи звукового файла щёлкнуть по кнопке *Остановить запись*.

После нажатия этой кнопки автоматически появится диалоговое окно *Сохранить как*, в котором необходимо выбрать место сохранения и имя для записанного звукового файла.

Редактирование и воспроизведение записанного цифрового звука с использованием программы Wave Editor

Wave Editor — это простой и быстрый редактор цифрового звука для Windows. В нём можно производить любые действия со звуковым файлом, не беспокоясь за его сохранность: файл изменяется только при его сохранении на диске.

1. В операционной системе Windows запустить Wave Editor [*Пуск—Все программы—Abyssmedia—Wave Editor—Wave Editor*].
2. Открыть файл, записанный в предыдущем задании, командой [*Файл—Открыть*] (рис. 2.48).

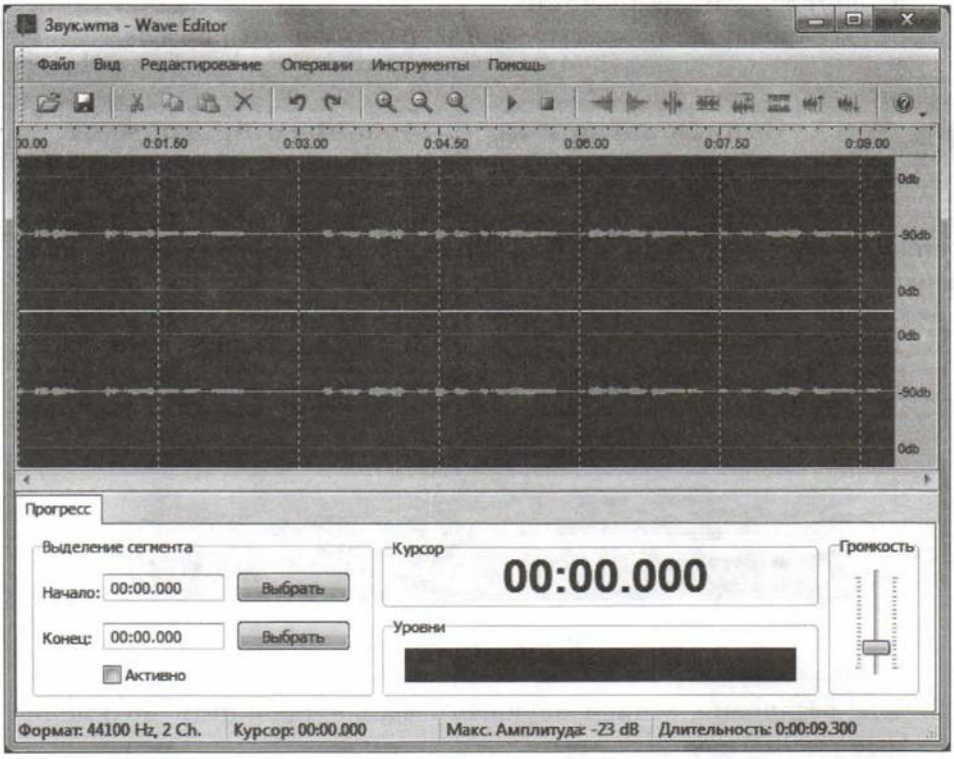


Рис. 2.48

Меню *Операции* позволяет увеличивать или уменьшать громкость и скорость воспроизведения, а также получать эффекты реверса и инверсии звукового файла, устанавливать плавное затухание звука.

3. С помощью мыши выделить фрагмент звукового файла и выполнить команду [*Операции—Усиление (усилить)*].
   В диалоговом окне *Усиление* установить процент усиления 150 (рис. 2.49).

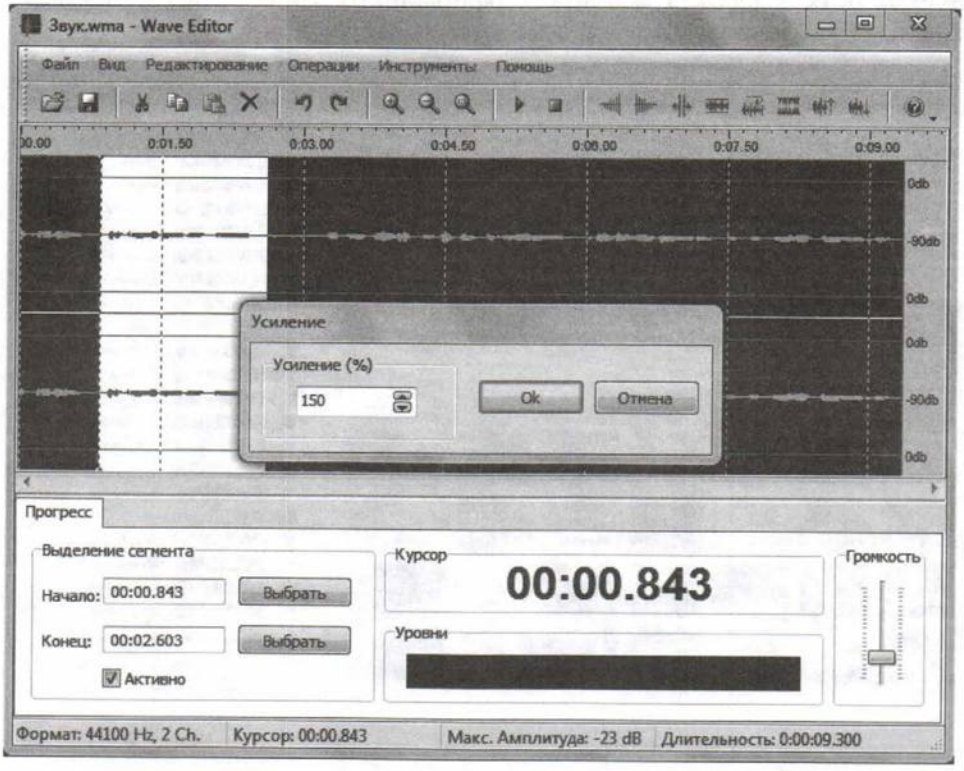


Рис. 2.49

4. Воспроизвести файл, выполнив команду [*Операции—Воспроизведение*].
5. Сохранить изменения в файле, выполнив команду [*Файл—Сохранить как...*].

2.4. Компьютерные презентации

Дизайн презентации. Создание презентации целесообразно начинать с разработки проекта, в котором необходимо определить примерное количество слайдов в презентации и содержание каждого слайда.

В зависимости от содержания презентации и категории её потенциальных зрителей необходимо выбрать дизайн презентации. Приложения, используемые для разработки презентаций, позволяют выбрать наиболее подходящий вариант дизайна с помощью нескольких десятков шаблонов оформления (рис. 2.50) и вариантов цветовых схем (рис. 2.51).

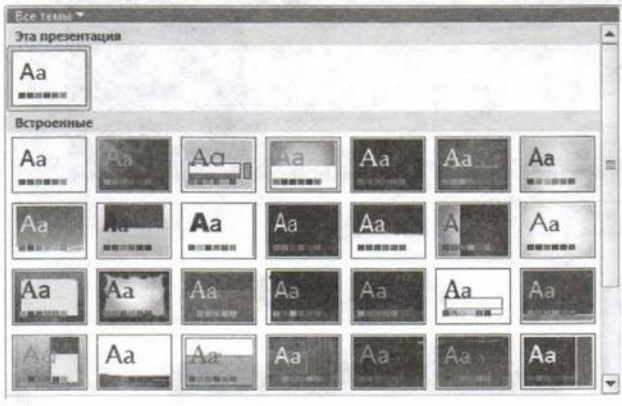


Рис. 2.50

Рис. 2.51

Каждый шаблон оформления предлагает свой вариант фона слайдов, а также тип и цвет используемых шрифтов.

Фон слайда может быть неоднородным, он может плавно переходить от одного оттенка цвета к другому, а также включать узоры и фоновые изображения.

Предлагаемые варианты цветовых схем позволяют выбрать наиболее подходящее соотношение между цветами фона, заголовка слайда и помещённого на слайд текста.

Макеты слайдов. Каждый раз при добавлении в презентацию нового слайда необходимо выбирать тип макета слайда. Макет слайда определяет, как на нём будут размещаться различные объекты: заголовок, текст, растровые изображения, векторные рисунки и др. (рис. 2.52).

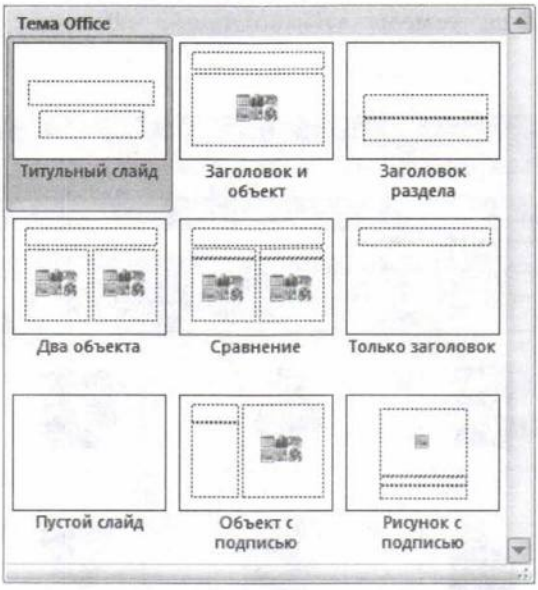


Рис. 2.52

Приложения, используемые для разработки презентаций, предлагают десятки различных вариантов разметки слайдов.

Самыми простыми являются макеты, содержащие пустые заготовки слайдов с заголовком и без него.

Текст на слайде может быть размещён в одну или две колонки.

Слайд целиком может быть занят растровым изображением или векторным рисунком.

На слайде могут быть размещены сразу несколько объектов различных типов: текст и изображение, рисунок и текст, изображение и рисунок и т. д.

Большинство макетов слайдов содержат заголовки.

Заполнение слайдов. Процедура заполнения слайда информацией одинакова для слайдов всех типов. Достаточно щёлкнуть мышью в выбранной области на макете слайда и ввести текст или нарисовать рисунок с использованием встроенного графического редактора.

Можно текст, растровое изображение или векторный рисунок заранее создать в текстовом или графическом редакторе и затем скопировать в выбранную область слайда.

Проще всего воспользоваться коллекцией рисунков, которая имеется как в Microsoft Office, так и в OpenOffice (рис. 2.53). Поиск нужного рисунка облегчается тем, что в коллекции они сгруппированы по темам: «Животные», «Люди», «Наука и техника» и др.

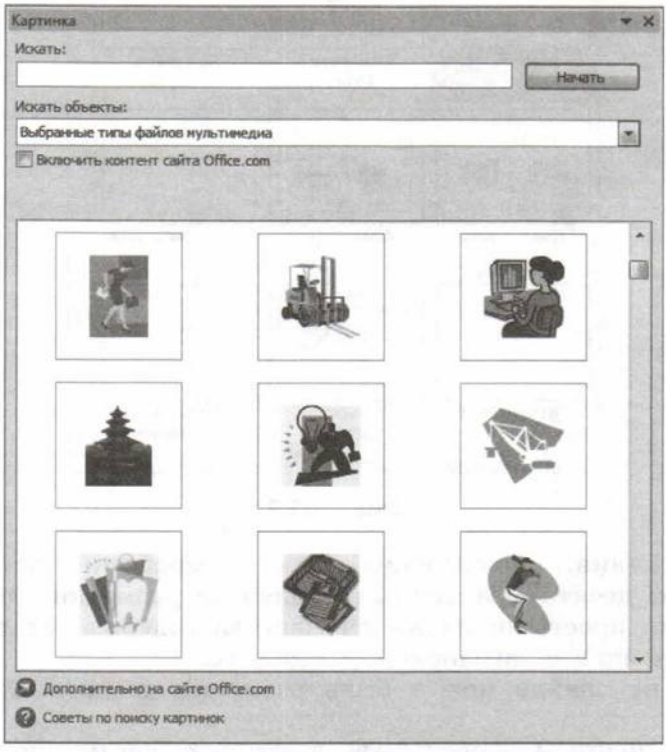


Рис. 2.53

С объектами, размещёнными на слайде, можно работать так же, как в векторном графическом редакторе. Можно перемещать выделенный объект по слайду, изменять его размеры, осуществлять группировку объектов, а также изменять взаимное расположение объектов по глубине при их наложении друг на друга.

Примеры приложений для разработки презентаций: Microsoft PowerPoint и OpenOffice Impress.

Общие рекомендации по размещению и оформлению материалов на слайде. Для создания качественной презентации необходимо соблюдать ряд требований. Рассмотрим основные из них.

1. Презентация должна быть разумно объёмной (10–12 слайдов).
2. Обязательно подписывайте свою презентацию, размещайте свои данные на первом (титульном) слайде. На слайде обязательно должны быть представлены: название работы; фамилия, имя автора; номер и название школы, номер класса, где учится автор работы.
3. Все слайды должны быть оформлены в едином стиле (одинаковый шаблон оформления слайдов). Будьте осторожны с пёстрыми и яркими фонами. Фон не должен напрягать глаза и мешать работе с объектами на слайде.
4. Количество используемых цветов на слайде: 3–5; количество типов шрифтов — не более трёх (не забывайте о стандартных цветовых схемах).
5. Цвет шрифта и цвет фона должны контрастировать (текст должен хорошо читаться), но не «резать глаза».
6. Каждый слайд должен излагать одну мысль.
7. Изложение материала должно быть кратким (оптимально не более 6–8 слов на одном слайде).
8. Иллюстрации должны быть в одном стиле, одного размера и формата. Они призваны дополнить текстовую информацию или передать её в более наглядном виде. Иллюстрации рекомендуется сопровождать пояснительным текстом.
9. Анимация присутствует только в тех местах, где она уместна и усиливать эффект восприятия текстовой части информации. Она не должна отвлекать внимание от содержания информации на слайде. Анимацию к объектам на титульном слайде и к заголовкам использовать не принято.
10. Звуковой фон должен соответствовать единой концепции и усиливать эффект восприятия текстовой части информации.
11. Соблюдайте авторские права! Обязательно размещайте в презентации ссылки на источники использованных материалов.

Анимация и звук в процессе смены слайдов. Анимационные эффекты и воспроизведение звука при демонстрации презентации могут быть использованы в процессе смены слайдов.

Программы разработки презентаций позволяют выбрать один из типов анимационных эффектов, который будет реализовываться в процессе перехода слайдов (рис. 2.54). Например, при использовании эффекта *Наплыв* следующий слайд будет появляться постепенно, наезжая на предыдущий слайд.

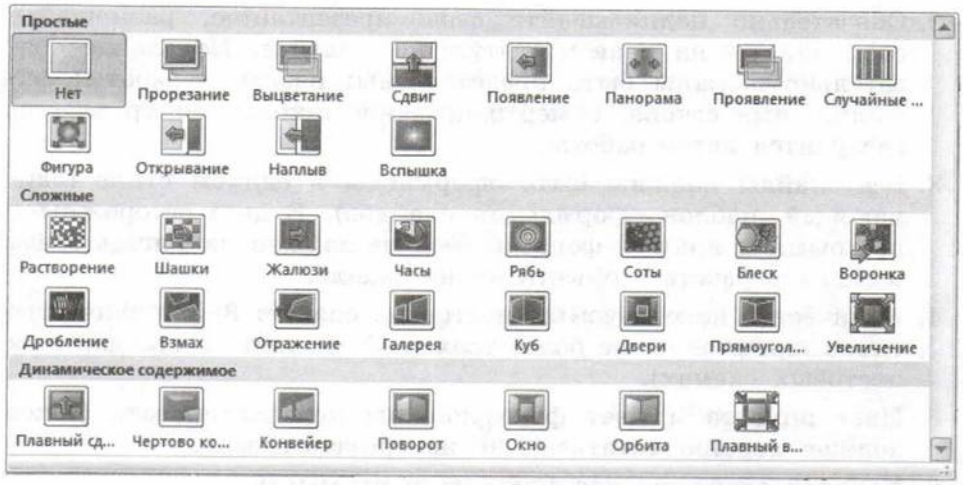


Рис. 2.54

В процессе разработки презентации можно выбрать звук, которым будет сопровождаться переход слайдов. Программы для разработки презентаций предлагают довольно широкий набор готовых звуков (аплодисменты, колокольчики, пишущая машинка и др.), однако можно использовать и любой другой звук, найдя соответствующий звуковой файл на локальном компьютере или в Интернете. Можно также записать любой звук с использованием звукового редактора.

Анимация и звук в процессе появления объектов на слайде. Любой объект, размещённый на слайде, можно заставить возникнуть на экране необычно: постепенно проявиться, вылететь сбоку, развернуться до заданного размера, уменьшиться, вспыхнуть, вращаться и т. д.

Появление объекта на слайде может сопровождаться различными звуками. Звук можно выбрать из набора, имеющегося в программе для разработки презентаций (барабан, буря оваций, касса и др.), можно найти подходящий звуковой файл или записать звук самостоятельно.

Демонстрация презентации. После запуска демонстрации презентации слайды будут последовательно по порядку их номеров выводиться на экран. Существуют разные возможности для перехода от одного слайда к другому, следующему за ним: нажатие клавиши *Enter*, щелчок левой кнопкой мыши, нажатие стрелок ← → на клавиатуре и т. д. Для перемещения по слайдам презентации вперёд или назад можно пользоваться клавишами клавиатуры *PageUp* или *PageDown*.

Можно сделать презентацию **интерактивной**, т. е. предоставить пользователю возможность в процессе демонстрации презентации изменять последовательность предъявления слайдов.

Организовать любые переходы между слайдами можно двумя способами: с помощью **гиперссылок** или **управляющих кнопок**.

Гиперссылка позволяет осуществлять переход с любого слайда на любой другой с использованием указателя ссылки и адреса перехода. На исходном слайде размещается указатель ссылки: фрагмент текста, который выделяется цветом и подчёркиванием, или графическое изображение. В адресе перехода указывается номер и название слайда, на который должен быть произведён переход. Активизация указателя ссылки (например, щелчком мышью) вызывает переход на слайд, заданный в адресе перехода.

На слайд можно поместить управляющие кнопки *Вперёд*, *Назад*, *В начало*, *В конец* и др. (рис. 2.55). Если в процессе демонстрации презентации активизировать кнопку (щёлкнуть на ней мышью), то произойдёт переход на указанный слайд.

Рис. 2.55

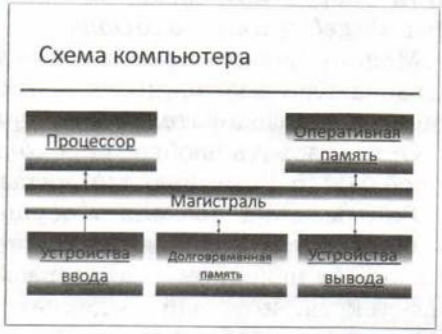
Вопросы и задания

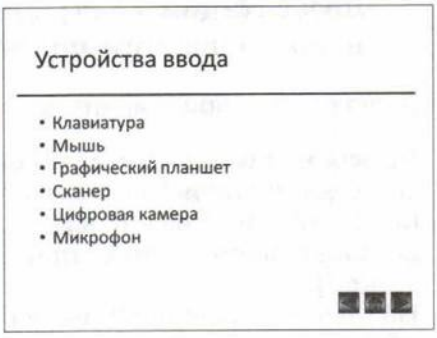
1. Какие параметры выбираются одновременно для всех слайдов презентации? Индивидуально для каждого слайда презентации?
2. Как можно использовать анимационные и звуковые эффекты в презентации?
3. В чём состоит различие между использованием гиперссылок и управляющих кнопок при реализации интерактивной презентации?

Практическая работа 2.10

Разработка мультимедийной интерактивной презентации «Устройство компьютера»

Задание. Разработать презентацию «Устройство компьютера», включающую шесть слайдов, которые должны быть созданы с использованием различных типов макетов слайдов:

№ слайда	Название и содержание слайда	Тип макета слайда	Примерный вид слайда
1	Схема компьютера. Векторный рисунок схемы компьютера	Заголовок и графика	
2	Процессор. Изображение процессора и поясняющий текст	Заголовок, графика и текст	
3	Оперативная память. Поясняющий текст и изображения модулей оперативной памяти	Заголовок, текст и два объекта	

№ слайда	Название и содержание слайда	Тип макета слайда	Примерный вид слайда
4	Устройства ввода. Перечень устройств ввода	Заголовок и список	
5	Долговременная память. Изображения устройств долговременной памяти	Заголовок и таблица с изображениями	
6	Устройства вывода. Перечень устройств ввода и их изображения	Заголовок, список и таблица	

Варианты выполнения работы:

- выбор различного дизайна презентации и различных типов макетов для отдельных слайдов.

Разработка мультимедийной интерактивной презентации «Устройство компьютера» с использованием приложения Microsoft PowerPoint

Электронное приложение к главе 2.

Выберем дизайн для слайдов презентации.

1. В операционной системе Windows запустить приложение Microsoft PowerPoint и ввести команду [*Файл—Создать*].
   В появившемся окне приложения выбрать [*Новая презентация*].
   На ленте перейти на вкладку *Дизайн* и выбрать *Тему* презентации.
   На вкладке *Дизайн* ленты в раскрывающемся списке *Цвета* выбрать цветовую схему для слайдов презентации.

Создадим заготовки для слайдов презентации.

2. На ленте перейти на вкладку *Главная* и с помощью раскрывающегося списка *Макеты* выбрать для каждого слайда разметку (макет).

Введём содержание слайдов.

3. Поместить на слайды заголовки, текст и изображения.

Сделаем презентацию интерактивной.

4. Обеспечить возможность переходов со слайда 1 «Схема компьютера» на слайды 2, 3, 4, 5 и 6 с помощью гиперссылок.
   Для этого выделить текст в качестве указателя гиперссылки и ввести команду *Гиперссылка* на вкладке *Вставка*.
   В появившемся диалоговом окне *Изменение гиперссылки* (рис. 2.56) указать нужный слайд в качестве адреса ссылки.
5. Обеспечить возможность обратных переходов со слайдов 2, 3, 4, 5 и 6 на слайд 1 «Схема компьютера» с помощью управляющих кнопок.
   Для этого на ленте перейти на вкладку *Главная*, открыть раскрывающийся список *Фигуры*, в нём перейти к *Управляющим кнопкам* и выбрать тип кнопки.
   В диалоговом окне *Настройка действия* указать нужный слайд в качестве адреса ссылки (рис. 2.57).

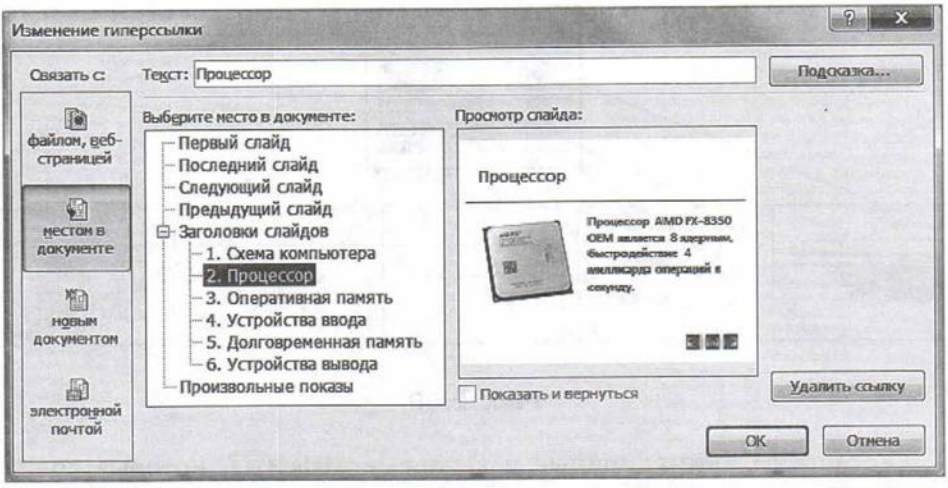


Рис. 2.56

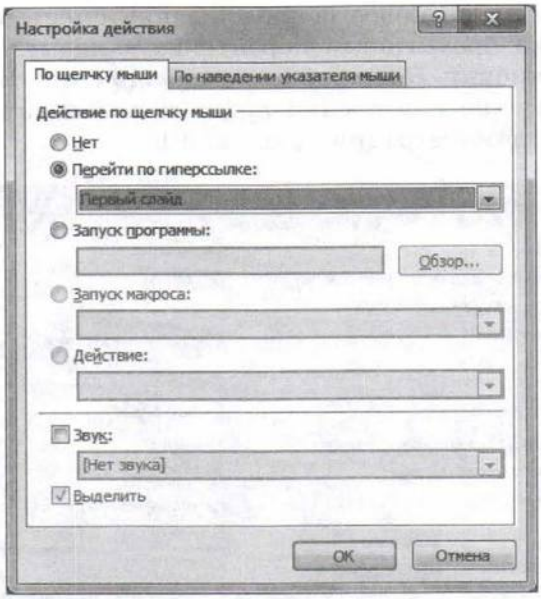


Рис. 2.57

В результате получим интерактивную презентацию, в которой последовательность показа слайдов управляется пользователем (рис. 2.58).

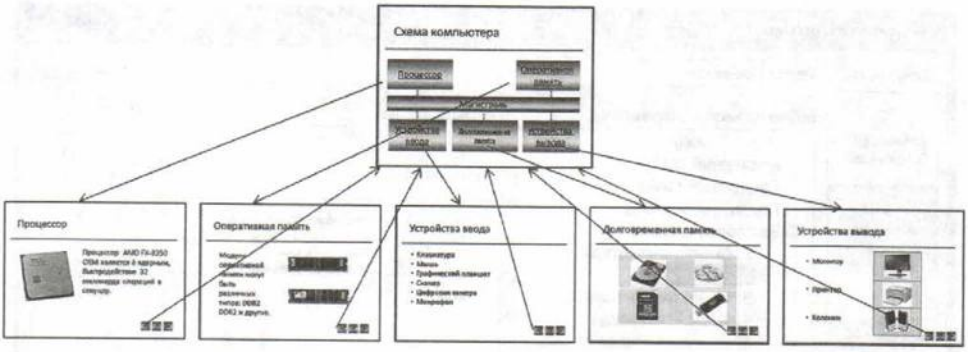


Рис. 2.58

Установим анимационные и звуковые эффекты, которые должны происходить при смене слайдов.

6. На ленте перейти на вкладку *Переходы* и выбрать тип перехода между слайдами, событие (по щелчку или автоматически по времени) и звуковое сопровождение смены слайдов.
   Для запуска презентации перейти во вкладку *Показ слайдов* и нажать кнопку *Настройка демонстрации*. В появившемся диалоговом окне *Настройка презентации* задать необходимые параметры демонстрации (рис. 2.59).

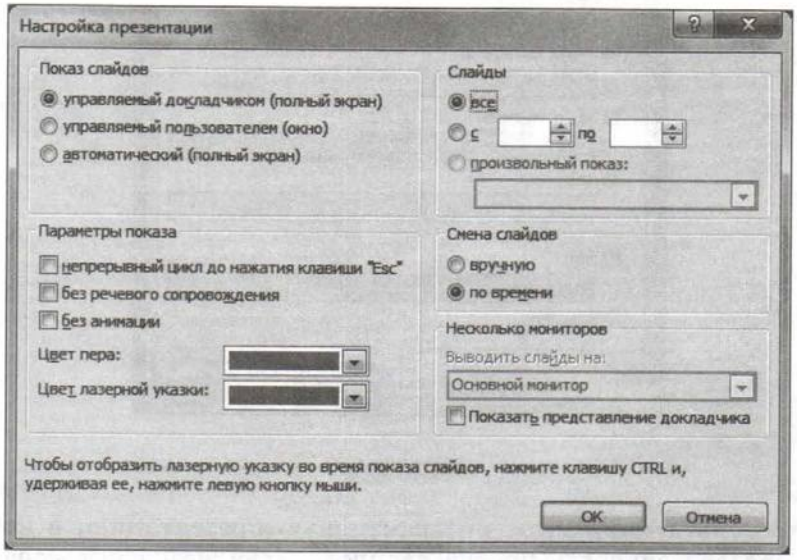


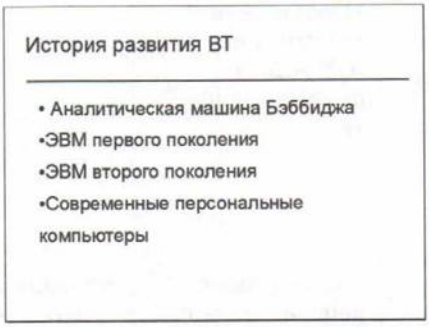
Рис. 2.59

Практическая работа 2.11

Разработка презентации «История развития вычислительной техники»

Задание 1. Разработать с использованием мастера презентацию «История развития вычислительной техники», включающую пять слайдов. Подобрать дизайн презентации и тип макета для каждого слайда, а также анимационные и звуковые эффекты, реализующиеся при появлении объектов на слайдах и при смене слайдов.

Задание 2. Доработать презентацию (в рамках коллективного проекта).

№ слайда	Название и содержание слайда	Тип макета слайда	Примерный вид слайда
1	История развития ВТ. Перечень основных этапов развития ВТ	Заголовок и список	
2	Аналитическая машина Бэббиджа. Портрет Бэббиджа и изображение его Аналитической машины	Заголовок, два объекта	

№ слайда	Название и содержание слайда	Тип макета слайда	Примерный вид слайда
3	ЭВМ первого поколения. Поясняющий текст и изображение МЭСМ	Заголовок, текст и графика	ЭВМ первого поколения Первое поколение (1945-1954) - компьютеры на электронных лампах. Большинство машин первого поколения были экспериментальными устройствами и строились с целью проверки тех или иных теоретических положений. 
4	ЭВМ второго поколения. Изображение транзистора и БЭСМ-6, поясняющий текст	Заголовок, два объекта над текстом	ЭВМ второго поколения   Во втором поколении (1955-1964) вместо электронных ламп использовались транзисторы, а в качестве устройств памяти стали применяться магнитные сердечники и магнитные барабаны - далекие предки современных жестких дисков. Все это позволило резко уменьшить габариты и стоимость компьютеров.
5	Современный персональный компьютер. Изображения современных персональных компьютеров	Заголовок, три объекта	Современные персональные компьютеры 

Варианты выполнения работы:

- выбор различного дизайна презентации и различных типов макетов для отдельных слайдов.

Разработка презентации «История развития вычислительной техники» в приложении OpenOffice Impress

Электронное приложение к главе 2.

1. Запустить приложение OpenOffice Impress, предварительно установив его с веб-ресурса.
2. Запустится *Мастер создания презентаций*, и появится его первая панель *Мастер презентаций* (рис. 2.60).
   Выбрать вариант создания новой презентации *Пустая презентация*.
   Щёлкнуть по кнопке *Далее*.

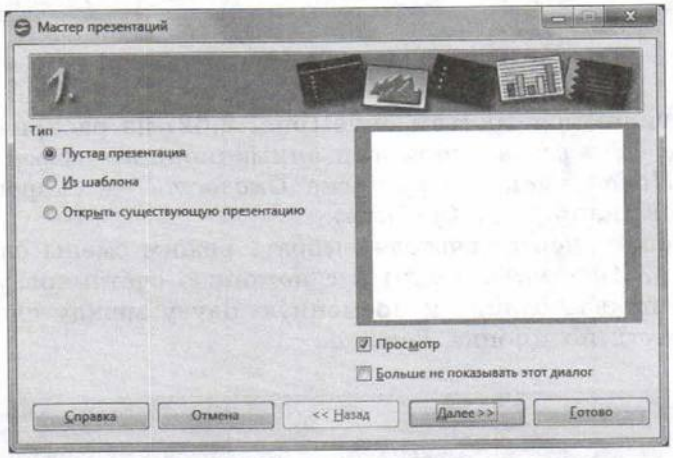


Рис. 2.60

Мастер поможет подобрать дизайн презентации из существующих шаблонов.

3. В появившемся втором окне (рис. 2.61) из раскрывающегося списка выбрать вариант дизайна создаваемой презентации, например *<Оригинал>*.
   Щёлкнуть по кнопке *Далее*.

Мастер поможет подобрать анимационный эффект, который будет происходить в процессе демонстрации презентации при смене одного слайда другим. Можно также выбрать событие, по которому будет происходить смена слайдов (по щелчку или автоматически по времени).

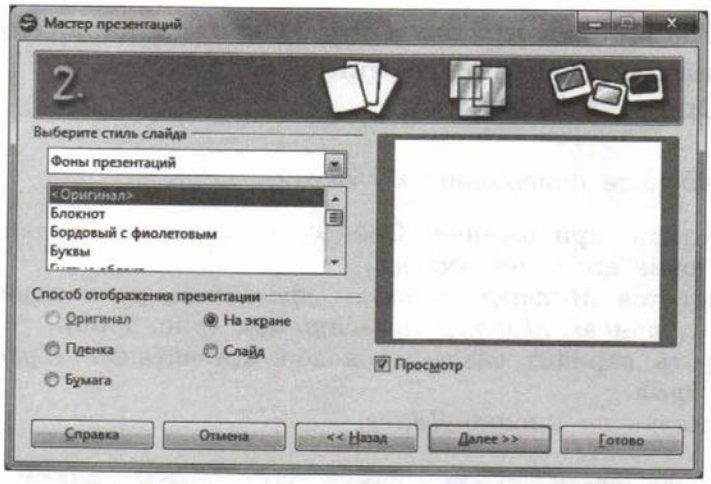


Рис. 2.61

4. В появившемся третьем окне (рис. 2.62) из раскрывающегося списка *Эффект* выбрать тип анимационного эффекта (например, *Появление*), а из списка *Скорость* — скорость смены слайдов (например, *Средняя*).
С помощью переключателя выбрать режим смены слайдов (например, *Автоматически*) и с помощью счётчиков установить время показа слайда и временну́ю паузу между слайдами.
Щёлкнуть по кнопке *Готово*.

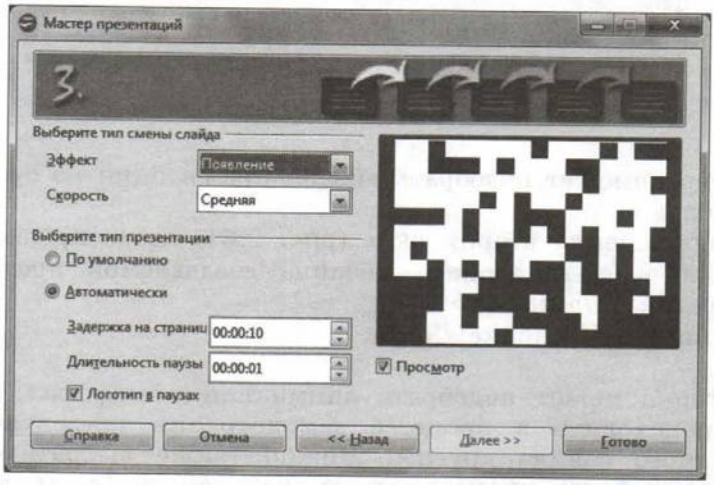


Рис. 2.62

Подберём для каждого слайда нужный тип макета.

5. В окне приложения ввести команду [*Формат—Макеты слайдов...*]. На панели задач появится диалоговое окно выбора типа макета слайдов *Макеты* (рис. 2.63).
   Выбрать для первого слайда будущей презентации вариант макета *Заголовок, текст*.
   Вставить в презентацию оставшиеся четыре слайда командой [*Вставка—Слайд...*] и подобрать для них требуемые типы макета.

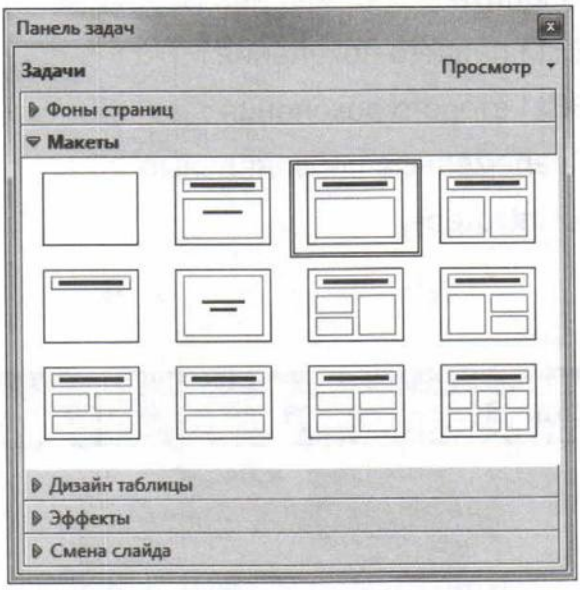


Рис. 2.63

В каждый слайд введём заголовки, тексты и изображения.

6. В окне приложения выбрать слайд и на вкладке *Режим рисования* произвести ввод текста или вставку изображения (рис. 2.64).

Подберём анимационные эффекты появления объектов на слайде.

7. Ввести команду [*Демонстрация—Эффекты...*].
   В появившемся диалоговом окне *Эффекты* (рис. 2.65) выделить в списке объект.
   Выбрать тип эффекта, событие наступления эффекта и направление эффекта.

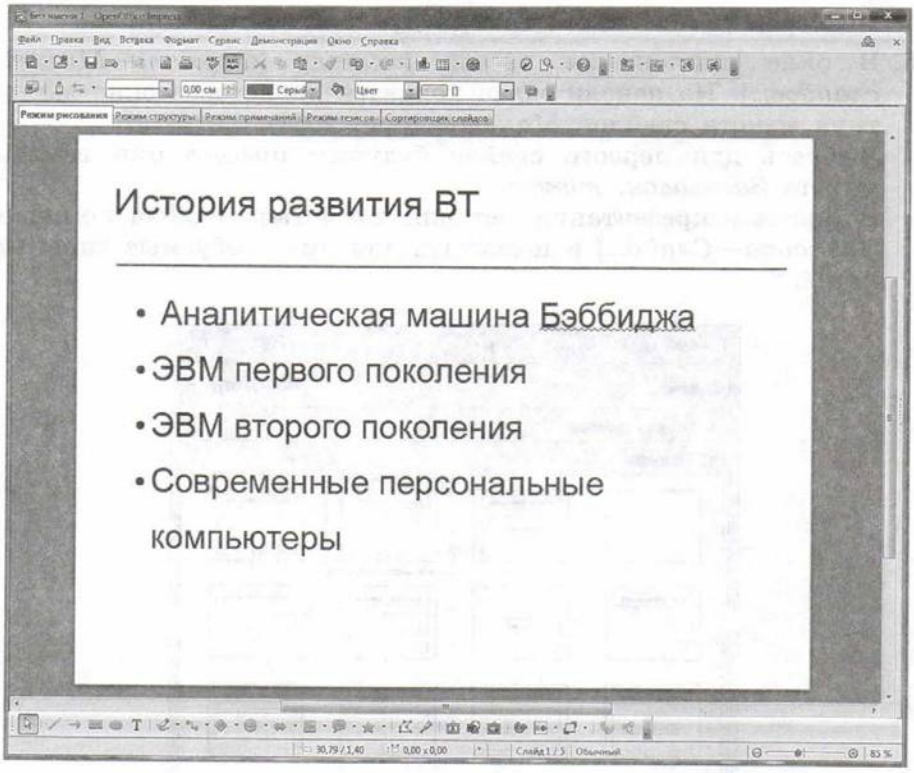


Рис. 2.64

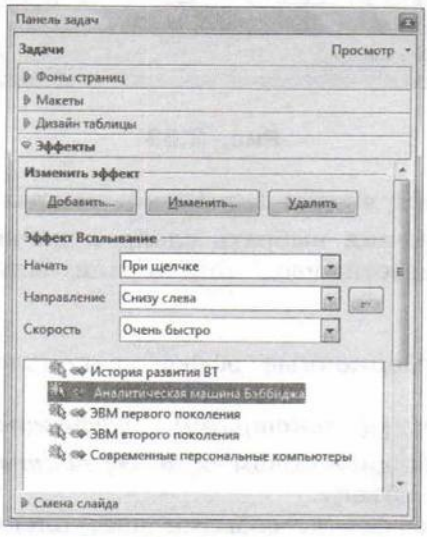


Рис. 2.65

Добавим или изменим ранее установленные эффекты, реализующиеся при переходах слайдов.

8. В окне приложения перейти на вкладку *Сортировщик слайдов* (рис. 2.66).
   Для того чтобы можно было применить настройки сразу ко всем слайдам презентации, выделить их командой [*Правка—Выделить все*].

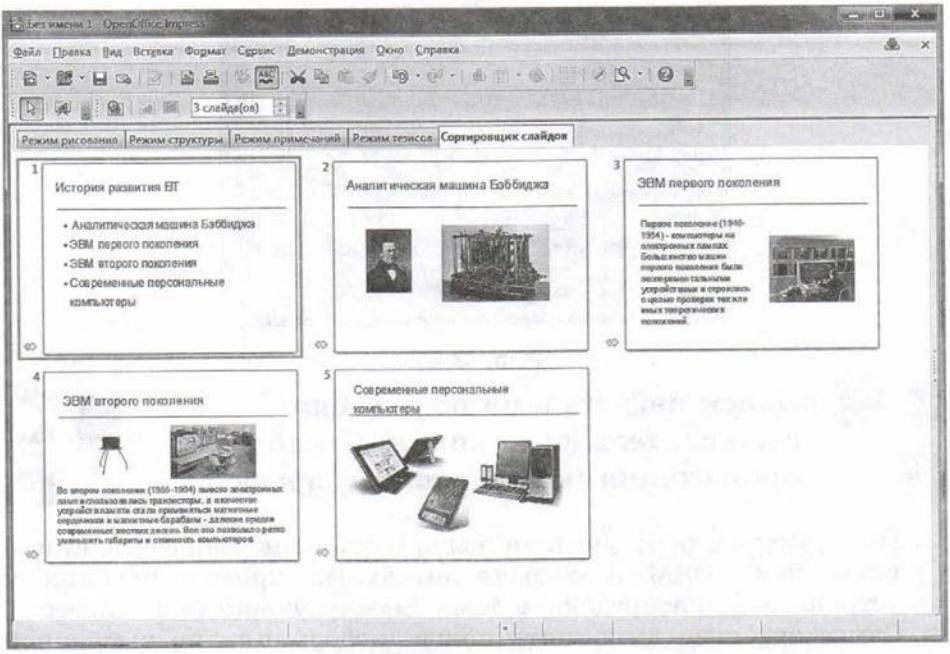


Рис. 2.66

9. Ввести команду [*Демонстрация—Смена слайда...*].
   В появившемся диалоговом окне *Смена слайда* (рис. 2.67) выбрать эффект анимации, событие (по щелчку или автоматически по времени) и звуковое сопровождение смены слайдов.

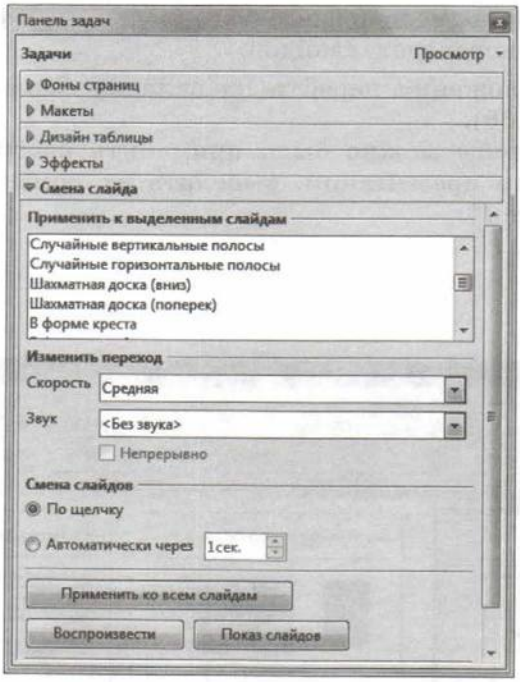


Рис. 2.67

Поиск информации об истории вычислительной техники. Доработка презентации (коллективный проект)

1. Под руководством учителя выполнить поиск информации о поколениях ЭВМ и о вычислительных приспособлениях и устройствах. Распределить темы между учащимися класса.
2. Самостоятельно выполнить поиск информации по выбранной теме.
3. Доработать презентацию об истории развития вычислительной техники.

2.5. Кодирование и обработка числовой информации

2.5.1. Системы счисления. Представление числовой информации

Для записи информации о количестве объектов используются числа. Числа записываются с использованием особых знаковых

систем, которые называются **системами счисления**. Алфавит системы счисления состоит из символов, которые называются цифрами.

Система счисления — это знаковая система, в которой числа записываются по определённым правилам с помощью знаков некоторого алфавита, называемых цифрами.

Все системы счисления делятся на две большие группы: позиционные и непозиционные. В позиционных системах счисления количественное значение цифры зависит от её положения в числе, а в непозиционных — не зависит.

Непозиционные системы счисления. Примером непозиционной системы, которая сохранилась до наших дней, может служить римская система счисления, которая начала применяться более двух с половиной тысяч лет назад в Древнем Риме. В основе римской системы счисления лежат знаки I (один палец) для числа 1, V (раскрытая ладонь) для числа 5, X (две сложенные ладони) для числа 10, а для обозначения чисел 50, 100, 500 и 1000 используются латинские буквы L, C, D и M.

В римской системе счисления количественное значение цифры не зависит от её положения в числе. Например, в римском числе XXX (30) цифра X встречается трижды и в каждом случае обозначает одну и ту же величину — число 10, три раза по 10 в сумме дают 30.

Чтобы записать число, римляне разлагали его на сумму тысяч, полутысяч, сотен, полусотен, десятков, пятков, единиц. Например, десятичное число 28 представляется следующим образом:

XXVIII = 10 + 10 + 5 + 1 + 1 + 1
(два десятка, пяток, три единицы).

Для записи чисел римляне использовали не только сложение, но и вычитание. При этом применялось следующее правило: каждый меньший знак, поставленный справа от большего, прибавляется к его значению, а каждый меньший знак, поставленный слева от большего, вычитается из него.

Например, IX обозначает 9, XI обозначает 11.

Десятичное число 99 имеет следующее представление:

XCIX = –10 + 100 – 1 + 10.

Римскими цифрами пользовались очень долго. Ещё 200 лет назад в деловых бумагах числа должны были обозначаться римскими цифрами (считалось, что обычные арабские цифры легко подделать). Римская система счисления сегодня используется в основном для наименования знаменательных дат, веков, томов, разделов и глав в книгах.

Позиционные системы счисления. Каждая позиционная система счисления имеет определённый **алфавит** цифр и **основание.** Основание системы счисления равно количеству цифр (знаков) в её алфавите.

В позиционных системах счисления количественное значение цифры зависит от её позиции в числе. Позиция цифры в числе называется **разрядом.** Разряды числа возрастают справа налево, от младших разрядов к старшим. Основание показывает, во сколько раз изменяется значение цифры при перемещении её в соседний разряд.

В настоящее время наиболее распространёнными позиционными системами счисления являются десятичная и двоичная. Десятичная система счисления имеет алфавит цифр, который состоит из десяти всем известных, так называемых арабских, цифр {0,1,2,3,4,5,6,7,8,9}, а двоичная имеет две цифры {0,1} (табл. 2.1).

Таблица 2.1

Позиционные системы счисления

Система счисления	Основание	Алфавит цифр
Десятичная	10	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Двоичная	2	0, 1

В системах счисления с основаниями, бóльшими 10, в качестве недостающих цифр используют буквы латинского алфавита. Например, в шестнадцатеричной системе основание равно 16 и алфавит состоит из шестнадцати цифр {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}, причём первые десять цифр имеют общепринятое обозначение, а для записи остальных цифр со значениями 10, 11, 12, 13, 14, 15 используются первые шесть букв латинского алфавита.

Десятичная система счисления. В десятичной системе счисления цифра в крайней справа позиции обозначает единицы, цифра, смещённая на одну позицию влево, обозначает десятки, ещё левее — сотни, затем — тысячи и т. д. Рассмотрим в качестве примера десятичное число 333. Цифра 3 встречается в числе трижды, причём самая правая обозначает три единицы, вторая справа — три десятка и, наконец, третья — три сотни.

Выше десятичное число 333 было записано в привычной для нас **свёрнутой форме.** Мы настолько привыкли к такой форме записи, что уже не замечаем, как в уме умножаем цифры числа на различные степени числа 10, которое является основанием десятичной системы счисления.

В **развёрнутой форме записи числа** умножение цифр числа на основание производится в явной форме. Так, в развёрнутой форме запись числа 333 в десятичной системе будет выглядеть следующим образом:

$$333_{10} = 3 \cdot 10^2 + 3 \cdot 10^1 + 3 \cdot 10^0.$$

Для записи десятичных дробей используются разряды с отрицательными значениями степеней основания. Например, число 333,33 в развёрнутой форме будет записываться следующим образом:

$$333{,}33_{10} = 3 \cdot 10^2 + 3 \cdot 10^1 + 3 \cdot 10^0 + 3 \cdot 10^{-1} + 3 \cdot 10^{-2}.$$

Запишем шестнадцатеричное число в свёрнутой и развёрнутой формах:

$$\mathrm{ABCDEF}_{16} = \mathrm{A} \cdot 16^5 + \mathrm{B} \cdot 16^4 + \mathrm{C} \cdot 16^3 + \mathrm{D} \cdot 16^2 + \mathrm{E} \cdot 16^1 + \mathrm{F} \cdot 16^0 =$$
$$= 10 \cdot 16^5 + 11 \cdot 16^4 + 12 \cdot 16^3 + 13 \cdot 16^2 + 14 \cdot 16^1 + 15 \cdot 16^0.$$

Числа в позиционных системах счисления записываются **в виде многочлена: суммы степеней основания**, в качестве коэффициентов которых выступают цифры данного числа.

Умножение или деление десятичного числа на 10 (величину основания) приводит к перемещению запятой, отделяющей целую часть от дробной, на один разряд вправо или влево. Например:

$$333{,}33_{10} \cdot 10 = 3333{,}3_{10},$$
$$333{,}33_{10} : 10 = 33{,}333_{10}.$$

Двоичная система счисления. Числа в двоичной системе в развёрнутой форме записываются в виде суммы ряда степеней основания 2 с коэффициентами, в качестве которых выступают цифры 0 или 1.

Пример: развёрнутая запись двоичного числа выглядит следующим образом:

$$A_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2},$$

а в свёрнутой форме:

$$A_2 = 110{,}01_2.$$

Умножение или деление двоичного числа на 2 (величину основания) приводит к перемещению запятой, отделяющей целую часть от дробной на один разряд вправо или влево. Например:

$$110{,}01_2 \cdot 2 = 1100{,}1_2,$$
$$110{,}01_2 : 2 = 11{,}001_2.$$

Арифметические операции во всех позиционных системах счисления выполняются по одним и тем же хорошо известным правилам.

Сложение. Рассмотрим сложение чисел в двоичной системе счисления. В его основе лежит таблица сложения одноразрядных двоичных чисел:

$$\begin{aligned} 0 + 0 &= 0 \\ 0 + 1 &= 1 \\ 1 + 0 &= 1 \\ 1 + 1 &= 10 \end{aligned}$$

Важно обратить внимание на то, что при сложении двух единиц происходит переполнение разряда и производится перенос в старший разряд. Переполнение разряда наступает тогда, когда величина числа в нём становится равной или большей основания системы счисления. Для двоичной системы счисления эта величина равна 2.

Сложение многоразрядных двоичных чисел производится в соответствии с вышеприведённой таблицей сложения с учётом возможных переносов из младших разрядов в старшие. В качестве примера сложим в столбик двоичные числа 110_2 и 11_2.

$$\begin{array}{r} 110_2 \\ +\ 11_2 \\ \hline 1001_2 \end{array}$$

Проверим правильность вычислений сложением в десятичной системе счисления. Переведём двоичные числа в десятичную систему счисления и затем их сложим.

$$110_2 = 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 6_{10};$$
$$11_2 = 1 \cdot 2^1 + 1 \cdot 2^0 = 3_{10};$$
$$6_{10} + 3_{10} = 9_{10}.$$

Теперь переведём результат двоичного сложения в десятичное число.

$$1001_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 9_{10}.$$

Сравним результаты. Сложение выполнено правильно.

Вычитание. Рассмотрим вычитание двоичных чисел. В его основе лежит таблица вычитания одноразрядных двоичных чисел. При вычитании из меньшего числа (0) большего (1) производится заём из старшего разряда. В таблице вычитания заём обозначен цифрой 1 с чертой над ней.

$$\begin{aligned} 0 - 0 &= \overline{1}0 \\ 0 - 1 &= 11 \\ 1 - 0 &= 1 \\ 1 - 1 &= 0 \end{aligned}$$

Вычитание многоразрядных двоичных чисел производится в соответствии с вышеприведенной таблицей вычитания с учётом возможных заёмов в старших разрядах. В качестве примера произведём вычитание двоичных чисел 110_2 и 11_2.

$$\begin{array}{r} 110_2 \\ -\ 11_2 \\ \hline 11_2 \end{array}$$

Умножение. В основе умножения лежит таблица умножения одноразрядных двоичных чисел:

$$0 \cdot 0 = 0$$
$$0 \cdot 1 = 0$$
$$1 \cdot 0 = 0$$
$$1 \cdot 1 = 1$$

Умножение многоразрядных двоичных чисел производится в соответствии с вышеприведённой таблицей умножения по обычной схеме, применяемой в десятичной системе счисления, с последовательным умножением множимого на очередную цифру множителя. В качестве примера произведём умножение двоичных чисел 110_2 и 11_2.

$$\begin{array}{r} \times\ 110_2 \\ \quad 11_2 \\ \hline \quad 110 \\ \quad 110 \\ \hline 10010_2 \end{array}$$

Деление. Операция деления выполняется по алгоритму, подобному алгоритму выполнения операции деления в десятичной системе счисления. В качестве примера произведём деление двоичного числа 110_2 на 11_2.

$$\begin{array}{r|l} 110_2 & 11_2 \\ \hline -11 & 10_2 \\ \hline 0 & \end{array}$$

Вопросы и задания

1. Чем отличаются позиционные системы счисления от непозиционных?
2. Каково основание десятичной системы счисления? Двоичной системы счисления?
3. Какие цифры входят в алфавит десятичной системы счисления? Двоичной системы счисления?
4. На какую величину в позиционных системах счисления различаются цифры соседних разрядов числа?

Практическая работа 2.12

Перевод чисел из одной системы счисления в другую с помощью калькулятора

Задание. Перевести:

- целое двоичное число в десятичную систему счисления;
- целое десятичное число в двоичную систему счисления;
- десятичное число в непозиционную римскую систему счисления.

Варианты выполнения работы:

- перевод различных чисел.

Перевод чисел из двоичной системы счисления в десятичную, из десятичной в двоичную и римскую с помощью электронного калькулятора NumLock Calculator

Для перевода чисел из двоичной системы счисления в десятичную необходимо установить в качестве исходной системы счисления двоичную, а в качестве конечной — десятичную.

1. В операционной системе Windows запустить электронный калькулятор NumLock Calculator командой [*Все программы—NumLock Calculator—NumLock Calculator*].
   В меню калькулятора выбрать вид *Простой*.
2. С помощью меню ввести команды [*Формат чисел в выражении—Двоичный*] и [*Формат результата—Десятичный*].
   Ввести двоичное число и нажать кнопку =. Результат для числа 11111111_2 показан на рис. 2.68.

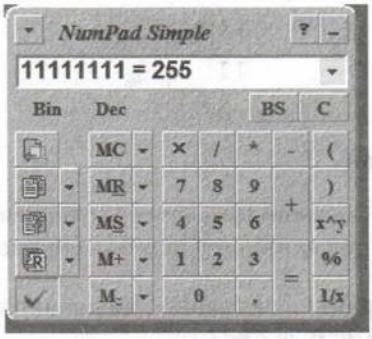


Рис. 2.68

Для перевода чисел из десятичной системы счисления в двоичную необходимо установить в качестве исходной системы счисления десятичную, а в качестве конечной — двоичную.

3. С помощью меню ввести команды [*Формат чисел в выражении—Десятичный*] и [*Формат результата—Двоичный*].
Ввести десятичное число и нажать кнопку =. Результат для числа 255_{10} показан на рис. 2.69.

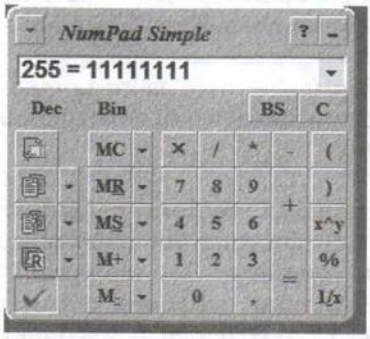


Рис. 2.69

Для перевода чисел из десятичной системы счисления в непозиционную римскую систему необходимо установить в качестве исходной системы счисления десятичную, а в качестве конечной — римскую систему.

4. С помощью меню ввести команды [*Формат чисел в выражении—Десятичный*] и [*Формат результата—Римский*].
Ввести десятичное число и нажать кнопку =. Результат для числа 255_{10} показан на рис. 2.70.

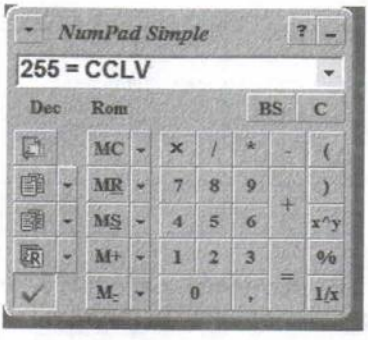


Рис. 2.70

2.5.2. Электронные таблицы

Электронные таблицы позволяют обрабатывать большие массивы числовых данных. В отличие от таблиц на бумаге электронные таблицы обеспечивают проведение динамических вычислений, т. е. пересчёт по формулам при вводе новых чисел. В математике с помощью электронных таблиц можно представить функцию в числовой форме и построить её график. В физике можно обработать результаты лабораторной работы. В географии или истории можно представить статистические данные в форме диаграммы.

Электронные таблицы — это работающее в диалоговом режиме приложение, хранящее и обрабатывающее данные в прямоугольных таблицах.

Весь файл электронных таблиц называется **рабочей книгой**, которая в свою очередь состоит из **рабочих листов**.

Столбцы, строки, ячейки. Электронная таблица состоит из столбцов и строк. Заголовки **столбцов** обозначаются буквами или сочетаниями букв (A, C, AB и т. п.), заголовки **строк** — числами (1, 2, 3 и далее).

На пересечении столбца и строки находится **ячейка**, которая имеет индивидуальный адрес. **Адрес** ячейки электронной таблицы составляется из заголовка столбца и заголовка строки, например A1, B5, E3. Ячейка, с которой производятся какие-то действия, выделяется рамкой и называется **активной**. Так, в таблице 2.2 активной является ячейка C3.

Таблица 2.2

Электронные таблицы (столбцы, строки, ячейки)

	A	B	C	D	E
1					
2					
3					
4					
5					

Диапазон ячеек. В процессе работы с электронными таблицами достаточно часто требуется выделить несколько ячеек. Выделенные ячейки образуют диапазон, который задаётся адресами ячеек верхнего левого и нижнего правого углов диапазона, раз-

делёнными двоеточием. Можно выделить несколько ячеек в столбце (диапазон A2:A4), несколько ячеек в строке (диапазон C1:E1) или прямоугольный диапазон (диапазон C3:E4) (табл. 2.3).

Таблица 2.3

Диапазоны ячеек в столбце, строке и прямоугольный диапазон

	A	B	C	D	E
1					
2					
3					
4					
5					

Основные типы и форматы данных. В работе с электронными таблицами можно выделить три основных типа данных: числа, текст и формулы.

Числа. Для представления чисел могут использоваться несколько различных типов форматов (**числовой**, **экспоненциальный**, **дробный** и **процентный**). Существуют специальные форматы для хранения **дат** (например, 25.05.2007) и **времени** (например, 13:30:55), а также **финансовый** и **денежный** форматы (например, 1500,00р.), которые используются при проведении бухгалтерских расчётов.

По умолчанию для представления чисел электронные таблицы используют числовой формат, который отображает два десятичных знака числа после запятой (например, 195,20).

Экспоненциальный формат применяется, если число, содержащее большое количество разрядов, не умещается в ячейке. В этом случае разряды числа представляются с помощью положительных или отрицательных степеней числа 10. Например, числа 2 000 000 и 0,000002, представленные в экспоненциальном формате как $2 \cdot 10^6$ и $2 \cdot 10^{-6}$, будут записаны в ячейке электронных таблиц в виде 2,00E+06 и 2,00E–06.

По умолчанию числа выравниваются в ячейке по правому краю. Это объясняется тем, что при размещении чисел друг под другом (в столбце таблицы) удобно иметь выравнивание по разрядам (единицы под единицами, десятки под десятками и т. д.).

Текст. Текстом в электронных таблицах является последовательность символов, состоящая из букв, цифр и пробелов, напри-

мер текстом может являться последовательность цифр «2004». По умолчанию текст выравнивается в ячейке по левому краю. Это объясняется традиционным способом письма (слева направо).

Формулы. Формула должна начинаться со знака равенства и может включать в себя числа, имена ячеек, функции и знаки математических операций.

Например, формула =A1+B1 обеспечивает сложение чисел, хранящихся в ячейках A1 и B1, а формула =A1*5 — умножение числа, хранящегося в ячейке A1, на 5. При изменении исходных значений, входящих в формулу, результат пересчитывается немедленно.

Относительные, абсолютные и смешанные ссылки. В формулах могут использоваться ссылки на адреса ячеек. Существуют два основных типа ссылок: относительные и абсолютные. Различия между относительными и абсолютными ссылками проявляются при копировании формулы из активной ячейки в другие ячейки.

Относительные ссылки. При перемещении или копировании формулы из активной ячейки относительные ссылки автоматически изменяются в зависимости от положения ячейки, в которую скопирована формула. При перемещении положения ячейки на одну строку в формуле изменяются на единицу номера строк, а при перемещении на один столбец смещаются на одну букву имени столбцов.

Абсолютные ссылки. Абсолютные ссылки в формулах используются для указания фиксированного адреса ячейки. При перемещении или копировании формулы абсолютные ссылки не изменяются. В абсолютных ссылках перед неизменяемым значением адреса ячейки ставится знак $ (например, $A$1).

Смешанные ссылки. В формуле можно использовать смешанные ссылки, в которых координата столбца относительная, а строки — абсолютная (например, A$1), или наоборот, координата столбца абсолютная, а строки относительная (например, $B1).

Вопросы и задания

1. Как обозначаются столбцы и строки электронной таблицы? Как задаётся имя ячейки?
2. Какие типы данных и в каких форматах могут обрабатываться в электронных таблицах?
3. Как изменяется при копировании в ячейку, расположенную в соседнем столбце и строке, формула, содержащая относительные ссылки? Абсолютные ссылки? Смешанные ссылки?

Практическая работа 2.13

Относительные, абсолютные и смешанные ссылки в электронных таблицах

Задание 1. В электронных таблицах осуществить копирование формулы, содержащей относительные ссылки, из активной ячейки C1 в ячейку D2 и в ячейку E3. Какие формулы будут в этих ячейках?

	A	B	C	D	E
1			=A1*B1		
2					
3					

Задание 2. В электронных таблицах осуществить копирование формулы, содержащей абсолютные ссылки, из активной ячейки C1 в ячейку D2 и в ячейку E3. Какие формулы будут в этих ячейках?

	A	B	C	D	E
1			=$A$1*$B$1		
2					
3					

Задание 3. В электронных таблицах осуществить копирование формулы, содержащей смешанные ссылки, из активной ячейки C1 в ячейку D2 и в ячейку E3. Какие формулы будут в этих ячейках?

	A	B	C	D	E
1			=A$1*$B1		
2					
3					

Варианты выполнения работы:

- ввод различных формул в разные ячейки и их копирование в различные ячейки.

Копирование в электронных таблицах формулы, содержащей относительные ссылки

Электронное приложение к главе 2.

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel или OpenOffice Calc. Или: в операционной системе Linux запустить электронные таблицы OpenOffice Calc. Присвоить листу *Лист1* имя *Относительные ссылки*.
2. В электронных таблицах *Microsoft Excel* для отображения в ячейках не чисел, а формул ввести команду [*Файл—Параметры*] и в появившемся диалоговом окне *Параметры Excel* на вкладке *Дополнительно* в разделе *Показать параметры для следующего листа* установить флажок *Показывать формулы, а не их значения* (рис. 2.71).

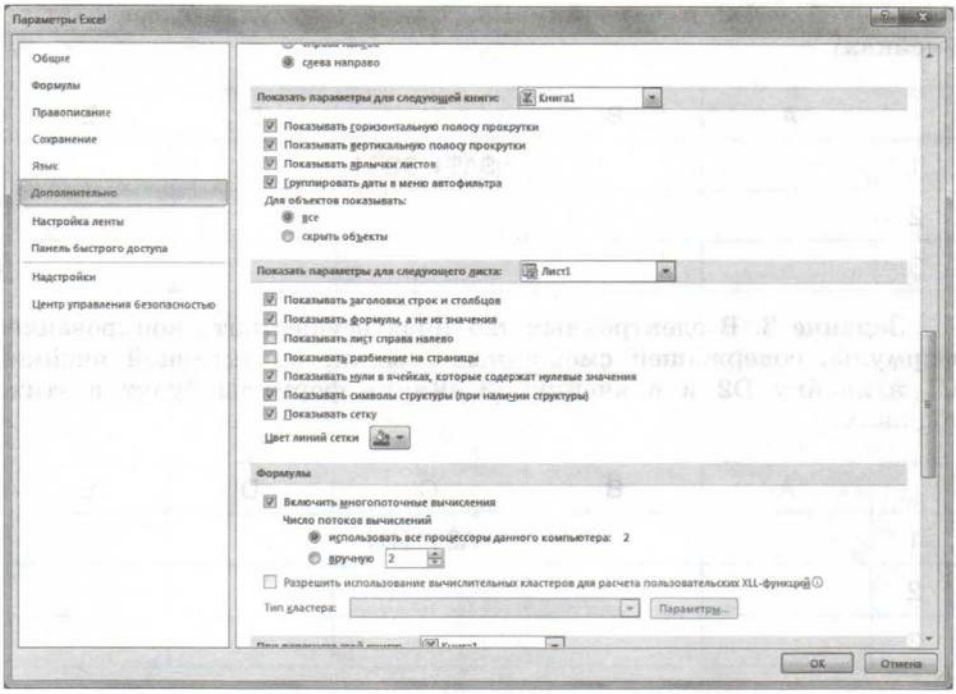


Рис. 2.71

В электронных таблицах OpenOffice Calc для отображения в ячейках не чисел, а формул ввести команду [*Сервис—Параметры...*] и в появившемся диалоговом окне *Параметры—OpenOffice* выбрать раздел *OpenOffice Calc*, вкладку *Вид* и установить флажок *Формулы* (рис. 2.72).

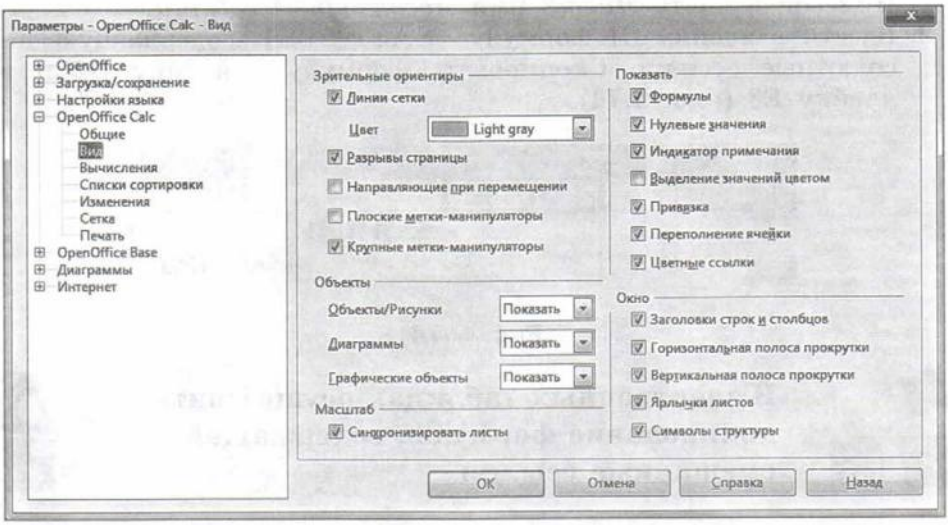


Рис. 2.72

3. Ввести в ячейку C1 формулу =A1*B1, содержащую относительные ссылки. Скопировать формулу в ячейку D2 и в ячейку E3 (рис. 2.73).

	A	B	C	D	E
1			=A1*B1		
2				=B2*C2	
3					=C3*D3

Рис. 2.73

Копирование в электронных таблицах формулы, содержащей абсолютные ссылки

Электронное приложение к главе 2.

1. Присвоить листу *Лист2* имя *Абсолютные ссылки*.
2. Ввести в ячейку C1 формулу =$A$1*$B$1, содержащую абсолютные ссылки. Скопировать формулу в ячейку D2 и в ячейку E3 (рис. 2.74).

	A	B	C	D	E
1			=$A$1*$B$1		
2				=$A$1*$B$1	
3					=$A$1*$B$1

Рис. 2.74

В электронных таблицах осуществить копирование формулы, содержащей смешанные ссылки

Электронное приложение к главе 2.

1. Присвоить листу *Лист3* имя *Смешанные ссылки*.
2. Ввести в ячейку C1 формулу =A$1*$B1, содержащую смешанные ссылки. Скопировать формулу в ячейку D2 и в ячейку E3 (рис. 2.75).

	A	B	C	D	E
1			=A$1*$B1		
2				=B$1*$B2	
3					=C$1*$B3

Рис. 2.75

2.5.3. Построение диаграмм и графиков

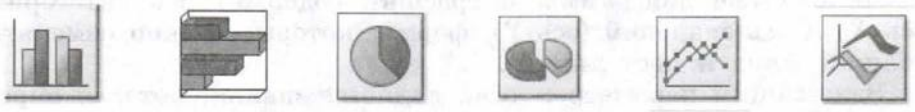
Электронные таблицы позволяют визуализировать данные, размещённые на рабочем листе, в виде диаграммы. Диаграммы наглядно отображают зависимости между данными, что облегчает восприятие и помогает при анализе и сравнении данных.

Типы диаграмм. Диаграммы могут быть различных типов и представлять данные в различной форме (рис. 2.76). Для каждого набора данных важно правильно подобрать тип создаваемой диаграммы.

Для наглядного сравнения величин используются **линейчатые и столбчатые диаграммы (гистограммы)**. В них высота столбца пропорциональна значению величины. Эти диаграммы могут быть плоскими или объёмными, причём столбцы могут быть расположены как вертикально (гистограмма), так и горизонтально (линейчатая диаграмма). Например, с помощью линейчатой диаграммы можно наглядно представить данные о численности населения различных стран мира.

Для отображения величин частей некоторого целого применяется **круговая диаграмма**, в которой величина кругового сектора пропорциональна значению части целого. Круговые диаграммы могут быть плоскими или объёмными, причём секторы могут быть раздвинуты (разрезанная круговая диаграмма). Круговая диаграмма позволяет, например, наглядно показать доли стоимости отдельных устройств компьютера в его общей цене.

Для построения графиков функций и отображения изменения величин в зависимости от времени используются диаграммы типа **график**. На плоских графиках маркерами отображаются значения числовой величины, которые соединяются между собой плавными линиями. Объёмные графики представляют изменение величины с помощью цветной трёхмерной фигуры.



Гистограмма с группировкой | Объёмная линейчатая с группировкой | Круговая | Объёмная разрезанная круговая | График с маркерами | Объёмный график

Рис. 2.76

Диапазон исходных данных: ряды данных и категории. При создании диаграммы в электронных таблицах прежде всего необходимо выделить диапазон ячеек, содержащий исходные данные для её построения. Диаграммы связаны с исходными данными на рабочем листе и обновляются при обновлении этих данных.

Выделенный диапазон исходных данных включает в себя как ряды данных, так и категории.

Ряд данных — это множество значений, которые необходимо отобразить на диаграмме. На линейчатой диаграмме значения

ряда данных отображаются с помощью столбцов, на круговой — с помощью секторов, на графике — точками, имеющими заданные координаты y.

Категории задают положение значений ряда данных на диаграмме. На линейчатой диаграмме категории являются «подписями» под столбцами, на круговой диаграмме — названиями секторов, а на графике категории используются для обозначения делений на оси X. Если диаграмма отображает изменение величины во времени, то категории всегда являются интервалами времени: дни, месяцы, годы и т. д.

Ряды данных и категории могут размещаться как в столбцах, так и в строках электронной таблицы.

Оформление диаграммы. Диаграммы могут располагаться как на отдельных листах, так и на листах с соответствующими данными (внедрённые диаграммы). **Область диаграммы** кроме обязательной области построения диаграммы может содержать названия оси категорий и оси значений, заголовок диаграммы и легенду.

Область построения диаграммы является основным объектом в области диаграммы, так как именно в ней производится графическое отображение данных. В линейчатых диаграммах можно изменять цвет столбцов, в круговых — цвет секторов, в графиках — форму, размер и цвет маркеров и соединяющих их линий.

Линейчатые диаграммы и графики содержат **ось категорий** (ось X) и **ось значений** (ось Y), формат которых можно изменять (толщину, вид и цвет линий).

Важнейшим параметром осей является **шкала**, которая определяет минимальное и максимальное значения шкалы, а также цену основных и промежуточных делений. Рядом с делениями шкалы по оси категорий размещаются названия категорий, а рядом с делениями шкалы по оси значений — значения ряда данных. В круговых диаграммах названия категорий и значения ряда данных отображаются рядом с секторами диаграммы.

Для более точного определения величины столбцов линейчатой диаграммы и положений маркеров графика можно использовать горизонтальные и вертикальные линии **сетки**. Основные линии сетки продолжают основные деления шкалы, а промежуточные линии — промежуточные деления шкалы.

Название диаграммы и **названия осей** можно перемещать и изменять их размеры, а также можно изменять тип шрифта, его размер и цвет.

Легенда содержит названия категорий и показывает используемый для их отображения цвет столбцов (в линейчатых диаграммах), цвет секторов (в круговых диаграммах), форму и цвет маркеров и линий (на графиках). Можно перемещать легенду и изменять её размеры, а также можно изменять тип используемого шрифта, его размер и цвет.

Вопросы и задания

1. Как отображаются на диаграммах ряды данных и категории?
2. Каковы основные элементы области диаграммы и их назначение?

Практическая работа 2.14

Построение диаграмм различных типов

Задание 1. В электронных таблицах построить на листе с данными линейчатую диаграмму с вертикальными столбцами (гистограмму), позволяющую отобразить рост количества серверов Интернета по годам.

Рост Интернета

Годы	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015
Количество серверов (млн)	233	318	395	433	542	625	732	818	1024	1268	1305	1445

Задание 2. В электронных таблицах построить круговую диаграмму, позволяющую наглядно представить долю серверов Интернета, зарегистрированных в разных доменах.

Распределение серверов Интернета по доменам

Домены	Административные домены	Япония	Италия	Германия	Франция	Нидерланды	Австралия	Россия	Другие страны
Количество серверов (млн)	253,0	30,8	13,8	13,1	10,3	9,0	8,5	2,4	92,1

Задание 3. В электронных таблицах построить графики кубической функции $y = x^3$ и линейной функции $y = 2 \cdot x$.

Числовое представление кубической функции $y = x^3$ и линейной функции $y = 2 \cdot x$

	A	B	C	D	E	F	G	H	I	J
1	x	–4	–3	–2	–1	0	1	2	3	4
2	$y = x^3$	–64	–27	–8	–1	0	1	8	27	64
3	$y = 2 \cdot x$	–8	–6	–4	–2	0	2	4	6	8

Построение линейчатой диаграммы в электронных таблицах

Электронное приложение к главе 2.

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel.
   Присвоить листу имя «*Рост Интернета*».

На листе с данными построим линейчатую диаграмму с вертикальными столбцами (гистограмму) без легенды.

2. Вставить в электронные таблицы данные из таблицы «Рост Интернета», оставив ячейку A1 незаполненной.
3. Выделить диапазон ячеек A1:M2, содержащий исходные данные (ячейка A1 не содержит текстовых данных).
4. На ленте на вкладке *Вставка* выбрать тип диаграммы *Гистограмма*.
   В меню раскрывающегося списка *Гистограмма* (рис. 2.77) выбрать тип *Гистограмма с группировкой*.

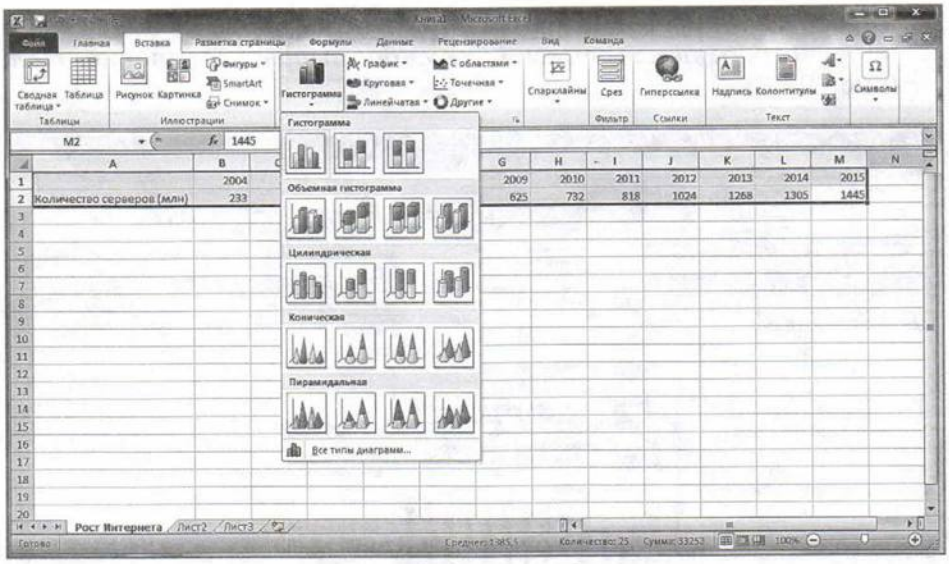


Рис. 2.77

5. На вкладке *Конструктор* в меню раскрывающегося списка *Макеты диаграмм* (рис. 2.78) выбрать нужный макет диаграммы — *Макет11*.

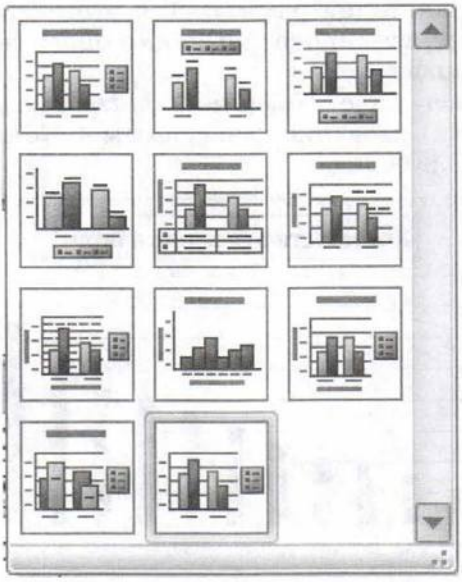


Рис. 2.78

6. Убрать *Легенду* с диаграммы. Для этого на вкладке *Макет* в раскрывающемся списке *Легенда* выбрать значение *Нет* (рис. 2.79).

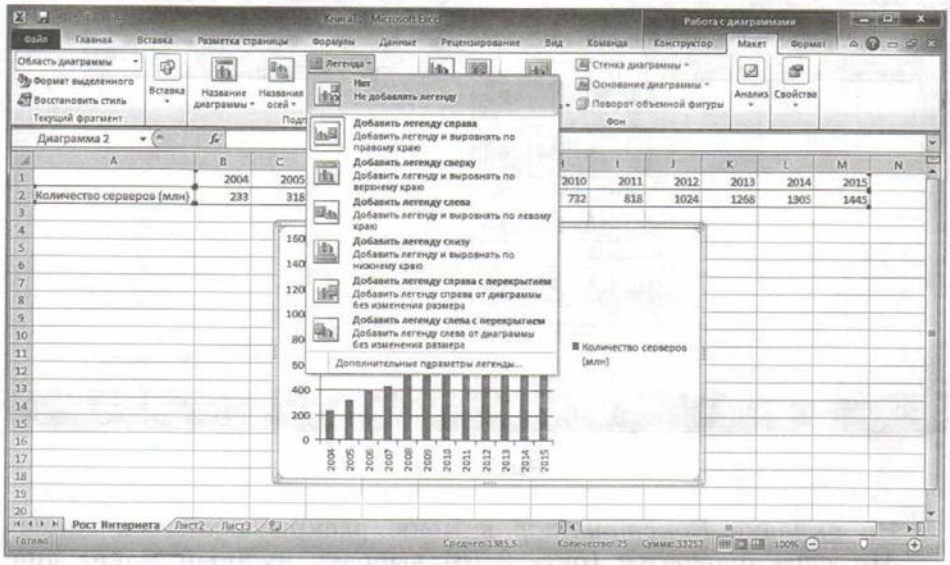


Рис. 2.79

7. Добавить название диаграммы. Для этого на вкладке *Макет* в раскрывающемся списке *Название диаграммы* выбрать значение *Над диаграммой*.

   В появившемся поле *Название диаграммы* в области диаграммы ввести название диаграммы «Количество серверов (млн)» (рис. 2.80).

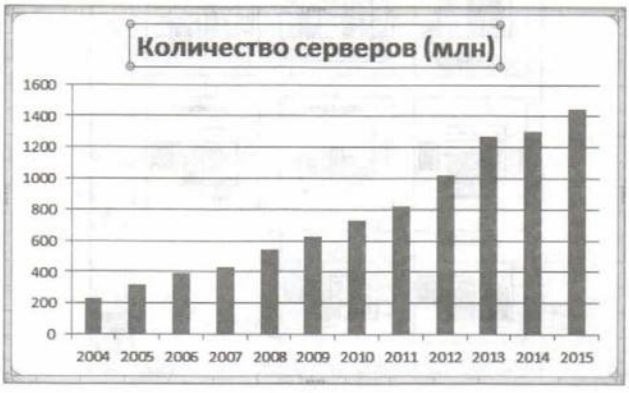


Рис. 2.80

8. Добавить название горизонтальной оси. Для этого на вкладке *Макет* в раскрывающемся списке *Название осей* последовательно выбрать *Название основной горизонтальной оси*, значение *Название под осью* (рис. 2.81).

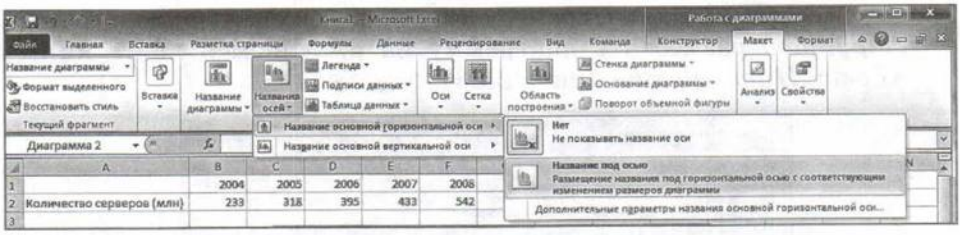


Рис. 2.81

9. В появившемся поле *Название оси* в области диаграммы ввести название оси «Годы».
10. Аналогично добавить название вертикальной оси. Для этого на вкладке *Макет* в раскрывающемся списке *Название осей* последовательно выбрать *Название основной вертикальной оси*, значение *Повёрнутое название* (рис. 2.82).

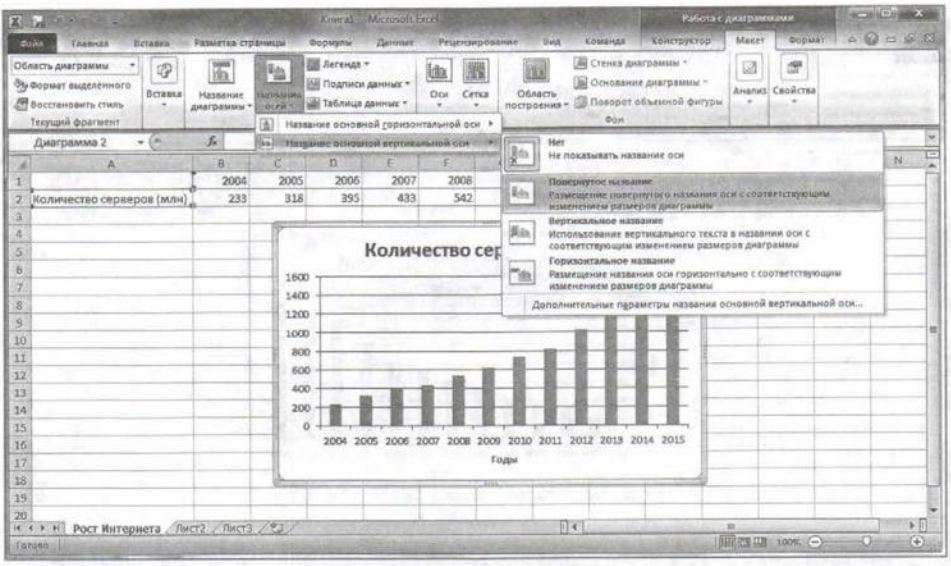


Рис. 2.82

11. В появившемся поле *Название оси* в области диаграммы ввести название оси «Млн».

12. На вкладке *Конструктор* в меню раскрывающегося списка *Стили диаграмм* изменить стиль диаграммы.
На вкладке *Главная* уточнить детали отображения текстовых данных диаграммы (шрифт и цвет шрифта, начертание, размер и т. д.).
13. С помощью кнопки *Переместить диаграмму* из вкладки *Конструктор* разместить диаграмму на имеющемся листе книги «*Рост Интернета*» (рис. 2.83).

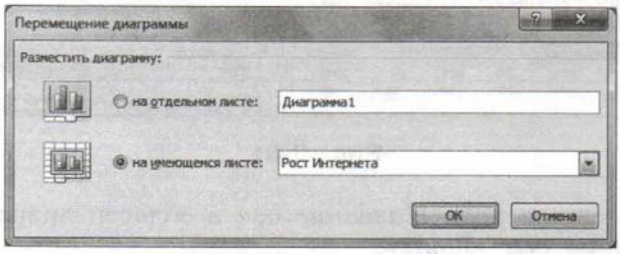


Рис. 2.83

14. Добавить подписи данных. Для этого на вкладке *Макет* в раскрывающемся списке *Подписи данных* выбрать значение *У вершины, снаружи* (рис. 2.84).

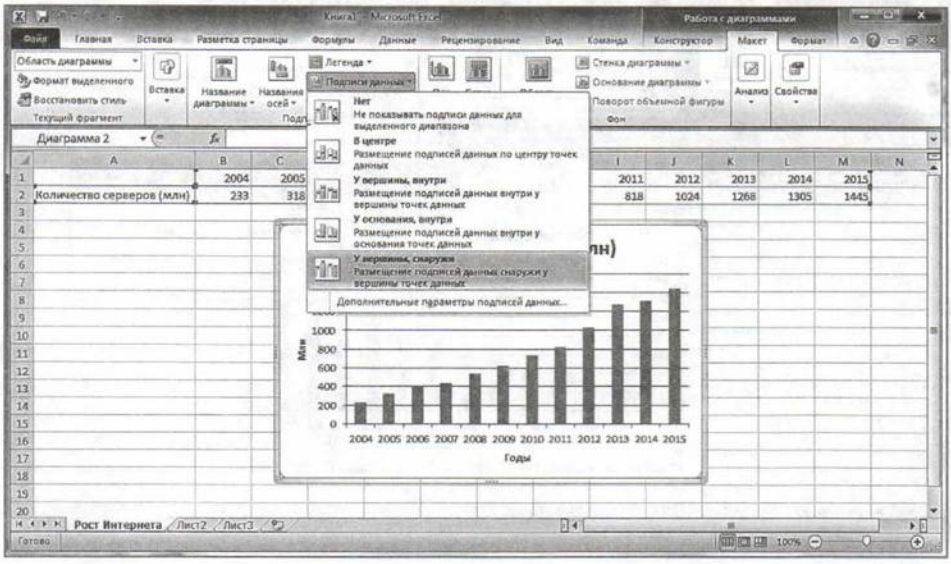


Рис. 2.84

15. В результате на листе с данными *Рост Интернета* получим гистограмму (рис. 2.85).

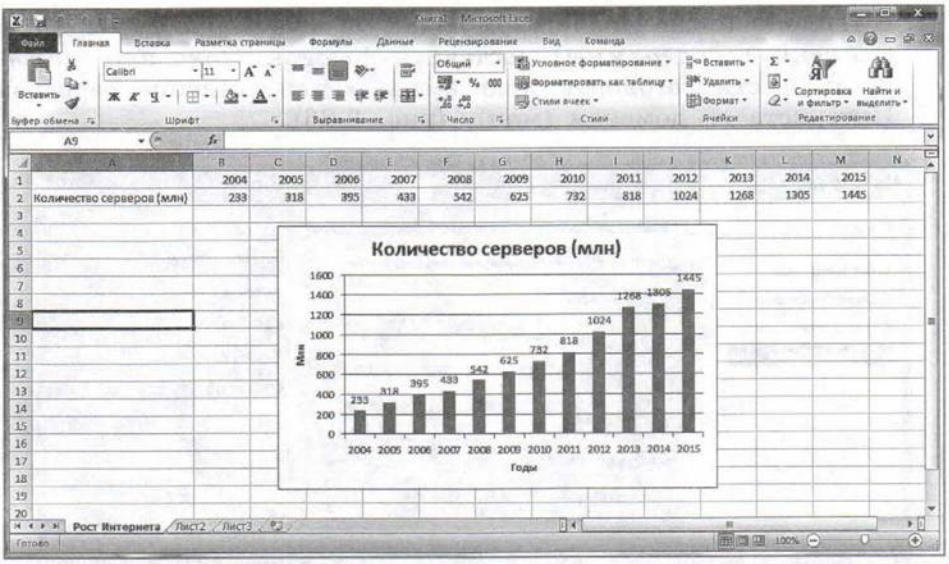


Рис. 2.85

Построение круговой диаграммы в электронных таблицах

Электронное приложение к главе 2.

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel или OpenOffice Calc. Или в операционной системе Linux запустить электронные таблицы OpenOffice Calc. Присвоить листу имя «*Распределение по доменам*».
2. Вставить в электронные таблицы данные из таблицы «Распределение имён серверов Интернета по доменам».
3. Выделить диапазон ячеек A1:J2, содержащий исходные данные. Запустить *Мастер диаграмм* с помощью команды [*Вставка—Диаграмма...*].
4. В появившемся диалоговом окне *Мастер диаграмм* (шаг *Тип диаграммы*) в списке *Выберите тип диаграммы* выбрать *Круговая*. В окне *Вид* выбрать *раздельная круговая*. Щёлкнуть по кнопке *Далее*.
5. В появившемся диалоговом окне *Мастер диаграмм* (шаг *Диапазон данных*) на вкладке *Выберите диапазон данных* выбрать значение переключателя *Ряды данных в строках*. Щёлкнуть по кнопке *Далее*.
6. В появившемся диалоговом окне *Мастер диаграмм* (шаг *Ряды данных*) на вкладке оставить все данные без изменений. Щёлкнуть по кнопке *Далее*.

7. В появившемся диалоговом окне *Мастер диаграмм* (шаг *Элементы диаграммы*) поставить флажок *Показать легенду* и ввести заголовок диаграммы «Распределение серверов Интернета по доменам (млн)» (рис. 2.86).

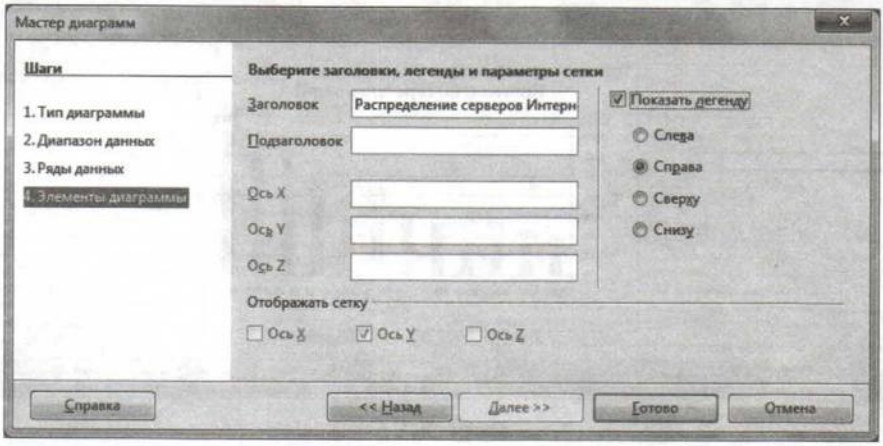


Рис. 2.86

Щёлкнуть по кнопке *Готово*.

В результате на листе «*Распределение по доменам*» получим диаграмму (рис. 2.87). Внешний вид диаграммы можно изменить с помощью контекстных меню элементов диаграммы (область построения диаграммы, круговые сегменты и т. д.).

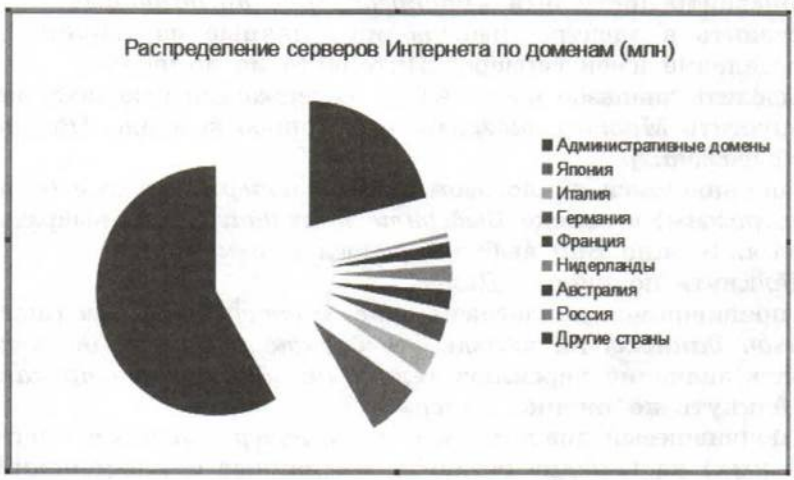


Рис. 2.87

Построение диаграммы типа график в электронных таблицах

Электронное приложение к главе 2.

1. В операционной системе Windows запустить электронные таблицы Microsoft Excel или OpenOffice Calc. Или: в операционной системе Linux запустить электронные таблицы OpenOffice Calc.
2. Вставить в электронные таблицы данные из таблицы «Числовое представление кубической и линейной функций».
3. Присвоить листу имя *«Диаграмма типа график»*.
4. Выделить диапазон ячеек A1:J3, содержащий в качестве исходных данных значения функций. Запустить *Мастер диаграмм* с помощью команды [*Вставка—Диаграмма...*].
5. В появившемся диалоговом окне *Мастер диаграмм* (шаг *Тип диаграммы*) в списке *Выберите тип диаграммы* выбрать *Диаграмма XY*. В окне *Вид* выбрать *Линии и точки*. Оставить флажок *Сглаживание линий* (рис. 2.88).

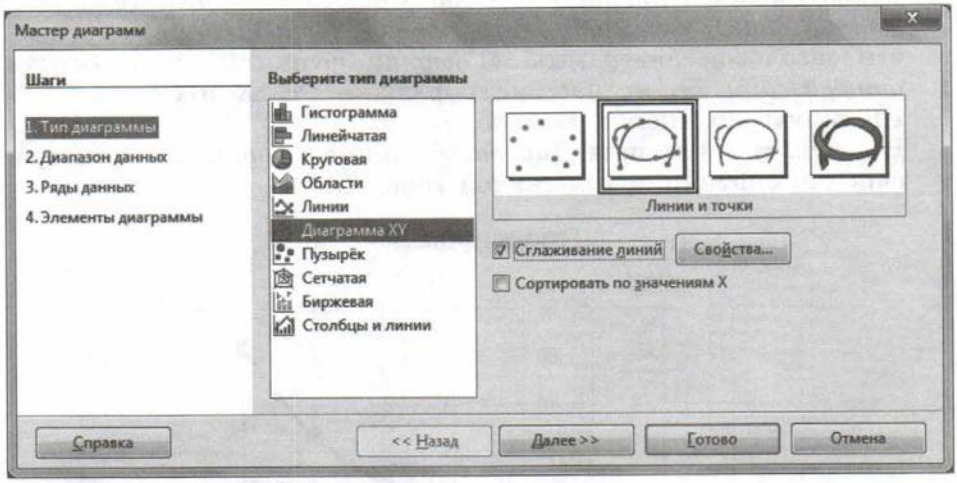


Рис. 2.88

Щёлкнуть по кнопке *Далее*.

6. В появившемся диалоговом окне *Мастер диаграмм* (шаг *Диапазон данных*) на вкладке *Выберите диапазон данных* выбрать значение переключателя *Ряды данных в строках* (рис. 2.89). Щёлкнуть по кнопке *Далее*.

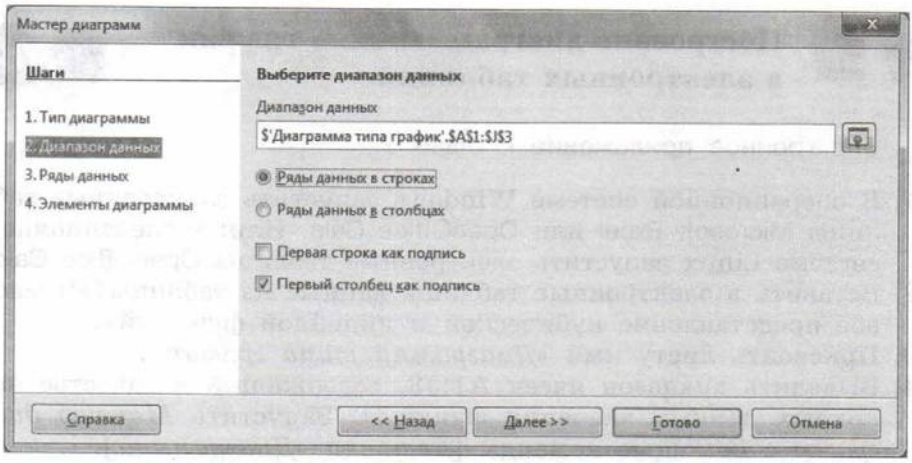


Рис. 2.89

7. В появившемся диалоговом окне *Мастер диаграмм* (шаг *Ряды данных*) оставить все данные без изменений. Щёлкнуть по кнопке *Далее*.
8. В появившемся диалоговом окне *Мастер диаграмм* (шаг *Элементы диаграммы*) поставить флажок *Показать легенду*, ввести заголовок диаграммы «Графики функций» и в области *отображать сетку* поставить флажки *Ось X* и *Ось У*. Щёлкнуть по кнопке *Готово*.

В результате на листе *Диаграмма типа график* получим график двух функций с легендой (рис. 2.90).

Рис. 2.90

ЭОР к главе 2 на сайте ФЦИОР (http://fcior.edu.ru)

- Аппаратное и программное обеспечение для представления звука
- Аппаратное и программное обеспечение для представления изображения
- Арифметические операции в позиционных системах счисления
- Графические редакторы и форматы файлов изображений
- Мультимедиа-презентации. Редакторы презентаций
- Основные программные средства для редактирования таблиц и работы с цифровыми данными
- Основные программные средства для создания и редактирования презентаций
- Основные программные средства и технологии работы с графическими объектами
- Понятие о системах счисления
- Практический модуль: работа в растровом редакторе Paint
- Программы-переводчики
- Работа в текстовом процессоре
- Работа с редакторами электронных таблиц
- Растровая и векторная графика
- Редакторы электронных таблиц
- Создание и редактирование презентаций в среде Microsoft PowerPoint и OpenOffice Impress

Глава 3
КОММУНИКАЦИОННЫЕ ТЕХНОЛОГИИ

При изучении данной главы рекомендуется установить следующее программное обеспечение для операционных систем Windows и Linux:

- визуальный HTML-редактор: KompoZer;

- программу интерактивного общения в Интернете: Skype;

- геоинформационную модель: GoogleEarth;

- браузер Mozilla Firefox;

- почтовую программу Mozilla Thunderbird.

3.1. Локальные компьютерные сети

При работе на персональном компьютере в автономном режиме пользователи могут обмениваться информацией (программами, документами и т. д.), только копируя её на носители информации (флеш-память, CD- и DVD-диски и др.). Однако перемещение носителя информации между компьютерами не всегда возможно и может занимать достаточно продолжительное время.

Создание компьютерных сетей вызвано практической потребностью совместного использования информации пользователями, работающими на удалённых друг от друга компьютерах. Сети предоставляют пользователям возможность не только быстрого обмена информацией, но и совместного использования принтеров и других периферийных устройств, и даже одновременной работы с документами.

Локальная сеть объединяет компьютеры, установленные в одном помещении (например, школьный компьютерный класс, состоящий из 8–12 компьютеров) или в одном здании (например, в здании школы могут быть объединены в локальную сеть несколько десятков компьютеров, установленных в различных предметных кабинетах).

Локальная сеть объединяет несколько компьютеров и позволяет пользователям совместно использовать ресурсы компьютеров, а также подключённых к сети периферийных устройств (принтеров, дисков и др.).

Одноранговые локальные сети. Общая схема соединения компьютеров в локальной сети называется **топологией сети.** Существуют различные топологии сети. В небольших локальных сетях все компьютеры обычно равноправны, т. е. пользователи самостоятельно решают, какие ресурсы своего компьютера (диски, папки, файлы) сделать общедоступными в сети. Такие сети называются **одноранговыми.**

В домашних локальных сетях, включающих обычно не более 2–3 компьютеров, для соединения компьютеров используется топология **линейная шина** (рис. 3.1). В этом случае все компьютеры подсоединяются к одному кабелю, а принтер — к одному из компьютеров.

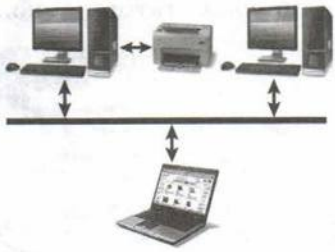


Рис. 3.1

Если из одного центрального сетевого устройства к каждому компьютеру подходит отдельный кабель, то реализуется локальная сеть с топологией **звезда**. В одноранговых локальных сетях с топологией звезда все компьютеры и сетевой принтер соединяются с **концентратором** или **коммутатором**, который обеспечивает передачу данных между устройствами сети (рис. 3.2). По назначению концен-

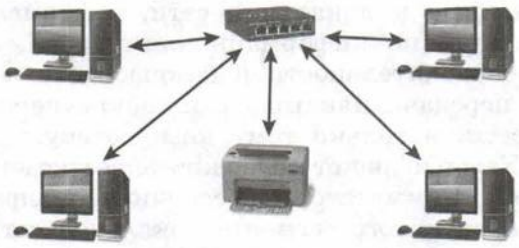


Рис. 3.2

тратор и коммутатор схожи, но концентратор — более простое и дешёвое устройство. Данные, которые поступают на один порт концентратора, рассылаются на все другие порты (на все компьютеры сети). Коммутатор же запоминает адреса компьютеров и передаёт данные строго по адресу.

Преимущество локальной сети типа звезда перед локальной сетью типа линейная шина состоит в том, что при выходе из строя сетевого кабеля у одного компьютера локальная сеть в целом продолжает нормально функционировать.

Сеть на основе сервера. Если к локальной сети подключено более 10 компьютеров, то одноранговая сеть может оказаться недостаточно производительной. Для увеличения производительности, а также в целях обеспечения большей надёжности хранения данных в сети выделяется мощный компьютер для хранения файлов и программных приложений. Такой компьютер называется **сервером**, а локальная сеть — **сетью на основе сервера.** Сеть на основе сервера имеет топологию звезда, т. е. сервер и все компьютеры сети подключаются к центральному концентратору или коммутатору (рис. 3.3).

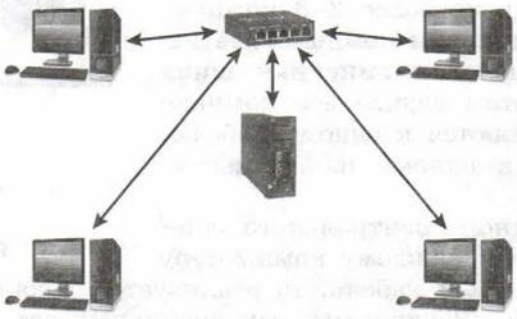


Рис. 3.3

Объединение сегментов сети в единую локальную сеть. Часто сегменты сети объединяются в единую локальную сеть. Компьютеры, подключённые к локальной сети, обмениваются данными в форме небольших по информационному объёму пакетов. Для повышения производительности и безопасности сети обеспечивается адресная передача пакетов, т. е. пакет передаётся не всем компьютерам сети, а только тому компьютеру, для которого он предназначен. Каждый пакет содержит адрес компьютера, которому он направлен. Коммутаторы обеспечивают адресную передачу пакетов в пределах одного сегмента локальной сети, а маршрутизаторы — в пределах всей локальной сети между её сегментами (рис. 3.4).

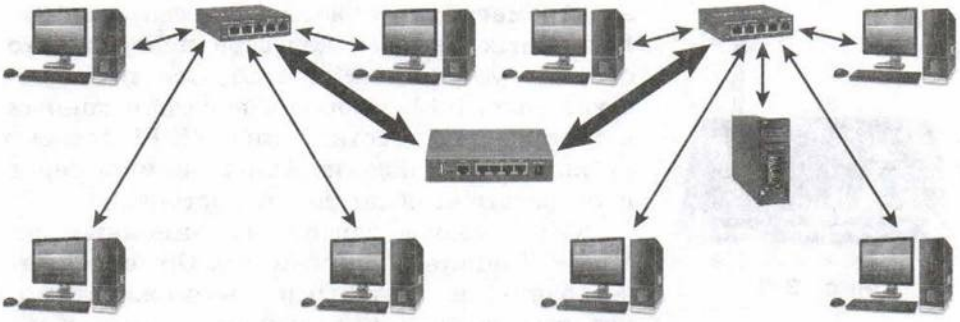


Рис. 3.4

Коммутатор хранит в памяти специальную таблицу, в которой указывается соответствие адресов компьютеров портам коммутатора. При включении коммутатора эта таблица пуста, и он работает в режиме обучения, в котором поступающие на какой-либо порт данные передаются на все остальные порты коммутатора. В течение определённого периода времени на основе анализа пакетов данных коммутатор строит полную таблицу для всех своих портов, в результате чего реализуется адресная передача пакетов данных.

Маршрутизатор для передачи пакетов данных в локальной сети, состоящей из нескольких сегментов, использует таблицу маршрутизации. Таблица маршрутизации состоит из маршрутов, в каждом из которых содержится адрес сегмента локальной сети и адрес компьютера-получателя.

Аппаратное и программное обеспечение проводных и беспроводных сетей. Каждый компьютер или принтер, подключённый к локальной сети, должен иметь специальную плату (сетевой адаптер). Основной функцией сетевого адаптера является передача и приём данных из сети.

В проводных локальных сетях соединение компьютеров (сетевых адаптеров) между собой производится с помощью кабеля (обычно **витой пары** или **оптоволоконного кабеля**). Кабели подключаются к сетевым адаптерам типа Ethernet (рис. 3.5).

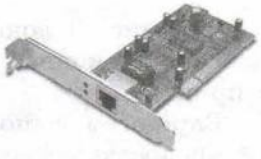


Рис. 3.5

Для подключения к локальной сети портативных компьютеров (ноутбуков, планшетов и т. д.) часто используется беспроводное подключение, при котором передача данных осуществляется с помощью

Рис. 3.6

электромагнитных волн. В беспроводных локальных сетях в качестве центрального сетевого устройства используется **точка доступа** (рис. 3.6). Скорость передачи данных в беспроводных сетях типа **Wi-Fi** зависит от количества подключённых компьютеров и от расстояния до точки доступа.

Современные версии операционных систем Windows, Linux и macOS обладают встроенными сетевыми возможностями, которые делают подключение компьютера к локальной сети простым и быстрым. Для установки на сервер локальной сети существуют специальные серверные версии операционных систем, которые позволяют администратору сети настраивать параметры доступа каждого компьютера к сетевым ресурсам.

Вопросы и задания

1. Каковы основные достоинства и недостатки локальных сетей:
   - одноранговых с топологией линейная шина;
   - одноранговых с топологией звезда;
   - сетей на основе сервера?
2. Каковы основные различия между концентратором, коммутатором и маршрутизатором?
3. В каких случаях целесообразно использование беспроводных сетей?

Практическая работа 3.1

Предоставление общего доступа к принтеру в локальной сети

Задание. В локальной сети, на компьютерах которой установлена операционная система Windows, предоставить общий доступ к принтеру.

Варианты выполнения работы:

- предоставление общего доступа к принтеру в локальной сети компьютерного класса (в данной работе используется локальная сеть в составе настольного компьютера, к которому подключён принтер, и ноутбука); предоставление общего доступа к принтеру в домашней локальной сети.

Предоставление общего доступа к принтеру в локальной сети

На компьютере, к которому подключён принтер, установим общий доступ к принтеру.

1. В операционной системе Windows ввести команду [*Пуск—Устройства и принтеры*].

   В контекстном меню принтера выбрать пункт *Свойства принтера*.

   В появившемся диалоговом окне выбрать вкладку *Доступ* и на ней установить флажок *Общий доступ к данному принтеру* (рис. 3.7).

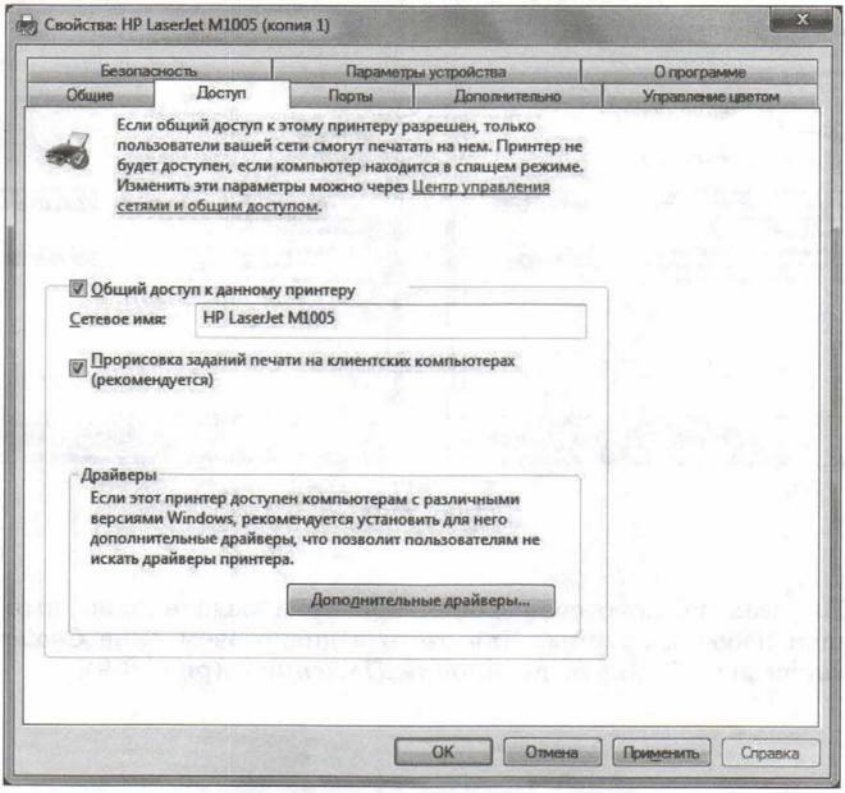


Рис. 3.7

На каждом компьютере, подключённом к локальной сети, зададим описание и имя каждого компьютера, имя общей рабочей группы и включим общий доступ к файлам и принтерам.

2. Ввести команду *Свойства* контекстного меню значка *Компьютер*.

3. В появившемся окне *Просмотр основных сведений о вашем компьютере* перейти к группе *Имя компьютера, имя домена и параметры рабочей группы* и щёлкнуть по ссылке *Изменить параметры*. Появится диалоговое окно *Свойства системы*, в нём задать описание и имя компьютера (рис. 3.8).

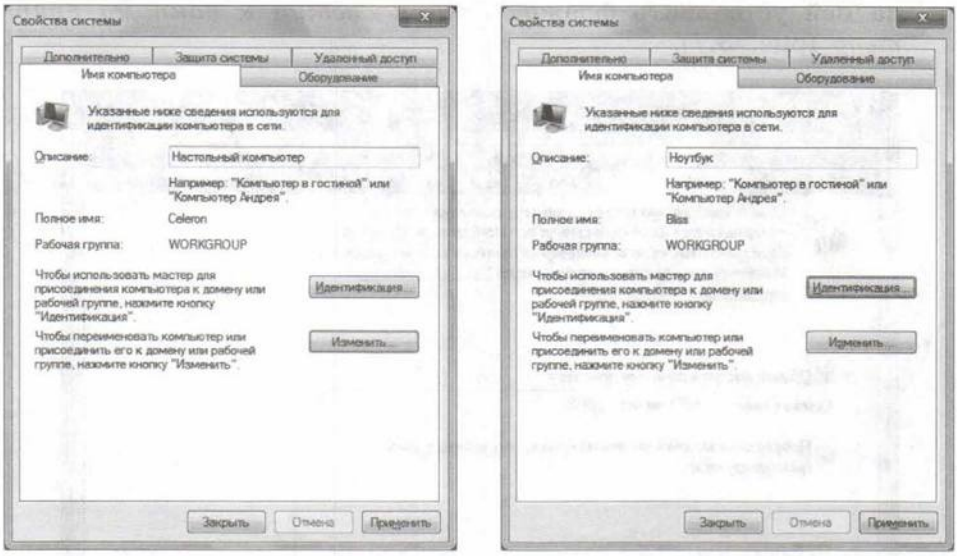


Рис. 3.8

4. На всех компьютерах локальной сети задать одно имя общей рабочей группы. Для этого в диалоговом окне *Свойства системы* щёлкнуть по кнопке *Изменить* (рис. 3.9).

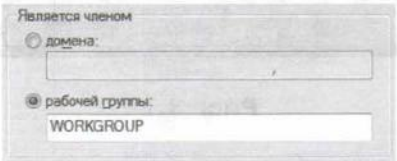


Рис. 3.9

5. На всех компьютерах локальной сети включить общий доступ к файлам и принтерам.

На каждом компьютере локальной сети посмотрим перечень компьютеров, входящих в выбранную рабочую группу локальной сети.

6. На *Рабочем столе* щёлкнуть по значку *Компьютер* и в открывшемся диалоговом окне щёлкнуть по группе *Сеть*. (рис. 3.10).

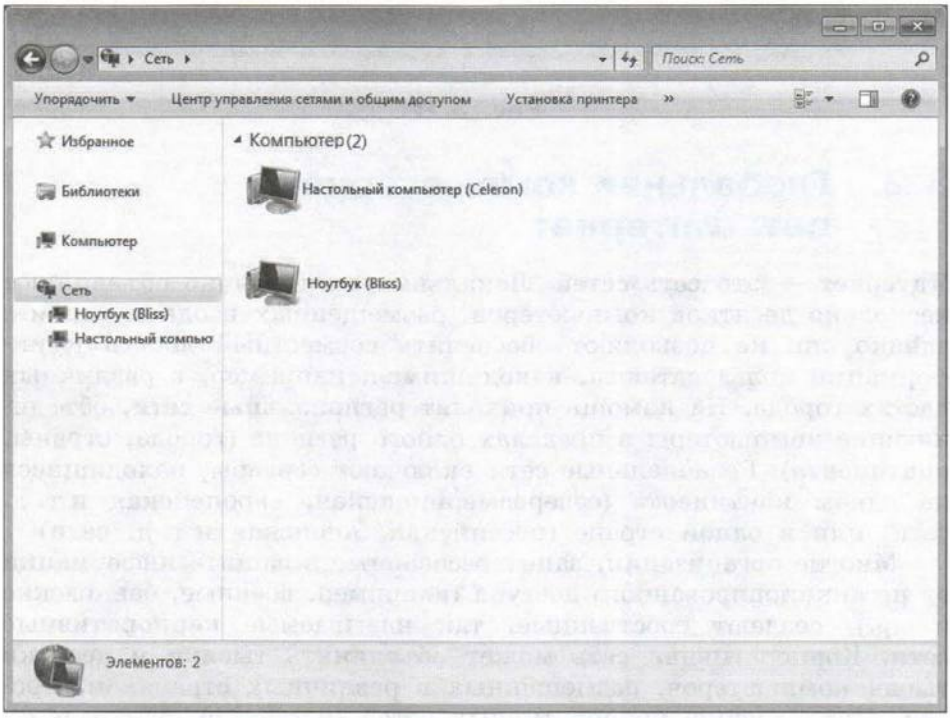


Рис. 3.10

На каждом компьютере, входящем в выбранную рабочую группу локальной сети, убедимся, что принтер доступен для печати документов.

7. В главном меню компьютера, к которому подключён принтер, выбрать пункт меню *Устройства и принтеры*. В диалоговом окне должен отобразиться значок принтера (рис. 3.11).

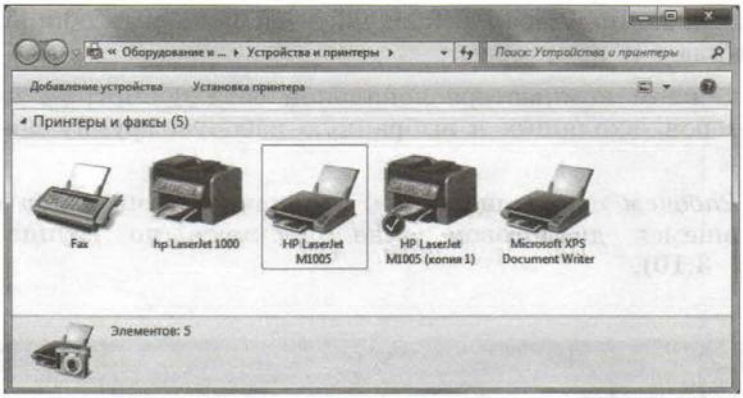


Рис. 3.11

3.2. Глобальная компьютерная сеть Интернет

Интернет — это сеть сетей. Локальные сети обычно объединяют несколько десятков компьютеров, размещённых в одном здании, однако они не позволяют обеспечить совместный доступ к информации пользователям, находящимся, например, в различных частях города. На помощь приходят **региональные сети**, объединяющие компьютеры в пределах одного региона (города, страны, континента). Региональные сети включают серверы, находящиеся на одном континенте (североамериканская, европейская и т. д. сети) или в одной стране (российская, японская и т. д. сети).

Многие организации, заинтересованные в защите информации от несанкционированного доступа (например, военные, банковские и пр.), создают собственные, так называемые **корпоративные сети**. Корпоративная сеть может объединять тысячи и десятки тысяч компьютеров, размещённых в различных странах и городах. Подключение корпоративных сетей (например, банковских) к Интернету осуществляется со строгим соблюдением мер безопасности. Такие меры должны препятствовать несанкционированному доступу к секретной или служебной информации.

Потребности формирования единого мирового информационного пространства привели к созданию **глобальной компьютерной сети Интернет**. В настоящее время на серверах Интернета хранится громадный объём информации (сотни миллиардов файлов, документов и т. д.). Глобальная сеть Интернет привлекает пользователей своими информационными ресурсами и сервисами (услугами), которыми пользуются люди во всех странах мира.

Надёжность и устойчивость функционирования глобальной компьютерной сети обеспечиваются большим количеством линий связи между различными сегментами сети. Внутри региональных сетей и между региональными сетями информация передаётся по многочисленным **оптоволоконным** и **спутниковым** каналам.

Интернет — это глобальная компьютерная сеть, объединяющая многие локальные, региональные и корпоративные сети и включающая серверы, постоянно подключённые к сети.

IP-адрес. Чтобы в процессе обмена информацией компьютеры могли найти друг друга, в Интернете существует единая система адресации, основанная на использовании IP-адреса.

Каждый компьютер, подключённый к Интернету, имеет свой уникальный 32-битовый (в двоичной системе) **IP-адрес**.

Вспомните, что существует формула (1.1), которая связывает между собой количество возможных информационных сообщений N и количество информации i, которое несёт полученное сообщение:

$$N = 2^i.$$

IP-адрес несёт количество информации $i = 32$ бита, тогда общее количество различных IP-адресов N равно:

$$N = 2^{32} = 4\ 294\ 967\ 296.$$

Для удобства восприятия двоичный 32-битовый IP-адрес можно разбить на четыре части по 8 бит и каждую часть представить в десятичной форме. Десятичный IP-адрес состоит из четырёх чисел в диапазоне от 0 до 255, разделённых точками (например, 81.19.70.3) (табл. 3.1).

Таблица 3.1

IP-адрес в двоичной и десятичной формах

Двоичный	01010001	00010011	01000110	00000011
Десятичный	81	19	70	3

Доменная система имён. Компьютеры легко могут найти друг друга по числовому IP-адресу, однако человеку запомнить числовой адрес нелегко. Для удобства была введена **доменная система имён (DNS**: *Domain Name System*).

Доменная система имён ставит в соответствие числовому IP-адресу компьютера уникальное доменное имя.

Доменная система имён имеет иерархическую структуру: домены верхнего уровня — домены второго уровня и т. д. Доменная система имён фактически является базой данных с иерархической структурой, она хранится на иерархии серверов DNS. На верхнем уровне иерархии находится сервер DNS, на котором хранится база данных о доменах верхнего уровня. На следующем уровне иерархии расположены серверы доменов верхнего уровня, на каждом из которых хранится база данных о доменах второго уровня. Затем идут серверы DNS второго уровня и т. д. Каждый домен поддерживается как минимум одним сервером DNS, на котором расположена информация о домене.

Существуют географические и административные домены верхнего уровня.

Географические домены верхнего уровня — двухбуквенные: каждой стране соответствует двухбуквенный код (России принадлежат географические домены ru и рф).

Административные домены позволяют определить профиль организации — владельца домена (com — коммерческая, net — Интернет, телекоммуникационные сети, edu — образовательная и др.).

Приведём примеры доменов верхнего уровня: aero — авиация, biz — бизнес, coop — кооперация, info — информационные организации, museum — музеи, name — личные, pro — юридические и бухгалтерские организации, org — первоначально: некоммерческие организации, сейчас доступен для регистрации любому желающему.

Домен второго уровня можно зарегистрировать в одном из доменов верхнего уровня. Так, корпорация Microsoft зарегистрировала домен второго уровня microsoft в административном домене верхнего уровня com, а поисковая система Яндекс — домен второго уровня yandex в географическом домене верхнего уровня ru.

Имена компьютеров, которые являются серверами Интернета, включают полное доменное имя и собственно имя компьютера. Так, основной сервер компании Microsoft имеет имя **www.microsoft.com**.

Между IP-адресом компьютера и его доменным именем имеется соответствие. Так, сервер поисковой системы Яндекс имеет IP-адрес 213.180.193.3 и соответствующее доменное имя **www.yandex.ru**.

Каждый компьютер, подключённый к Интернету, имеет IP-адрес, однако он может не иметь доменного имени. Доменные

имена имеют серверы Интернета, но их обычно нет у компьютеров, подключающихся к Интернету время от времени.

Доменные имена и IP-адреса распределяются международным координационным центром доменных имен и IP-адресов **ICANN** (адрес в Интернете **www.icann.org**).

В России официальным регистратором доменов второго уровня в доменах верхнего уровня ru и рф (а также su — домен бывшего СССР) и в административных доменах com, net, org, biz и info является **RU-CENTER** (адрес в Интернете: **www.nic.ru**).

Протокол передачи данных TCP/IP. Сеть Интернет, являющаяся сетью сетей и объединяющая громадное количество различных локальных, региональных и корпоративных сетей, функционирует и развивается благодаря использованию единого протокола передачи данных TCP/IP. Термин TCP/IP включает название двух протоколов:

- *Transmission Control Protocol* (**TCP**) — транспортный протокол;
- *Internet Protocol* (**IP**) — протокол маршрутизации.

Транспортный протокол обеспечивает разбиение файлов на IP-пакеты в процессе передачи и сборку файлов в процессе получения. Если послать большой по информационному объёму файл целиком, то он может надолго «закупорить» канал связи, сделать его недоступным для пересылки других сообщений. Чтобы этого не происходило, на компьютере-отправителе необходимо разбить большой файл на мелкие части, пронумеровать их и транспортировать в отдельных IP-пакетах до компьютера-получателя. А на компьютере-получателе необходимо вновь собрать исходный файл из отдельных частей в правильной последовательности.

Протокол маршрутизации обеспечивает передачу информации между компьютерами сети. IP-пакеты на пути к компьютеру-получателю проходят через многочисленные промежуточные серверы Интернета, на которых производится операция маршрутизации. В результате маршрутизации IP-пакеты направляются от одного сервера Интернета к другому, постепенно приближаясь к компьютеру-получателю.

Маршруты доставки IP-пакетов могут быть совершенно разными, поэтому они могут приходить не в том порядке, в котором их отправили. «География» Интернета существенно отличается от привычного нам пространственного распределения. Скорость получения информации зависит не от удалённости сервера Интернета, а от маршрута прохождения информации, т. е. от количества промежуточных серверов и качества линий связи (их пропускной способности), по которым передаётся информация от сервера к серверу.

Вопросы и задания

1. Почему глобальная компьютерная сеть Интернет продолжает нормально функционировать даже после выхода из строя отдельных серверов и линий связи?
2. Обязательно ли компьютер, подключённый к Интернету, имеет IP-адрес?
3. Как формируется доменная система имён?
4. Как работает транспортный протокол? Протокол маршрутизации?

3.3. Подключение к Интернету

Интернет-провайдеры. Подключение пользователей к Интернету обеспечивают организации — интернет-провайдеры, серверы которых обычно имеют несколько высокоскоростных линий связи с Интернетом. В число предоставляемых интернет-провайдером услуг, как правило, входят доступ в Интернет, выделение дискового пространства для хранения и обеспечения работы сайтов, поддержка работы электронных почтовых ящиков и др.

Интернет-провайдеры предоставляют пользователям доступ в Интернет как с постоянным, так и с динамическим IP-адресом, который может меняться при каждом подключении к сети. В процессе сеанса работы в Интернете можно определить свой текущий IP-адрес.

ADSL-подключение по телефонной линии. Для подключения отдельных компьютеров или небольших локальных сетей может использоваться технология ADSL. В этом случае информация передаётся в виде цифровых сигналов по аналоговым телефонным каналам со значительно более высокочастотной модуляцией, чем та, которая обычно используется для традиционной аналоговой телефонной связи (рис. 3.12).

Рис. 3.12

Для создания соединения ADSL требуются два ADSL-модема — один у интернет-провайдера, а другой у конечного пользователя. Между этими двумя модемами — обычный телефонный

провод. На телефонной линии организуются три информационных канала: канал обычной телефонной связи, исходящий поток передачи данных и входящий поток передачи данных. Благодаря этому телефонный разговор можно вести одновременно с приёмом/передачей данных по той же телефонной линии.

Обмен данными между ADSL-модемами идёт, соответственно, на трёх диапазонах частот. Для использования обычной телефонной связи резервируется небольшой диапазон частот, для передачи данных — больший, а для приема данных — ещё больший (рис. 3.13). Чем шире полоса пропускания частот, тем быстрее передаются данные, поэтому технология ADSL является асимметричной. Скорость исходящего потока данных в основном превышает скорость входящего потока, что отвечает интересам пользователя, так как он всегда больше информации получает, чем передаёт.

Рис. 3.13

Подключение компьютера к Интернету с использованием мобильного телефона. Для доступа в Интернет можно использовать сеть мобильной телефонной связи, которая позволяет передавать не только голосовые сообщения, но и данные. Операторы мобильной телефонной связи и интернет-провайдеры обеспечивают возможность передачи данных между этими сетями.

Мобильный телефон, оснащённый модемом, подключается обычно к компьютеру через USB-порт или с помощью беспроводной связи (инфракрасного или bluetooth-адаптера), что обеспечивает возможность высокоскоростного доступа в Интернет по технологии GPRS, EDGE, 3G, 4G(LTE). Скорость передачи данных уменьшается с увеличением расстояния до антенны станции сотовой связи и загрузки линии телефонными разговорами. Важно, что эта технология позволяет одновременно вести разговор по мобильному телефону и проводить обмен данными между компьютером и Интернетом.

Беспроводное подключение. В аэропортах, на вокзалах, кафе и других общественных местах устанавливаются точки беспровод-

Рис. 3.14

ного доступа в Интернет. Посетители этих мест с ноутбука или смартфона, оснащённого сетевой картой Wi-Fi, могут соединяться с Интернетом. Скорость передачи данных зависит от расстояния до точки доступа, наличия препятствий прохождению электромагнитных волн и от количества подключённых компьютеров.

Смартфон (коммуникатор, рис. 3.14) предоставляет пользователю выбор способов подключения к Интернету на основе беспроводных технологий (GPRS, Wi-Fi и др.).

Подключение по локальной сети. Интернет-провайдер подводит кабель локальной сети непосредственно в квартиру потребителя и подключает её к сетевой карте компьютера. Скорость выхода в Интернет зависит не только от скорости сетевых карт, но и от скорости каналов подключения интернет-провайдера к Интернету и от количества подключённых пользователей.

Подключение по оптоволоконной линии. Для подключения больших локальных сетей (из нескольких десятков компьютеров) обычно используется оптоволоконный канал. Оптоволокно позволяет передавать цифровую информацию на большие расстояния и с высокой скоростью передачи данных. На концах оптоволоконной линии у потребителя и интернет-провайдера устанавливаются оптические модемы, которые преобразуют электрические импульсы в оптический сигнал, и наоборот — оптический сигнал в электрические импульсы.

Подключение по спутниковому каналу. В случаях подключения неудобно расположенных или удалённых компьютерных сетей, когда прокладка кабеля затруднена или невозможна, используются спутниковые линии связи между интернет-провайдером и клиентом.

Асимметричный доступ в Интернет использует приёмную антенну для реализации высокоскоростного канала, по которому поступают данные из Интернета через спутник. Исходящие от пользователя данные (запросы на загрузку страниц, исходящие сообщения и т. д.) передаются через наземный канал.

Двунаправленный спутниковый доступ к сети Интернет использует приёмопередающую антенну для реализации высокоскоростных каналов для приёма/передачи данных из Интернета через спутник.

Способы подключения к Интернету проиллюстрированы на рис. 3.15.

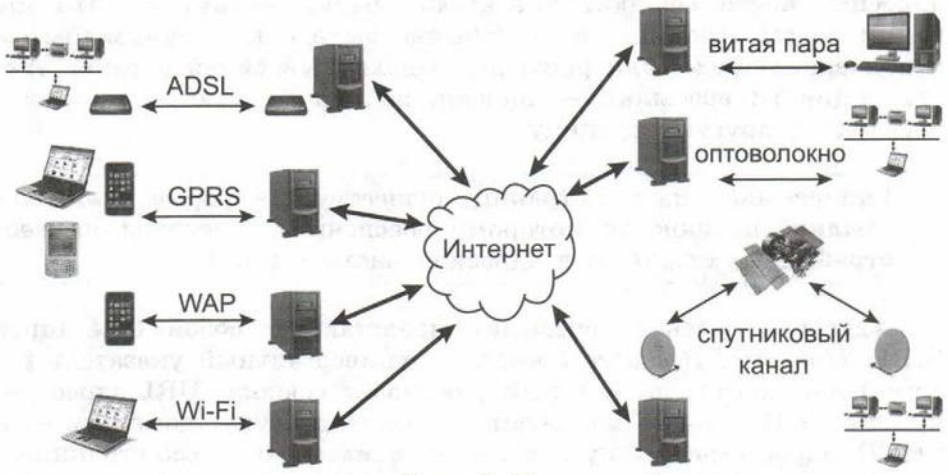


Рис. 3.15

Вопросы и задания

1. Какие существуют способы подключения к Интернету? Какой способ подключения целесообразно выбрать для подключения к Интернету вашего школьного компьютерного класса? Домашнего настольного компьютера? Ноутбука?
2. От чего зависит реальная скорость подключения к Интернету с помощью GPRS-соединения? Соединения с помощью локальной сети?
3. Какие существуют технологии беспроводного подключения к Интернету?

3.4. Всемирная паутина

«Всемирная паутина» — это вольный перевод английского словосочетания *World Wide Web*, которое часто обозначается как **WWW**, или **Веб.** Бурное развитие сети Интернет, которое происходило на протяжении 1990-х годов, в первую очередь обусловлено появлением новой технологии WWW.

Технология WWW. Технология WWW позволяет создавать ссылки — их называют **гиперссылками**, — которые реализуют переходы на любой документ любого компьютера, подключённого в данный момент к Интернету. Связанные гиперссылками вебстраницы образуют **гипертекст.** Гипертекст позволяет пользователю выбирать информацию нелинейно, в требуемой последовательности. Всемирная паутина является глобальной гипертекстовой информационной системой.

Тематически связанные веб-страницы образуют веб-сайты.

Гиперссылка состоит из двух частей: **указателя ссылки** и **адресной части ссылки**. Указатель ссылки — это то, что мы видим на веб-странице: текст (обычно выделенный синим цветом и подчёркиванием) или рисунок, выделенный синей рамкой. Активизация гиперссылки — щелчок на ней мышью — вызывает переход на другую страницу.

Гиперссылка на веб-странице существует в форме указателя ссылки, щелчок по которому обеспечивает переход на веб-страницу, указанную в адресной части ссылки.

Адресная часть гиперссылки представляет собой **URL-адрес** (**URL**: *Universal Recource Locator* — универсальный указатель ресурса) документа, на который указывает ссылка. URL-адрес документа в Интернете включает протокол доступа, доменное имя или IP-адрес сервера, путь к файлу и имя файла веб-страницы.

Протокол доступа к документу определяет способ доступа к нему. Для доступа к веб-страницам используется **протокол передачи гипертекста HTTP** (*Hyper Text Transfer Protocol*). При записи протокола после его имени записывают двоеточие и два символа «косая черта»: http://

Запишем, например, URL-адрес новостной ленты веб-сайта издательства «БИНОМ. Лаборатория знаний» (сервер: **www.Lbz.ru**). Начальная страница этой новостной ленты расположена в файле index.htm каталога news. Следовательно, универсальный указатель ресурсов примет вид:

http://www.Lbz.ru/news/index.htm

Он состоит из трёх частей:

http://	— протокол доступа;
www.Lbz.ru	— доменное имя сервера;
/news/index.htm	— путь к файлу и имя файла веб-страницы.

Язык разметки гипертекста. Создание веб-страниц осуществляется с помощью языка разметки гипертекста **HTML** (*Hyper Text Markup Language*). Основа используемой в HTML технологии состоит в том, что в обычный текстовый документ вставляются управляющие символы — **теги**. В результате мы получаем текстовый документ, который при просмотре в браузере видим в форме веб-страницы. С помощью тегов можно изменять размер, начертание и цвет символов, выбирать фон, определять положение текста на странице, вставлять гиперссылки и т. д.

Веб-страница может быть *мультимедийной*, т. е. содержать ссылки на различные мультимедийные объекты: графические изображения, анимацию, звук и видео.

Интерактивные веб-страницы содержат формы, которые может заполнять посетитель. На форме могут быть размещены элементы управления: текстовые поля, списки, переключатели, флажки и др.

Язык разметки гипертекста рассматривает документ как совокупность объектов, свойства которых можно изменять. Это позволяет создавать *динамические веб-страницы*, т. е. страницы, которые могут меняться уже после загрузки в браузер. Например, текст может менять цвет, когда к нему подводится курсор, заголовок — перемещаться и т. д.

Флеш-технология. Флеш-технология (*flash*) основана на использовании векторной графики и позволяет создавать мультимедийные интерактивные веб-страницы. Флеш-страницы отличаются малым информационным объёмом (высокой скоростью загрузки) и высоким качеством графики.

Веб-сайт. Публикации во Всемирной паутине реализуются в форме веб-сайтов. Веб-сайт содержит информацию, посвящённую какой-либо теме или проблеме. Веб-сайт состоит из веб-страниц.

Обычно сайт имеет титульную страницу (страницу с оглавлением), на которой имеются гиперссылки на основные разделы сайта (веб-страницы). Гиперссылки также имеются на других веб-страницах сайта, что обеспечивает пользователю возможность свободно перемещаться по сайту.

Интернет-портал. Интернет-портал предоставляет пользователю Интернета возможность получения информации с других сайтов с использованием внешних ссылок (ведущих на другие ресурсы сети). «Горизонтальными» принято называть порталы, охватывающие много тем. Типичными «горизонтальными порталами» являются порталы поисковых систем (Яндекс, Google и др.). «Вертикальными» называются специализированные тематические порталы. Например, «вертикальным» является Российский общеобразовательный портал, который размещён в Интернете по адресу **http://school.edu.ru**.

Браузер. Это программное средство для доступа к информационным ресурсам Всемирной паутины. Браузер загружает веб-страницу и отображает её в соответствии с тегами языка разметки гипертекста (HTML). В настоящее время наиболее распространёнными браузерами являются Chrome, Яндекс. Браузер, Opera, Mozilla FireFox, Internet Explorer, Safari.

После загрузки в браузер начальной (домашней) страницы «путешествовать» по Всемирной паутине можно различными способами:

- воспользоваться ссылками загруженной в браузер веб-страницы;
- в строку *Адрес* ввести адрес (URL) нужной веб-страницы;
- воспользоваться закладками веб-страниц.

Сохранение веб-страниц. В процессе просмотра в браузере веб-страниц их копии и связанные с ними мультимедийные файлы автоматически сохраняются в **кэш-памяти** локального компьютера. Пользователь может установить максимальный объём кэш-памяти и место её размещения в определённой папке на жёстком диске. В случае нехватки места на жёстком диске кэш-память можно очистить.

Загрузка копий веб-страниц из кэш-памяти локального компьютера существенно ускоряет их просмотр, однако в этом случае может возникнуть ситуация, когда веб-страница в Интернете изменилась, а в браузере мы будем просматривать её устаревшую копию.

Найти важные и интересные веб-страницы в кэш-памяти довольно трудно, поэтому полезно специально сохранять их в определённой папке на локальном компьютере. Можно выбрать различные варианты сохранения веб-страниц:

- сохранение страницы в формате *web-страница* (*только HTML*) приведёт к сохранению самой страницы, но при этом не сохранятся связанные с ней рисунки, звуковые и прочие файлы;
- сохранение страницы в формате *TXT* приведёт к сохранению самой страницы в текстовом формате;
- сохранение страницы в формате *web-страница полностью* приведёт к сохранению не только самой страницы, но и связанных с ней рисунков, звуковых и прочих файлов в отдельной папке.

Можно сохранить не только веб-страницу полностью, но и отдельные её части: текст, изображения или ссылки.

Off-line браузеры. Для быстрой загрузки веб-сайтов с целью их дальнейшего неспешного просмотра в автономном режиме используются специальные программы — off-line браузеры.

Off-line браузеры (например, Offline Explorer) позволяют загружать на локальный компьютер веб-сайты целиком или отдельные части сайта по выбору. Пользователь может установить необходимую «глубину» загрузки веб-сайта (количество вложенных каталогов), загрузку связанных со страницами мультимедиа файлов, загрузку по гиперссылкам веб-страниц с других веб-серверов и т. д. Имеется возможность продолжения загрузки сайта после разрыва соединения и обновления ранее загруженных сайтов.

Вопросы и задания

1. Из каких двух частей состоит гиперссылка? Какую функцию выполняет каждая из её частей?
2. Из каких частей состоит URL-адрес документа в Интернете?
3. Как вы думаете, в чём главное различие между языком разметки гипертекста и языками программирования?
4. Всемирная паутина состоит из веб-страниц? Веб-сайтов? Интернет-порталов?

Практическая работа 3.2

Настройка браузера

Задание. Произвести настройку браузера:

- установить начальную веб-страницу;
- настроить кэш-память браузера;
- установить правильную кодировку для отображения веб-страниц.

Варианты выполнения работы:

- настройка различных браузеров: Mozilla FireFox; Internet Explorer, Chrome.

Настройка браузера Mozilla FireFox

Установим первую загружаемую в браузер веб-страницу.

1. На панели инструментов нажать кнопку *Открыть меню* и выбрать пункт *Настройки*. Откроется вкладка *Настройки*.
2. В разделе *Основные* в поле *Домашняя страница* ввести URL-адрес страницы, которую вы хотите назначить первой загружаемой в браузер (например, начальной страницы поисковой системы Google: **http://www.Google.ru**).
3. Чтобы убедиться, что введён правильный адрес, закрыть окно браузера и открыть его заново. Браузер должен загрузить начальную страницу веб-сайта Google.ru.

Настроим кэш-память браузера.

4. Нажать кнопку *Открыть меню* и выбрать пункт *Настройки*. Откроется вкладка *Настройки*.
5. Во вкладке *Настройки* в разделе *Дополнительные* перейти на вкладку *Сеть*:
   - для очистки кэш-памяти нажать кнопки *Очистить сейчас* для разделов *Кэшированное веб-содержимое* и *Автономное веб-содержимое и данные пользователя*;

- для установки объёма кэш-памяти установить флажок *Отключить автоматическое управление кэшем* и указать числовое значение, которое может быть отведено под кэш, в поле *Использовать под кэш не более ... Мб на диске* (рис. 3.16).

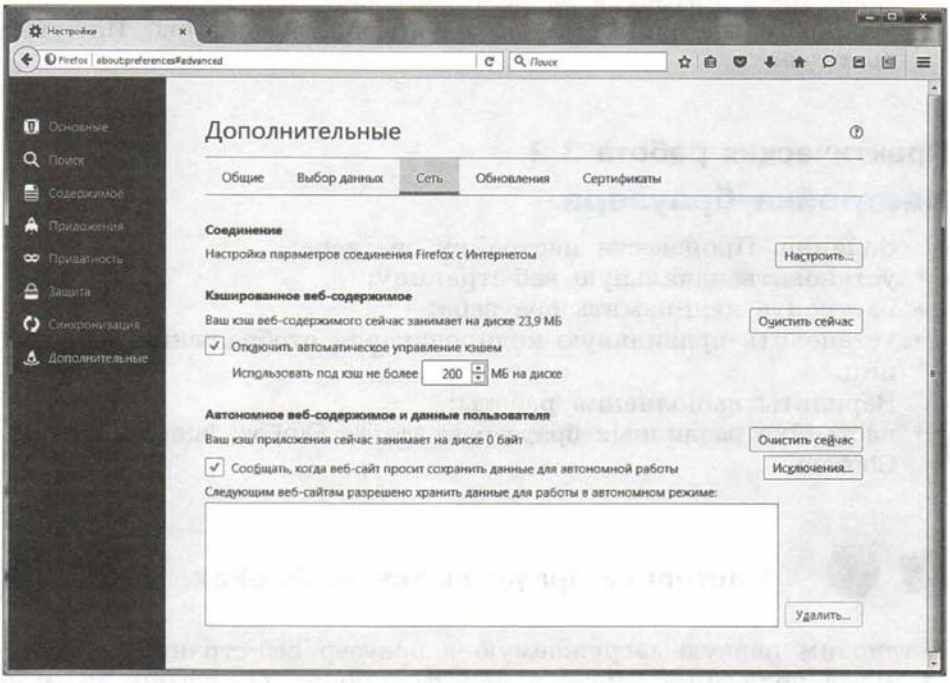


Рис. 3.16

Установим кодировку.

Большое значение имеет настройка браузера на просмотр веб-страницы в правильной кодировке, т. е. в той кодировке, в которой веб-страница была создана. Как правило, браузер автоматически определяет кодировку и, соответственно, правильно отображает веб-страницу. Однако в некоторых случаях пользователю необходимо настроить браузер на требуемую кодировку вручную.

6. Просмотр веб-страницы в кодировке *Windows-1251* можно выполнить двумя способами:
   - вызвать главное меню (клавиша F10) и выполнить команду [*Вид—Кодировка текста—Кириллица* (Windows)];
   - на панели инструментов нажать кнопку *Показать настройки кодировки текста* и в появившемся списке выбрать кодировку *Кириллица* (*Windows*).

3.5. Электронная почта

Возможности электронной почты. Электронная почта (**e-mail**) позволяет:

- обеспечить передачу сообщения в течение нескольких десятков секунд;
- включать в сообщение не только текст, но и вложенные файлы (программы, графику, звук и т. д.);
- посылать сообщение сразу нескольким абонентам;
- пересылать письма на другие адреса;
- создать правила для выполнения определённых действий с однотипными сообщениями (например, удалять рекламные сообщения, приходящие от определённых адресов) и т. д.

Адрес электронной почты. Первая часть почтового адреса (user_name — имя пользователя) имеет произвольный характер и задаётся пользователем при регистрации электронного почтового ящика. Вторая часть (server_name — имя сервера) является доменным именем почтового сервера, на котором пользователь зарегистрировал свой почтовый ящик.

Адрес электронной почты записывается по определённой форме и состоит из двух частей, разделённых символом @:

user_name@server_name

Например, почтовый сервер Mail.Ru имеет имя mail.ru. Соответственно, имена почтовых ящиков пользователей будут иметь вид:

user_name@mail.ru

Допустимый набор символов в адресе электронной почты уточняется провайдером.

Любой пользователь Интернета может зарегистрировать почтовый ящик на почтовом сервере своего интернет-провайдера. В почтовом ящике будут накапливаться передаваемые и получаемые электронные письма. В настоящее время также существует достаточно большое количество почтовых серверов Интернета, которые предоставляют возможность бесплатно зарегистрировать почтовый ящик (Yandex.ru, Mail.Ru, GMail.com и др.).

Почтовые программы. Для работы с электронной почтой используют почтовые программы, причём для любой компьютерной платформы существует большое количество почтовых программ, например Mozilla Thunderbird, Claws Mail, The Bat!

Для пользования почтовым ящиком необходимо создать в почтовой программе **учётную запись**. В ней указываются имя почтового ящика, пароль доступа к нему, имена почтовых серверов, которые получают и отсылают почту, имя абонента, которое будет отображаться в его сообщениях, и другие параметры. Если пользователь имеет несколько почтовых ящиков, то учётные записи создаются для каждого из них отдельно.

Почтовые сообщения хранятся в папках, которые автоматически создаёт почтовая программа:

- *Входящие* — содержит получаемые адресатом письма;
- *Исходящие* — содержит отправляемые адресатом письма, с момента их создания и до их отправки с локального компьютера пользователя на почтовый сервер провайдера;
- *Отправленные* — содержит копии всех отправленных писем;
- *Удалённые* — временно хранит удалённые письма.

Пользователь может создавать собственные папки для хранения тематически сгруппированных сообщений. В папках могут храниться не только сообщения, но и файлы, созданные с помощью других приложений.

Почтовые программы обычно предоставляют пользователю набор параметров, которые можно настраивать:

- шрифт и кодировку текстовых сообщений;
- проверку правописания;
- порядок получения и отправки сообщений и т. д.

С помощью почтовой программы на локальном компьютере создаётся почтовое сообщение. На этом этапе кроме написания текста сообщения необходимо указать адрес получателя сообщения, тему сообщения и при необходимости вложить в сообщение файлы. Адрес получателя сообщения удобно брать из *адресной книги*, хранящей адреса электронной почты абонентов, а также другие сведения (фамилия, имя, отчество, место работы, телефон и т. д.).

Процесс передачи сообщения начинается с подключения к Интернету и доставки сообщения в свой почтовый ящик на удалённом почтовом сервере. Почтовый сервер сразу же отправляет это сообщение через систему почтовых серверов Интернета на почтовый сервер получателя в его почтовый ящик.

Адресат для получения письма должен соединиться с Интернетом и доставить почту из своего почтового ящика на удалённом почтовом сервере на свой локальный компьютер.

Веб-почта. Существует достаточно много веб-серверов, которые предоставляют возможность бесплатно зарегистрировать почтовый ящик и пользоваться им. Сообщения веб-почты хранятся на

веб-сервере, а не доставляются на локальный компьютер, поэтому с веб-почтой работают с использованием браузера.

Для регистрации почтового ящика необходимо загрузить в браузер домашнюю страницу сервера, предоставляющего почтовые услуги, и активизировать ссылку, например *Зарегистрироваться*, *Получить адрес* и т. д. Пользователь может выбрать имя своего почтового ящика и пароль доступа к нему.

Спам и защита от спама с использованием правил для сообщений. Спам — это массовая автоматическая рассылка рекламных электронных сообщений со скрытым или фальсифицированным обратным адресом. Спам приходит потому, что электронный адрес получателя становится известен спаммерам (рассыльщикам спама). Чаще всего владелец почтового ящика сам указывает свой электронный почтовый адрес при регистрации на каком-нибудь сайте либо оставляет его на своей веб-странице для обратной связи, и его обнаруживает специальная программа — робот, «бродящий» по сайтам наподобие индексирующего робота поисковых систем.

Для борьбы со спамом используются антиспамовые фильтры, которые могут быть установлены в почтовых программах на локальных компьютерах пользователей. При создании антиспамового правила для сообщений необходимо указать:

- условие применения правила (от кого получено, наличие вложения, наличие определённых слов в сообщении, размер сообщения и т. д.);
- действие для данного правила (удалить с почтового сервера, не загружать с почтового сервера, переместить в папку и т. д.);
- порядок действия данного правила (указать предельный размер сообщения и т. д.).

Почтовые черви. Почтовые черви — это вредоносные программы, которые для своего распространения используют электронную почту. Червь отсылает свою копию в виде вложения в электронное письмо или вставляет в письмо ссылку на свой файл, расположенный на каком-либо сетевом ресурсе. В первом случае код червя активизируется при открытии (запуске) заражённого вложения, во втором — при открытии ссылки на заражённый файл. В обоих случаях эффект одинаков: активизируется код червя и компьютер оказывается заражённым.

Лавинообразная цепная реакция распространения почтового червя базируется на том, что червь после заражения компьютера начинает рассылать себя по всем адресам электронной почты, которые имеются в адресной книге пользователя.

Профилактическая защита от почтовых червей состоит в том, что не рекомендуется открывать вложенные в почтовые сообщения файлы, полученные из сомнительных источников, и переходить по неизвестным ссылкам.

Вопросы и задания

1. Из каких частей состоит адрес электронной почты?
2. Как функционирует электронная почта с помощью почтовой программы?
3. Что такое веб-почта?
4. Что такое спам и как с ним бороться?
5. Как проникают на компьютер и размножаются почтовые черви?

Практическая работа 3.3

Работа с электронной почтой

Задание 1. В почтовой программе создать учётную запись почты.

Задание 2. В почтовой программе создать, отправить и получить сообщение.

Варианты выполнения работы:

- работа с электронной почтой в почтовой программе Mozilla Thunderbird или Claws Mail.

Создание учётной записи почты

В первую очередь необходимо в соответствии с полученными в процессе регистрации почтового ящика данными (имя почтового ящика, пароль и др.) настроить почтовую программу.

Создадим в почтовой программе Mozilla Thunderbird учётную запись user_name@server_name, при помощи которой можно будет отправлять и принимать электронную почту с конкретного почтового ящика.

1. В окне программы Mozilla Thunderbird ввести команду [*Файл—Создать—Настроить мою учётную запись почты...*].
2. Откроется диалоговое окно *Настройка учётной записи почты* (рис. 3.17).

В поле *Ваше имя* указать имя, которое будет видеть человек, получивший от вас письмо.
В поле *Адрес эл. почты:* указать адрес, который вы задали при регистрации почтового ящика. Этот адрес следует указать целиком и именно в том виде, в котором вы его создали.
В поле *Пароль:* необходимо указать пароль, который был получен при регистрации почтового ящика.
Нажать кнопку *Продолжить*.

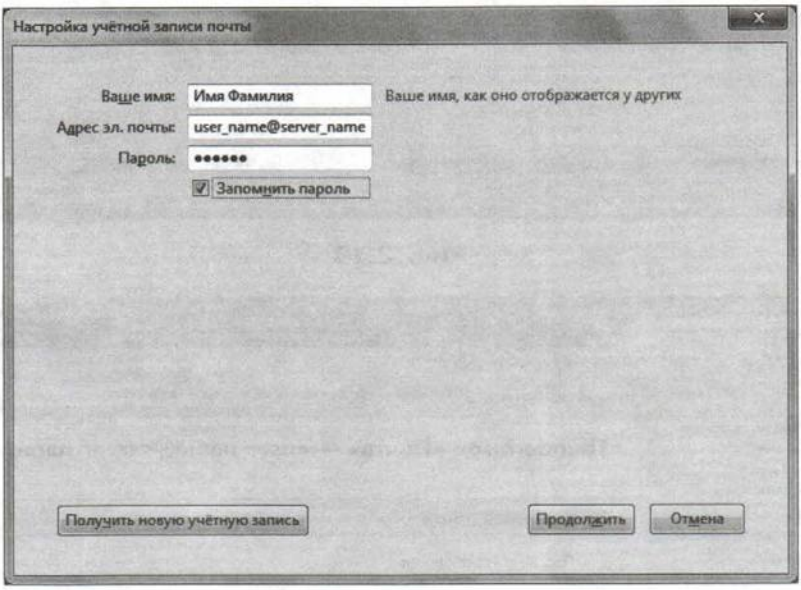


Рис. 3.17

Заданные выше параметры электронной почты объединяются под одним именем — именем учётной записи.

На следующем шаге необходимо указать имя сервера исходящих сообщений и имя сервера входящих сообщений.

3. Для настройки данных нажать кнопку *Настройка вручную* и ввести в поле *Входящая* и *Исходящая* (рис. 3.18) имена серверов входящей и исходящей почты, которые сообщает провайдер при регистрации почтового ящика.
Нажать кнопку *Готово*.

После этого в почтовой программе Mozilla Thunderbird появится новая учётная запись почты (рис. 3.19).

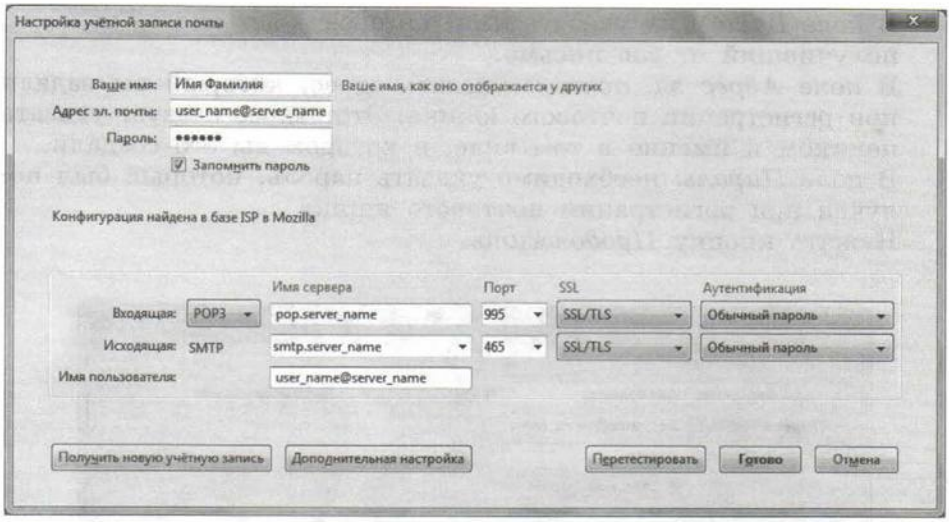


Рис. 3.18

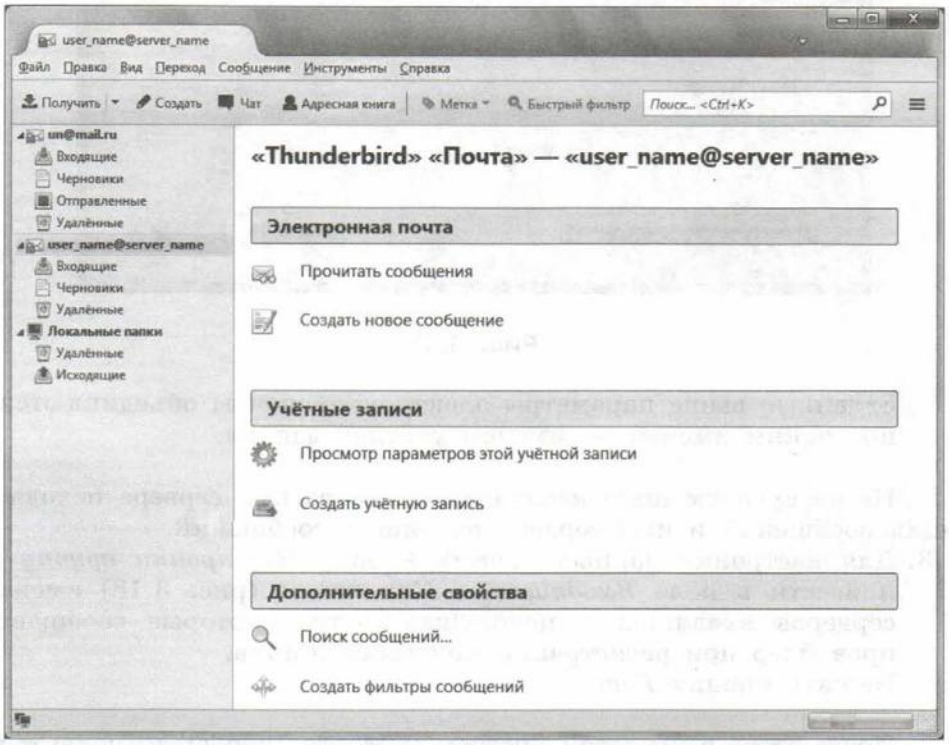


Рис. 3.19

4. Выполнить команду [*Правка—Свойства*]. Появится диалоговое окно *Параметры учётной записи* (рис. 3.20), во вкладках которого можно внести изменения во введённые ранее значения параметров данной учётной записи электронной почты.

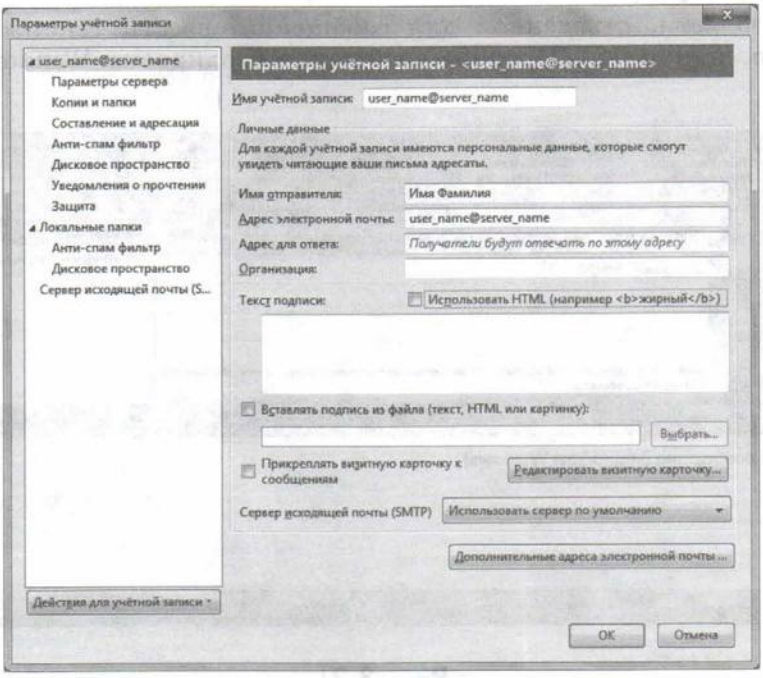


Рис. 3.20

Создание, отправка и получение сообщений

Почтовый ящик Ugrinovich@Lbz.ru (в авторской мастерской Н. Д. Угриновича на методическом сайте издательства **http://metodist.Lbz.ru/authors/informatika/1/**).

Создадим пробное сообщение в определённой кодировке с вложенным файлом.

1. Ввести команду [*Сообщение—Создать*].

В окне *Создать сообщение* в поле *Кому:* указать электронный адрес адресата, например Ugrinovich@Lbz.ru (рис. 3.21).
В поле *Копия:* можно указать адреса получателей копии сообщения.
В поле *Тема:* указать тему сообщения, например «Пробное сообщение».

2. В области, отведённой для сообщения, ввести текст сообщения, например «Пробное сообщение (кодировка Windows)».

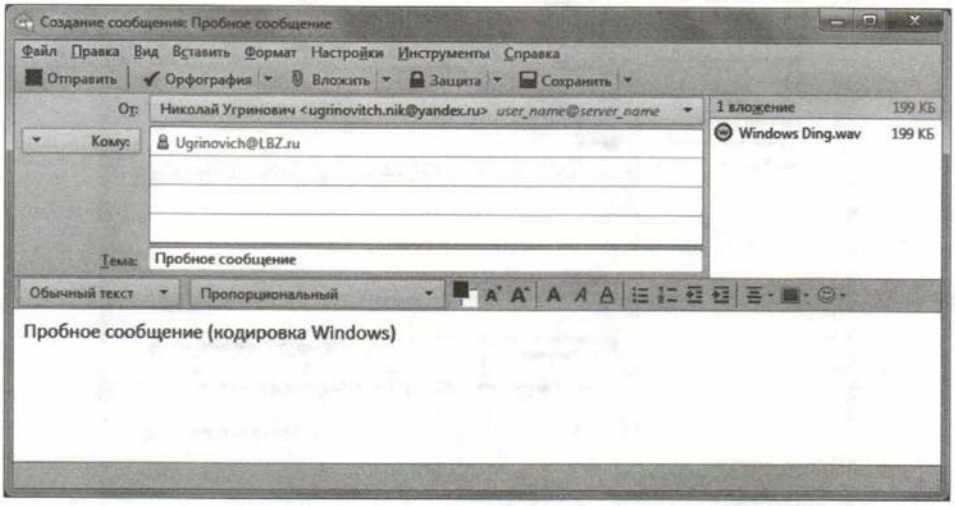


Рис. 3.21

Достаточно важен выбор правильной кодировки русских букв сообщения. При пользовании электронной почтой чаще всего используются кодировки Юникод и Windows-1251.

3. Выбор кодировки осуществить с помощью команды [*Настройки—Кодировка—Кириллица (Windows-1251)*].

В сообщение можно вставлять файлы (текстовые, графические, звуковые и т. д.).

4. Для вставки файла в сообщение ввести команду [*Файл—Вложить—Файл(ы)...*]. В появившемся окне *Вложить файлы* выбрать требуемый файл: он будет вложен в сообщение.
Вставить в сообщение, например, звуковой файл Windows Ding.wav. Название вложенного файла появится в правой части окна сообщения (см. рис. 3.21).

Если создание сообщения производилось в автономном режиме без подключения к Интернету, то сообщение автоматически сохранится в папке *Исходящие*.

5. После завершения работы над сообщением щёлкнуть по кнопке *Отправить*. Сообщение будет помещено в папку *Отправленные*.

Чтобы отправить сообщение адресату, необходимо подключиться к Интернету.

6. Если выполнить команду [*Файл—Отправить неотправленные сообщения*], то произойдёт соединение с почтовым сервером, и все сообщения, находящиеся в папке *Исходящие* на локальном компьютере, будут доставлены на почтовый сервер. Одновременно эти сообщения будут перемещены на локальном компьютере в папку *Отправленные*.
   Почтовый сервер провайдера передаст сообщения в Интернет, и через некоторое время они будут доставлены на почтовые серверы получателей. В данном случае пробное сообщение попадёт в почтовый ящик Ugrinovich@Lbz.ru.

3.6. Общение в Интернете в реальном времени

Электронная почта позволяет абонентам обмениваться сообщениями, между отправкой и получением которых проходит некоторое время. Системы мгновенных сообщений позволяют абонентам обмениваться информацией в реальном времени, т. е. практически без задержки. Увеличившаяся скорость передачи данных и возросшая производительность компьютеров позволяют не только обмениваться текстовыми сообщениями в реальном времени, но и осуществлять аудио- и видеосвязь.

Системы мгновенных сообщений существуют как для глобальной компьютерной сети Интернет, так и для локальных сетей.

Серверы общения в реальном времени. В Интернете существуют тысячи серверов, на которых реализуется общение в реальном времени. Любой пользователь может подключиться к такому серверу, зарегистрироваться на нём, и начать общение с одним из посетителей этого сервера или участвовать в коллективной встрече.

Простейший способ общения — **чат** (англ. *chat*) — это обмен сообщениями, набираемыми с клавиатуры. Абонент вводит сообщение с клавиатуры, и оно высвечивается в окне, которое одновременно видят все участники чата.

Если компьютер абонента, а также компьютеры собеседников оборудованы звуковой картой, микрофоном и наушниками или аудиоколонками, то они могут обмениваться звуковыми сообщениями. Однако «живой» разговор одновременно возможен только между двумя собеседниками.

Чтобы абоненты могли видеть друг друга, т. е. обмениваться видеоизображениями, к компьютерам должны быть подключены видеокамеры **(веб-камеры)**. Веб-камеры обычно подключаются к USB-порту. Сайт системы Skype, реализующей такое общение: **http://www.skype.com/ru**.

Конечно, качество передаваемого звука и изображения в большой мере зависит от скорости подключения компьютера к Интернету, которое должно быть не менее 1 Мбит/с.

Интернет-телефония. Интернет-телефония даёт возможность пользователю Интернета использовать телефонную связь компьютер — компьютер, компьютер — телефон и телефон — компьютер. Провайдеры интернет-телефонии обеспечивают такую связь с помощью специальных телефонных серверов Интернета, которые подключены и к Интернету, и к телефонной сети.

Всё большее распространение получают IP-телефоны (рис. 3.22), которые подсоединяются непосредственно к Интернету и не требуют компьютера для использования интернет-телефонии.

Рис. 3.22

SMS- и MMS-сообщения. Пользователи мобильных телефонов могут обмениваться SMS- и MMS-сообщениями. SMS-сообщения можно отправлять также с компьютера, подключённого к Интернету, на мобильный телефон.

Общение при помощи мобильных цифровых устройств. Кроме мобильных телефонов сегодня существует большое количество

различных мобильных цифровых устройств, таких как смартфоны (коммуникаторы), планшетные компьютеры и др. Эти устройства могут предоставлять возможности интернет-коммуникации как через сети сотовой связи (технологии 3G, 4G и др.), так и через беспроводные локальные сети (Wi-Fi, Wi-MAX).

При помощи мобильных цифровых устройств пользователи в любом месте и в любое время могут обмениваться письмами (электронная почта) и короткими текстовыми сообщениями (приложения-мессенджеры Telegram, Whatsapp, Viber и др.), а также использовать видеотелефонную связь через сети Интернет при помощи приложения Skype (при этом требуется оплата только за передачу закодированной голосовой и видеоинформации через Интернет, что обычно обходится гораздо дешевле, чем обычные междугородние, а тем более международные звонки).

Вопросы и задания

Какие формы общения в реальном времени существуют в Интернете?

Практическая работа 3.4

Общение в реальном времени в глобальной и локальных компьютерных сетях

Задание. Зарегистрироваться в системе интернет-телефонии Skype, настроить программу и реализовать звонок с компьютера на компьютер по адресу электронной почты.

Интернет-телефония в системе Skype

Для использования интернет-телефонии с помощью системы Skype необходимо подключить к компьютеру и настроить аудиооборудование (микрофон, наушники или колонки) и подключить веб-камеру к USB-порту.

1. Запустить программу интернет-телефонии Skype и зарегистрироваться.
   В окне программы ввести команду [*Инструменты—Настройки...*] и в появившемся диалоговом окне *Настройки* (рис. 3.23) выбрать из списка и настроить оборудование.

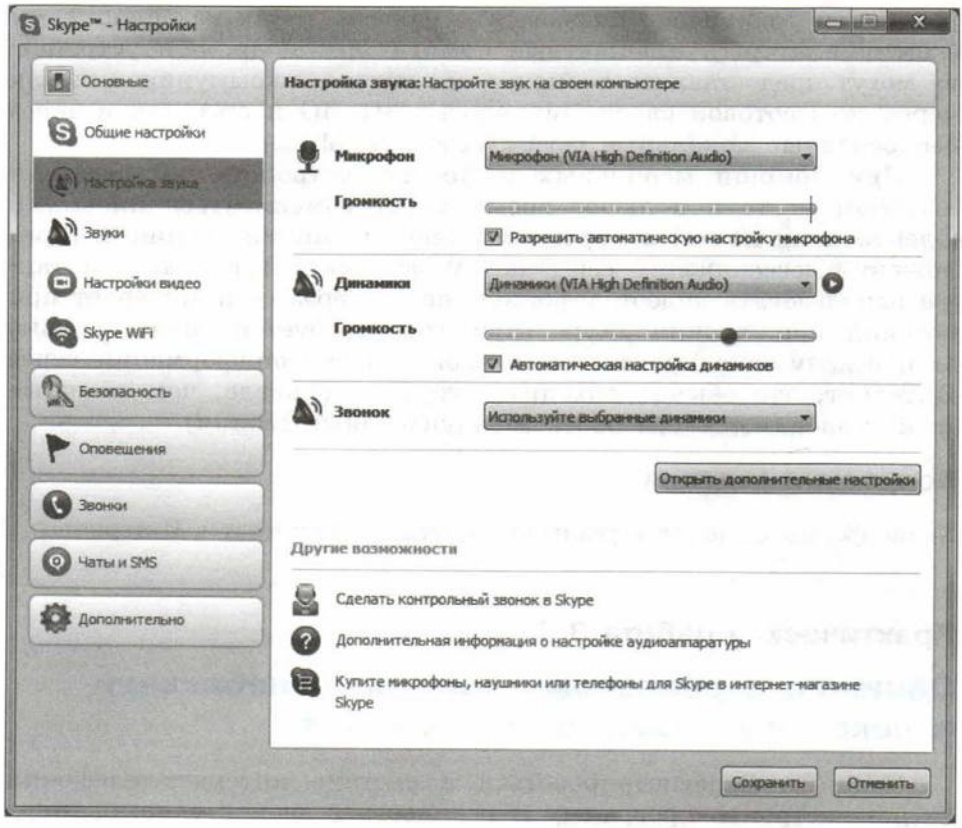


Рис. 3.23

С помощью системы Skype возможны бесплатные звонки с компьютера на компьютер (по адресу электронной почты) и платные звонки на стационарные и мобильные телефоны (по телефонному номеру).

2. В окне программы выполнить команду [*Контакты—Добавить контакт...—Поиск в справочнике Skype*] и в появившемся диалоговом окне (рис. 3.24) ввести в текстовое поле адрес электронной почты абонента.
   Щёлкнуть по кнопке *Добавить в список контактов*.

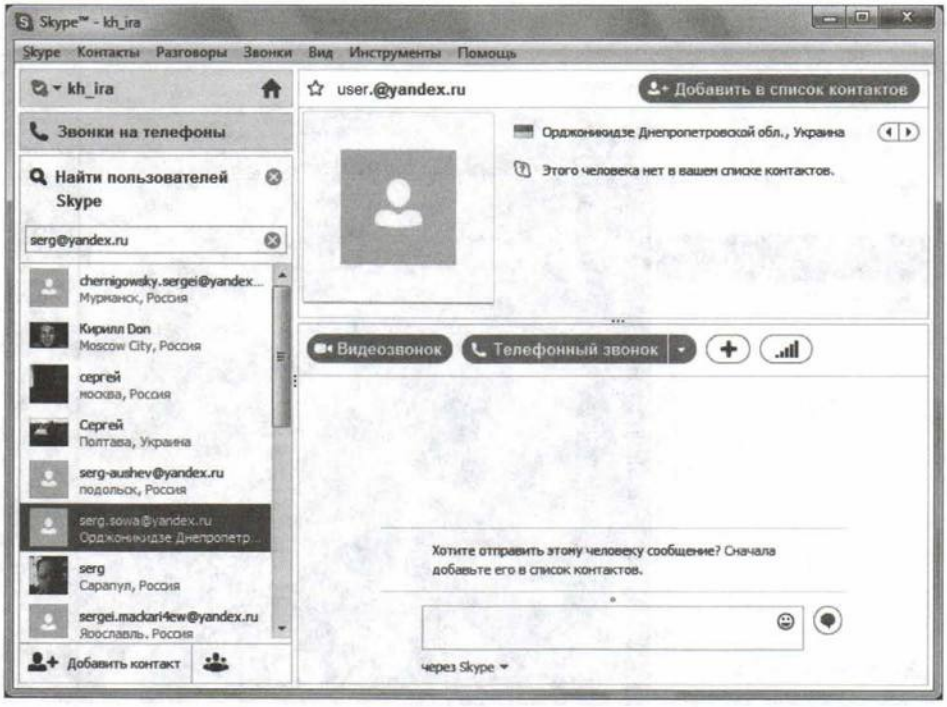


Рис. 3.24

С помощью системы Skype позвоним выбранному абоненту (будем не только разговаривать с ним, но и видеть его).

3. В окне программы на вкладке *Контакты* выбрать абонента и нажать кнопку *Видеозвонок*.

4. После соединения с абонентом в окне программы появится его видеоизображение (рис. 3.25), и можно будет начинать разговор.
   Для окончания разговора нажать кнопку *Положить трубку*.

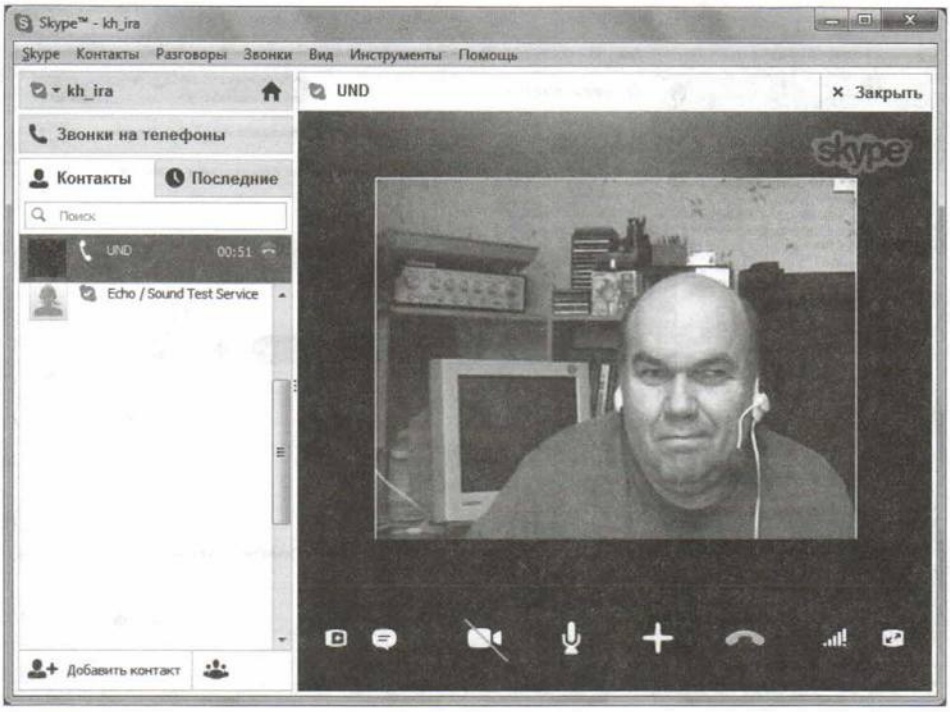


Рис. 3.25

3.7. Файловые архивы

Серверы файловых архивов. Сотни тысяч серверов Интернета являются серверами файловых архивов, и на них хранится огромное количество файлов. Файловые серверы поддерживаются многими компаниями — разработчиками программного обеспечения и производителями аппаратных компонентов компьютера и периферийных устройств. Размещаемое на таких серверах программное обеспечение является свободно распространяемым (freeware) или условно-бесплатным (shareware), поэтому, скачивая тот или иной файл, пользователь не нарушает закона об авторских правах на программное обеспечение.

Последнее время широкое распространение получили серверы музыкальных архивов, хранящие альбомы и музыкальные композиции популярных исполнителей в формате MP3.

Протоколы передачи файлов. Доступ к файлам на серверах файловых архивов возможен как по протоколу HTTP, так и по

специальному протоколу передачи файлов **FTP** (File Transfer Protocol). Протокол FTP позволяет загружать файлы (download) с удалённых серверов файловых архивов на локальный компьютер и, наоборот, производить передачу файлов (upload) с локального компьютера на удаленный веб-сервер, например, в процессе публикации веб-сайта.

Так, для загрузки с сервера файлового архива ftp.server.com файла file.exe, хранящегося в папке pub, необходимо указать URL-адрес этого файла. При указании URL-адреса файла протокол FTP записывается следующим образом: ftp://

В результате универсальный указатель ресурсов принимает вид:

ftp://ftp.server.com/pub/file.exe

Он состоит из трёх частей:

ftp://	— протокол доступа;
ftp.server.com	— доменное имя сервера файлового архива;
/pub/file.exe	— путь к файлу и имя файла.

FTP-клиенты. Серверы, с которыми может производиться обмен файлами по протоколу FTP, называются FTP-серверами. FTP-серверы по своему функциональному назначению могут являться как серверами файловых архивов, так и веб-серверами, на которых размещаются веб-сайты. Обмен файлами (загрузка и передача) с серверами файловых архивов и веб-серверами производится с помощью специализированных программ — FTP-клиентов (например, FTP-клиент входит в состав **файлового менеджера** Total Commander).

Доступ к серверам файловых архивов для загрузки файлов на локальный компьютер обычно является анонимным и не требует ввода имени пользователя и пароля. Наоборот, доступ к веб-серверам с целью передачи файлов на удалённый сервер в процессе публикации веб-сайта требует идентификации пользователя, т. е. ввода имени пользователя и пароля.

FTP-клиент включает *менеджер сайтов*, позволяющий создать список серверов, с которыми планируется работа. FTP-клиент представляет в удобном для пользователя виде каталоги локального и удалённого компьютеров, обеспечивает продолжение загрузки файла после обрыва соединения и т. д. В процессе передачи файла отображается необходимая информация: процент переданного объёма файла, скорость передачи, оставшееся время и др.

Загрузка файлов с помощью браузера. Для удобства пользователей многие серверы файловых архивов (freeware.ru, www.freesoft.ru, www.download.ru и др.) имеют веб-интерфейс, что позволяет работать с ними с использованием браузеров.

Браузеры являются интегрированными системами для работы с различными информационными ресурсами Интернета и поэтому включают **менеджеры загрузки файлов**.

После активизации ссылки на файл в открывшемся окне требуется указать папку на локальном компьютере, в которой файл должен быть сохранён. Начнётся загрузка файла, процесс которой отображается на информационной панели (скорость передачи, объём загруженной и оставшейся частей файла и т. д.).

Файлообменник (файловый хостинг) — служба, предоставляющая пользователю место под его файлы и круглосуточный доступ к ним. Такой сервис позволяет удобно обмениваться файлами. Примеры файлообменников: Облако Mail.Ru, Яндекс.Диск, файлообменник.рф, Microsoft OneDrive, Google Диск.

Вопросы и задания

1. Что такое протокол FTP?
2. Какие вы знаете способы загрузки файлов с FTP-серверов?

Практическая работа 3.5

Работа с файловыми архивами

Задание 1. Загрузить файл с локального компьютера на удалённый сервер — файловый обменник (файлообменник) с помощью браузера.

Задание 2. Загрузить файл на локальный компьютер с файлообменника.

Передача файлов на удалённый сервер с локального компьютера с помощью браузера

1. Открыть в браузере коммуникационный портал mail.ru.
2. Войти на портал под своим логином и паролем.
3. Перейти к сервису *Облако*. Для этого в меню портала mail.ru выполнить команду [*Все проекты—Облако*] (рис. 3.26).
4. В окне Облако mail.ru щёлкнуть на кнопке *Загрузить* (рис. 3.27).

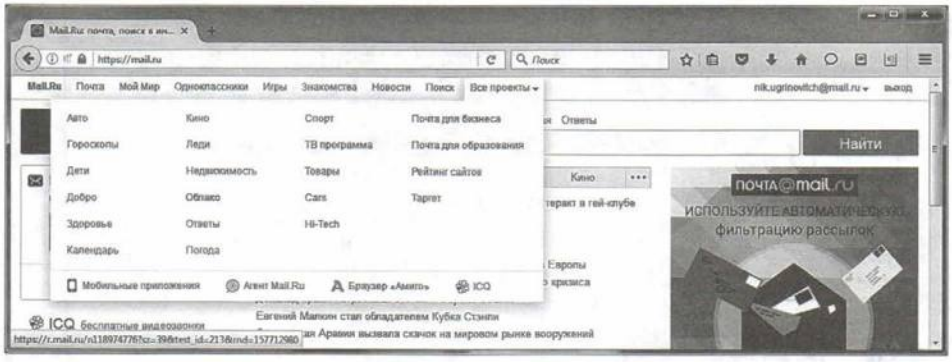


Рис. 3.26

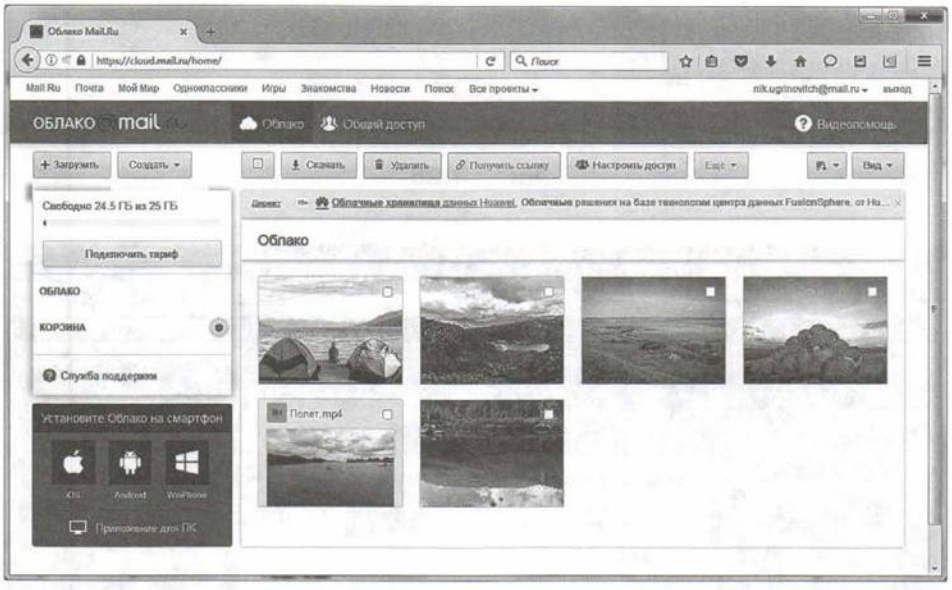


Рис. 3.27

5. В диалоговом окне *Загрузить в облако* нажать кнопку *Выбрать файлы*.
6. В диалоговом окне *Выгрузка файла* указать файл, который требуется загрузить на файлообменник (рис. 3.28).
7. Для начала загрузки щёлкнуть по кнопке *Открыть*.
   Процесс загрузки файла на файлообменник занимает некоторое время.
8. Для получения ссылки на загруженный файл выделить файл, установив галочку в правом верхнем углу, и щёлкнуть по кнопке *Получить ссылку*.
   Откроется поле со ссылкой на файл (рис. 3.29).

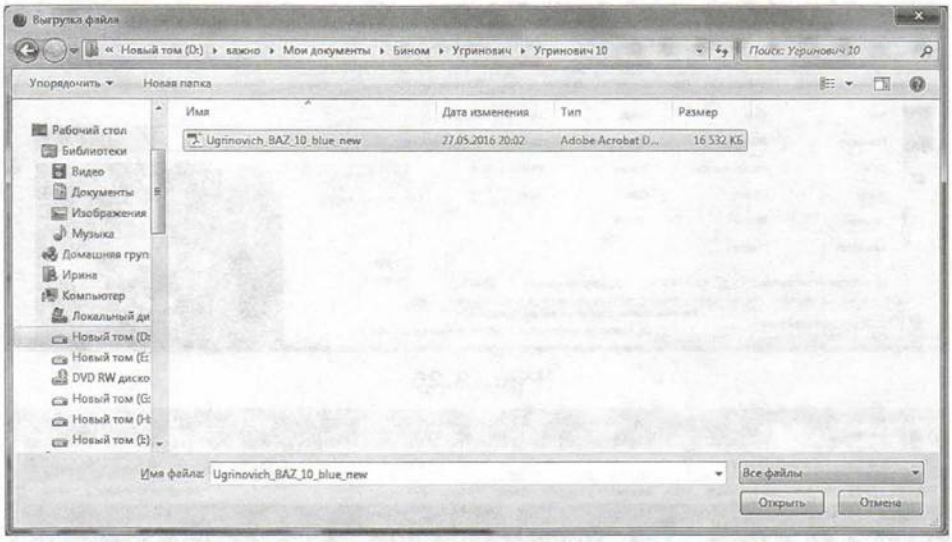


Рис. 3.28

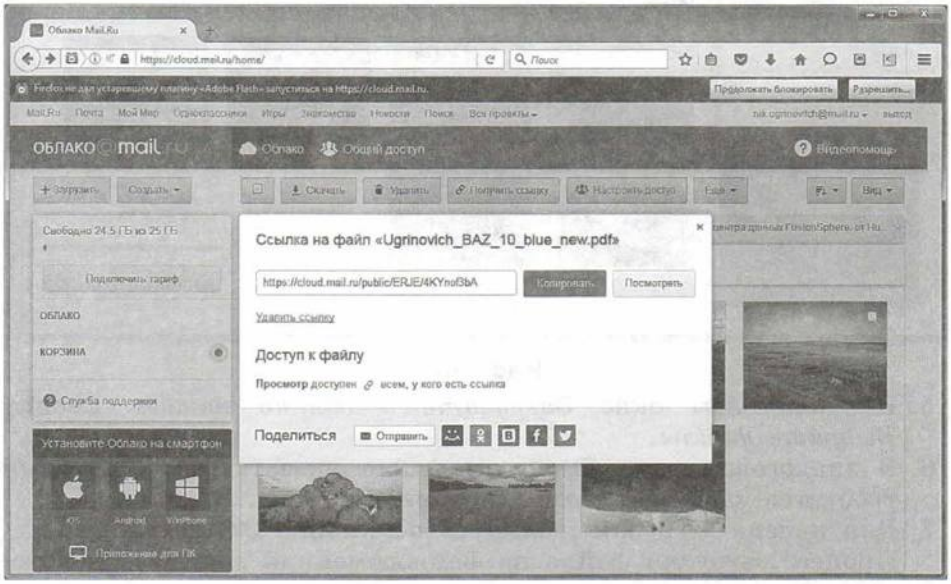


Рис. 3.29

9. Скопировать ссылку и отправить её в письме своему соседу по парте (рис. 3.30).

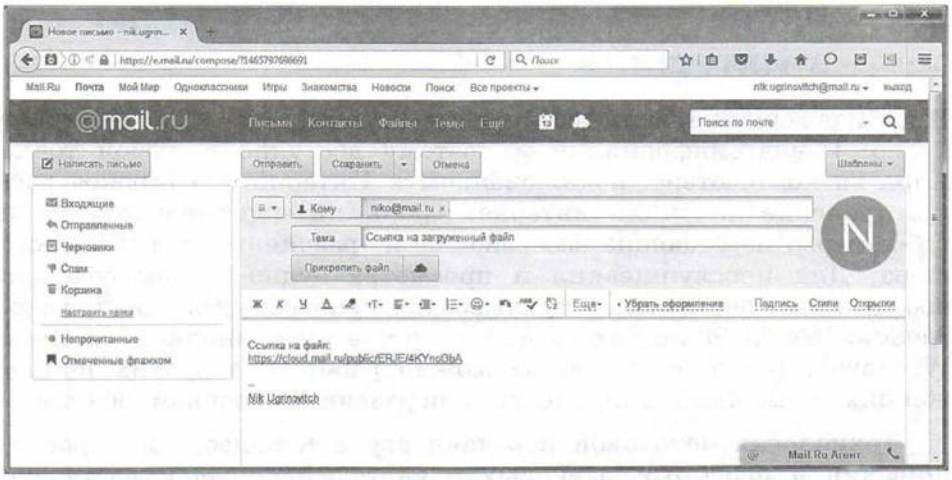


Рис. 3.30

Загрузка файлов на локальный компьютер с файлообменника

Перейти по ссылке, полученной в письме от вашего соседа по парте. Откроется страница браузера, на которой отображается содержимое файла (рис. 3.31). Нажать кнопку *Скачать* и загрузить файл на свой компьютер.

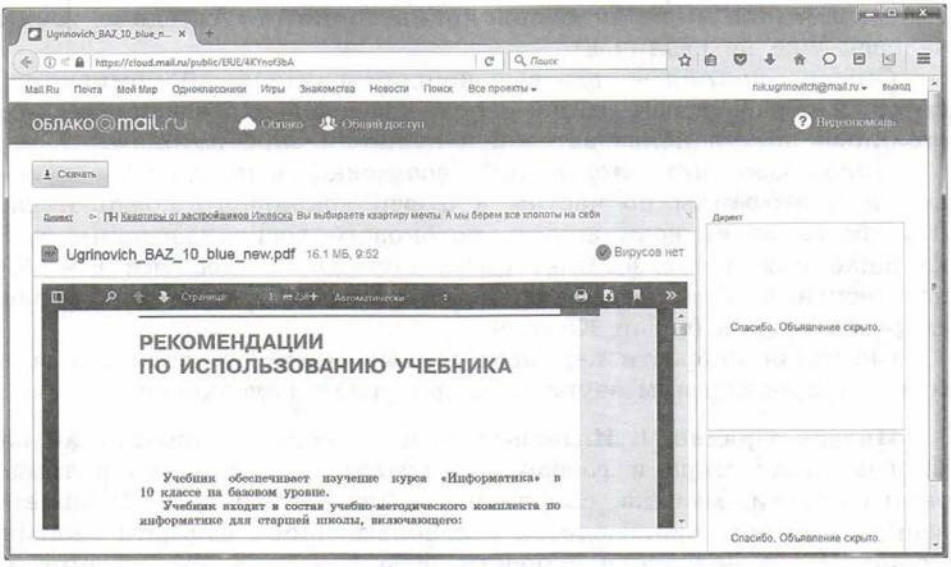


Рис. 3.31

3.8. Радио, телевидение и веб-камеры в Интернете

Интернет-вещание включает в себя передачу по сети Интернет аудио- и видеоинформации и поэтому доступно в любой точке мира на компьютере, подключённом к Интернету. Широкое распространение получили интернет-радио, интернет-телевидение и веб-камеры, передающие изображение из различных точек земного шара. Для прослушивания и просмотра аудио- и видеофайлов необходимы специальные мультимедиа проигрыватели (например, Windows Media Player, входящий в состав операционной системы Windows). В процессе приёма можно работать в других приложениях и слушать или смотреть передачи в фоновом режиме.

Технология потоковой передачи звука и видео. Для прослушивания и просмотра звуковых и видеофайлов непосредственно в процессе их получения из Интернета в режиме реального времени была разработана технология потокового сжатия, передачи и воспроизведения звуковых и видеоданных.

Принцип *сжатия* основан на удалении психофизиологической избыточности передаваемой звуковой или видеоинформации, т. е. на удалении некоторых частот в звуковом или видеосигнале, которые человек всё равно не воспринимает. Например, если воспроизводится громкий звук на частоте 1000 Гц, то более слабый звук на частоте 1100 Гц уже не будет слышен человеку; если в изображении имеются очень яркие точки, то соседние точки человек уже не различает.

Сервер вещания — это высокопроизводительный компьютер, который обрабатывает запросы пользователей и обеспечивает постоянный доступ пользователей к аудио- и видеопотокам.

Потоковые технологии дают возможность передавать звуковые и видеофайлы по частям в буфер локального компьютера. Это обеспечивает возможность потокового воспроизведения звука даже при использовании низкоскоростного (десятки Кбит/с) подключения. Для потоковой передачи видео требуется бóльшая скорость канала (сотни Кбит/с).

Снижение скорости передачи данных может приводить к временным пропаданиям звука или пропускам видеокадров.

Интернет-радио. В Интернете вещают сотни радиостанций из многих стран мира и городов, придерживаясь различной тематики (новости, музыка, спортивные репортажи и др.). Интернет-вещание обычно производится с использованием формата сжатия звука MP3 в диапазоне скорости передачи от 8 до 224 Кбит/с и, соответственно, с худшим или лучшим качеством звука.

Ссылки на сайты интернет-радио можно найти в Интернете. Пример: **http://www.moreradio.org/online_radio.php?Countries.**

Интернет-телевидение. Интернет-вещание может быть как прямым («живое вещание»), так и в записи («видео по запросу»). Интернет-телевидение может дублировать вещание традиционного телевидения либо представлять оригинальные интернет-программы. Во время «живого вещания» пользователь имеет возможность в реальном времени взаимодействовать с продюсером передачи: получать от него дополнительную информацию, задавать вопросы участникам программы, участвовать в интерактивных опросах, влиять на развитие программы и т. п.

При использовании «видео по запросу» пользователь может выстроить сетку вещания под себя и посмотреть передачу или фильм в удобное для него время.

Веб-камеры. Широкой популярностью пользуются веб-камеры, установленные в самых разных местах (на улицах городов, в музеях, в заповедниках и т. д.) по всему миру и непрерывно передающие изображение. Одна из таких веб-камер установлена на 13-м этаже главного здания МГУ им. М. В. Ломоносова на Воробьевых горах (**http://webcam.mnc.ru**), откуда открывается панорама Москвы (рис. 3.32).

Рис. 3.32

Пользователь даже может управлять камерой через Интернет: поворачивать её вокруг горизонтальной и вертикальной осей, приближать или удалять изображение и т. д.

Веб-камеры могут использоваться для обеспечения безопасности дома или офиса. Камера подключается к компьютеру или локальной сети, имеющим широкополосное соединение с Интернетом, и передаёт изображение в Интернет. Пользователь может с помощью браузера наблюдать за своим офисом или квартирой, находясь за тысячи километров от них. Камера может быть оборудована датчиком движения, который подаёт сигнал тревоги при обнаружении какого-либо движения в помещении.

Вопросы и задания

1. На чём основан принцип сжатия аудио- и видеоданных при технологии потоковой передачи?
2. Как могут использоваться веб-камеры?

3.9. Геоинформационные системы в Интернете

Географические информационные системы (ГИС). Геоинформационные системы предназначены для сбора, хранения, анализа и графической визуализации географических данных. Геоинформационные системы включают растровые или векторные карты, а также данные о географических объектах, хранящиеся в базах данных. Таким образом, ГИС позволяют пользователям искать, анализировать и редактировать цифровые карты, а также дополнительную информацию об объектах (например, адрес здания, высоту объекта над уровнем моря и т. д.).

Интерактивные карты в Интернете. В Интернете можно найти интерактивные карты мира, стран и городов. Для нас наибольший интерес представляют интерактивные карты России и российских городов (рис. 3.33). Интерактивной картой можно управлять: увеличивать и уменьшать масштаб и сдвигать её по всем географическим направлениям (север, юг, запад, восток), чтобы просмотреть участки карты за пределами краев экрана. Пример сайта интерактивных карт: **http://www.eatlas.ru**.

Картографический ресурс Google Earth. Этот ресурс представляет собой программу-навигатор Google Earth, устанавливаемую на локальный компьютер, и удалённую, находящуюся на серверах в

сети Интернет, базу географических данных. Основу этих данных представляют спутниковые снимки и топографические карты. Для визуализации изображения используется трёхмерная модель всего земного шара (с учётом высоты над уровнем моря), которая отображается на экране компьютера.

Программа-навигатор Google Earth позволяет изменять масштаб изображения (увеличение, уменьшение), осуществлять сдвиг по осям (вверх, вниз, вправо, влево) и поворачивать изображение. Можно получить координаты любой точки земной поверхности, а также её высоту над уровнем моря.

Рис. 3.33

Практически вся поверхность суши визуализируется изображениями, имеющими разрешение 15 м/пиксель. Есть и отдельные участки поверхности (как правило, столицы и некоторые крупные города), имеющие более подробное разрешение. Например, Москва снята с разрешением 0,6 м на пиксель (рис. 3.34).

По желанию пользователя можно отобразить на земной поверхности названия населённых пунктов, водоёмов, аэропортов, автомобильные и железные дороги и другую информацию. Имеется также функция измерения расстояния между точками на земной поверхности.

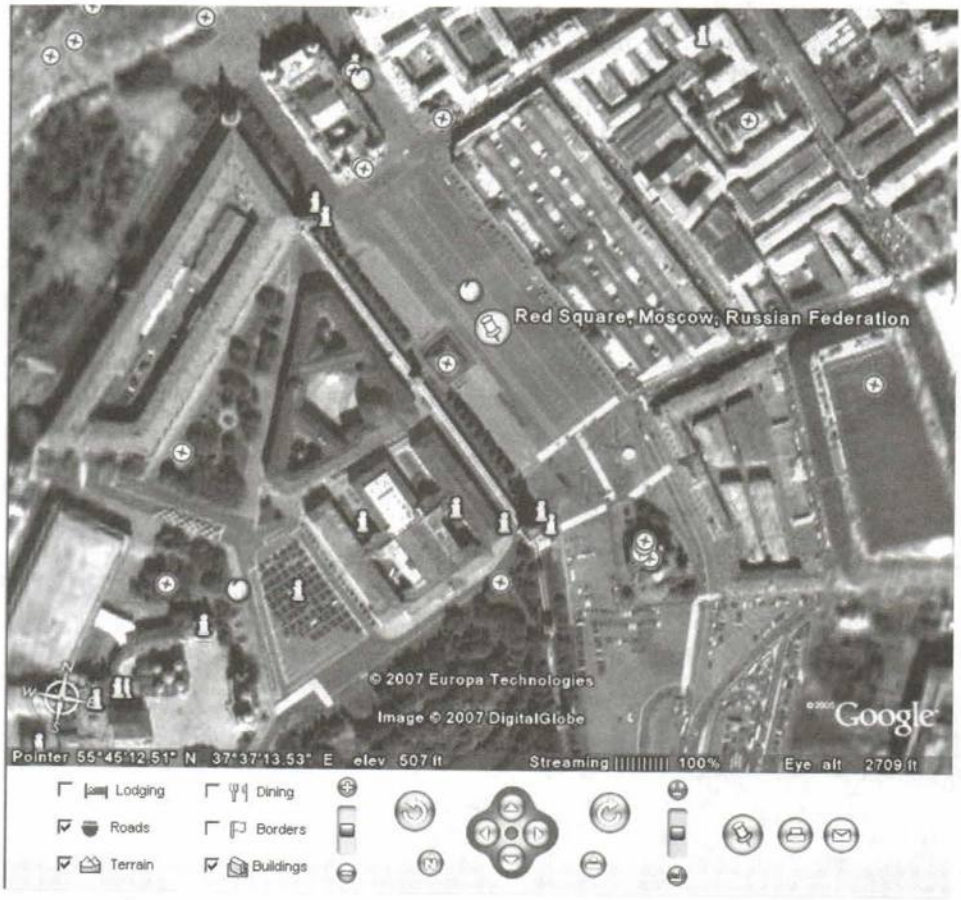


Рис. 3.34

Спутниковая навигация. Для определения географических координат точки, в которой находится пользователь, используются данные, полученные с помощью радиосигналов со спутников. Основой системы должны являться 24 спутника (рис. 3.35), причём для определения пространственных координат и точного времени

требуется принять и обработать навигационные сигналы не менее чем от 4 спутников.

В настоящее время существуют две системы глобальной спутниковой навигации: американская **GPS** и российская **ГЛОНАСС**. Максимальная точность измерения военных навигаторов составляет несколько метров, обычная точность гражданских моделей навигаторов составляет несколько десятков метров.

Существуют GPS-приёмники, имеющие собственный процессор для необходимых расчётов и дисплей для отображения информации, а также GPS-приставки к смартфонам и ноутбукам. Кроме географических координат и точного времени GPS-приемник способен показать текущее положение на электронной карте местности (рис. 3.36).

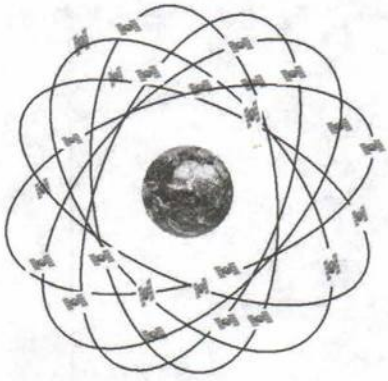


Рис. 3.35

Рис. 3.36

Вопросы и задания

1. В чём состоит отличие географических информационных систем от обычных географических карт?
2. Что можно определить с использованием спутниковой навигации?

Практическая работа 3.6

Геоинформационные системы в Интернете

Задание 1. Найти в Интернете интерактивную карту вашего города и на ней ваш район.

Задание 2. С помощью картографической системы Google Earth найти ваш город и ваш район.

Варианты выполнения работы:

- различные города и районы.

Просмотр интерактивной карты с помощью браузера

Найдём в Интернете интерактивную карту вашего города и на ней ваш район.

1. Запустить браузер и ввести адрес сайта с интерактивными картами (например, **http://www.eatlas.ru**).
   Выбрать интерактивную карту города (например, Санкт-Петербурга).
   С помощью системы управления найти определённый район города (например, Петропавловскую крепость — рис. 3.37).

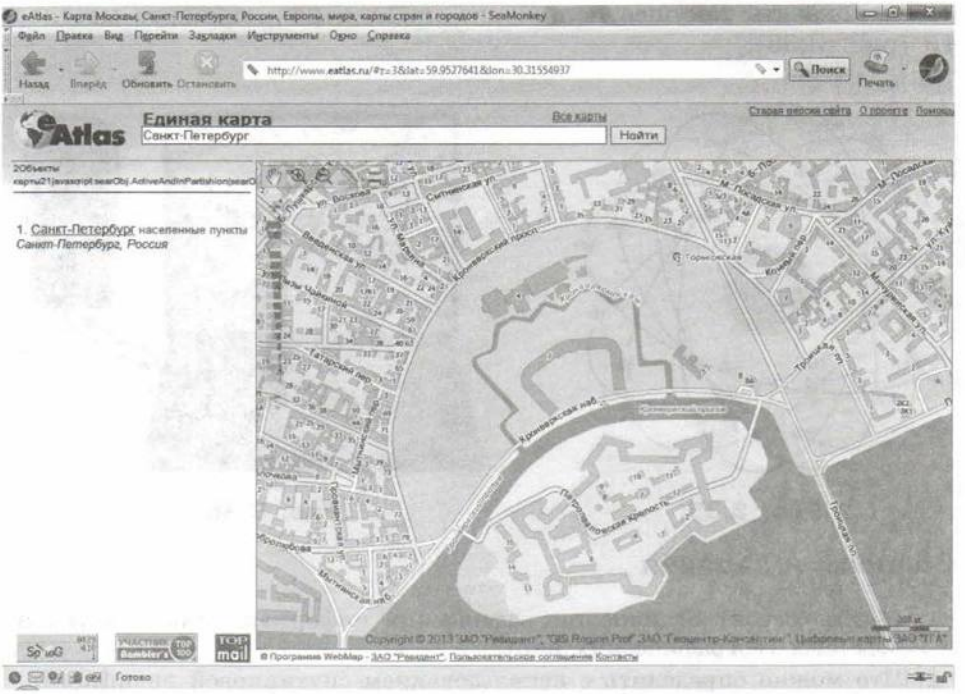


Рис. 3.37

Просмотр участка земной поверхности с помощью картографической системы Google Earth

Найдём в картографической системе Google Earth (**http://www.google.com/earth/index.html**) ваш город и ваш район.

2. Запустить программу-навигатор Google Earth и с помощью системы управления найти ваш город (например, Санкт-Петербург).
   Найти и приблизить определённый район города (например, Петропавловскую крепость — рис. 3.38).

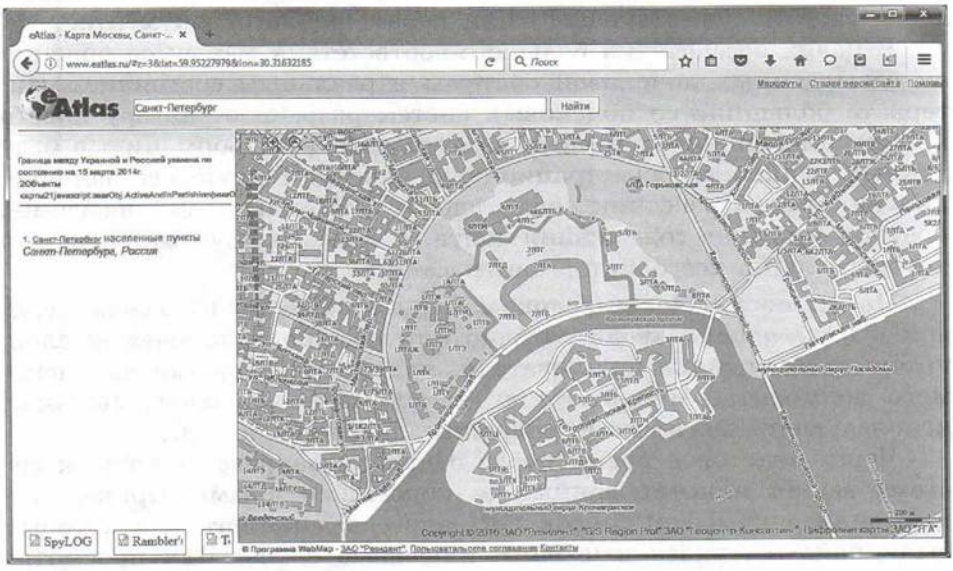


Рис. 3.38

3.10. Поиск информации в Интернете

Для поиска информации в Интернете используются специальные поисковые системы, которые содержат в базах данных постоянно обновляемую информацию о веб-сайтах, файлах и других информационных ресурсах Интернета. Разные поисковые серверы могут использовать различные механизмы поиска, хранения и предоставления пользователю информации.

Поисковые системы общего назначения. Интерфейс поисковых систем общего назначения содержит **поле поиска** и список **разделов каталога**.

Заполнение баз данных поисковых систем. Заполнение баз данных поисковой системы осуществляется с помощью специальных программ-роботов, которые периодически «обходят» веб-серверы Интернета. Программы-роботы читают все встречающиеся документы, выделяют в них ключевые слова и заносят в базу данных, содержащую URL-адреса документов.

Так как информация в Интернете постоянно меняется (создаются новые веб-сайты и страницы, удаляются старые, меняются их URL-адреса и т. д.), поисковые роботы не всегда успевают отследить все эти изменения. Информация, хранящаяся в базе данных поисковой системы, может отличаться от реального состояния Интернета, и тогда пользователь в результате поиска может получить адрес уже не существующего или перемещённого документа.

В целях обеспечения большего соответствия между содержанием базы данных поисковой системы и реальным состоянием Интернета большинство поисковых систем разрешают автору нового или перемещенного веб-сайта самому внести информацию в базу данных, заполнив регистрационную анкету. В процессе заполнения анкеты разработчик сайта вносит URL сайта, его название, краткое описание содержания сайта, а также ключевые слова, по которым легче всего будет найти сайт.

Поиск по ключевым словам. При поиске по ключевым словам *в поле поиска* вводится одно или несколько ключевых слов, которые, по мнению пользователя, являются главными для искомого документа. Можно также использовать сложные запросы, использующие логические операции, шаблоны и т. д.

Через некоторое время после отправки запроса поисковая система вернёт аннотированный (с короткими комментариями содержания документа) список URL-адресов документов, в которых были найдены указанные вами ключевые слова. Для просмотра этого документа в браузере достаточно активизировать указывающую на него ссылку.

Если ключевые слова были выбраны неудачно, то список URL-адресов документов может быть слишком большим (содержать десятки и даже сотни тысяч ссылок). Чтобы уменьшить список, можно в поле поиска ввести дополнительные ключевые слова или воспользоваться каталогом поисковой системы.

Язык построения запросов. Поиск по одному ключевому слову всегда производится однозначно (ключевое слово либо имеется в рассматриваемом тексте, либо отсутствует). При поиске же по нескольким ключевым словам возможны различные способы их комбинирования:

- в искомом тексте должны присутствовать все заданные ключевые слова («И то, И другое»);
- достаточно, чтобы в искомом тексте имелось хотя бы одно из заданных ключевых слов («то ИЛИ другое»);
- в тексте должно иметься одно ключевое слово, но обязательно должно отсутствовать другое («то, но НЕ другое»);
- ключевые слова составляют фразу, которая обязательно должна присутствовать в тексте «как есть»;

- ключевые слова обязательно должны быть взаимосвязаны (т. е. должны располагаться близко друг от друга, — например, в фразе могут быть разделены союзом или прилагательным).

Чтобы «объяснить» поисковой системе, как нужно понимать заданную последовательность ключевых слов, используется особый **язык построения запросов**. Принципы построения языка запросов, как правило, универсальны для большинства поисковых систем, но для той или иной системы язык запросов может иметь свои особенности.

Пример языка построения запросов в поисковой системе Яндекс:

- *заяц & кролик* — поиск текстов, в которых есть оба заданных ключевых слова (и «заяц», и «кролик»);
- *заяц | кролик* — поиск текстов, в которых имеется хотя бы одно из заданных ключевых слов («заяц» или «кролик»);
- *заяц –кролик* — знак минуса перед словом «кролик» предписывает искать только такие тексты, в которых есть слово «заяц», но нет слова «кролик»;
- *заяц +кролик* — знак плюса перед словом «кролик» предписывает искать только такие тексты, в которых есть слово «заяц» и обязательно есть слово «кролик»;
- *"братец Кролик"* — ключевые слова, заключённые в кавычки, ищутся в тексте как цитата (обязательная фраза);
- *кровать /2 диван* — ключевые слова в искомом тексте могут следовать в любом порядке и должны быть разделены не более чем одним любым словом (например, это может быть текст «диван и кровать» или «кровать, тахта, диван»).

Кроме того, большинство современных поисковых систем являются «интеллектуальными» — при поиске учитываются все возможные формы (падежи, склонения, спряжения, единственное и множественное число) заданных ключевых слов. Например, при задании ключевого слова *заяц* поисковая система будет искать все варианты ключевых слов — «заяц», «зайца», «зайцами» и т. п.

Поиск в иерархической системе каталогов. Каталоги составляются редакторами, просматривающими каждый новый сайт до его включения в иерархическую систему каталогов. Каталоги обычно организованы в соответствии с предметной классификацией.

Поиск информации в каталоге сводится к выбору определённого каталога, после чего пользователю будет представлен список ссылок на URL-адреса наиболее посещаемых и важных веб-сайтов. Каждая ссылка обычно аннотирована.

Современные поисковые системы. Одной из наиболее полных и мощных поисковых систем является Google (**www.google.ru**).

В Рунете (российской части Интернета) обширные базы данных имеет поисковая система Яндекс (**www.yandex.ru**).

Сайты в базе данных ранжируются по количеству их посещений в день, неделю или месяц. Посещаемость сайтов определяется с помощью специальных счётчиков, которые могут быть установлены на сайте. Счётчики фиксируют каждое посещение сайта и передают информацию о количестве посещений на сервер поисковой системы.

Современные поисковые системы часто являются **информационными порталами**, которые предоставляют пользователям не только возможности поиска документов в Интернете, но и доступ к другим информационным ресурсам (к новостям, к информации о погоде, о валютном курсе, к интерактивным географическим картам и т. д.). Пример: Российский образовательный портал: **http://edu.ru**.

Вопросы и задания

1. Каким образом наполняются базы данных поисковых систем?
2. В каких случаях активизация найденной с помощью поисковой системы ссылки на документ может выдавать сообщение об ошибке?

Практическая работа 3.7

Поиск в Интернете

С использованием поисковых систем Google, Яндекс и Mail.Ru найти веб-сайты, содержащие энциклопедии.

Задание 1. Осуществить поиск по ключевым словам.

Задание 2. Осуществить поиск в иерархической системе каталогов.

Варианты выполнения работы:

- поиск различных сайтов.

Поиск информации в Интернете

Для поиска воспользуемся поисковыми системами, а в качестве ключевого слова укажем, например, «энциклопедия».

1. Для поиска в системе Google в адресной строке браузера ввести адрес **http://www.google.ru**.

   В поле поиска поисковой системы ввести ключевое слово: «энциклопедия».

   Через определённое время (в нашем примере 0,13 с) будет выведен список ссылок (в нашем примере 30 600 000) на веб-страницы, содержащие заданное ключевое слово (рис. 3.39).

   Внимание: результаты поиска могут изменяться со временем.

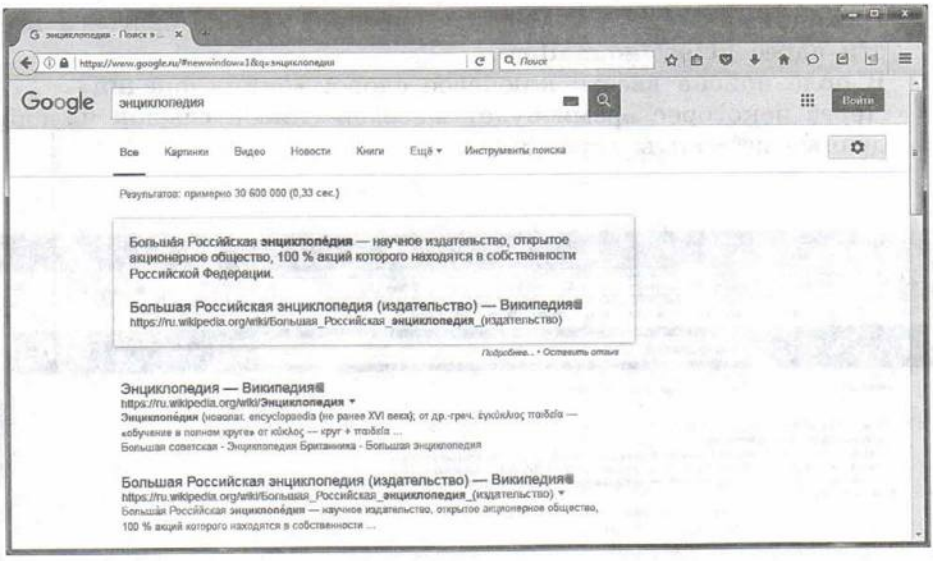


Рис. 3.39

2. Для поиска в системе Яндекс в адресной строке браузера ввести адрес **http://www.yandex.ru**.
   В поле поиска ввести ключевое слово: «энциклопедия».
   Через определённое время будет выведен список ссылок на сайты, содержащие заданное ключевое слово (рис. 3.40).

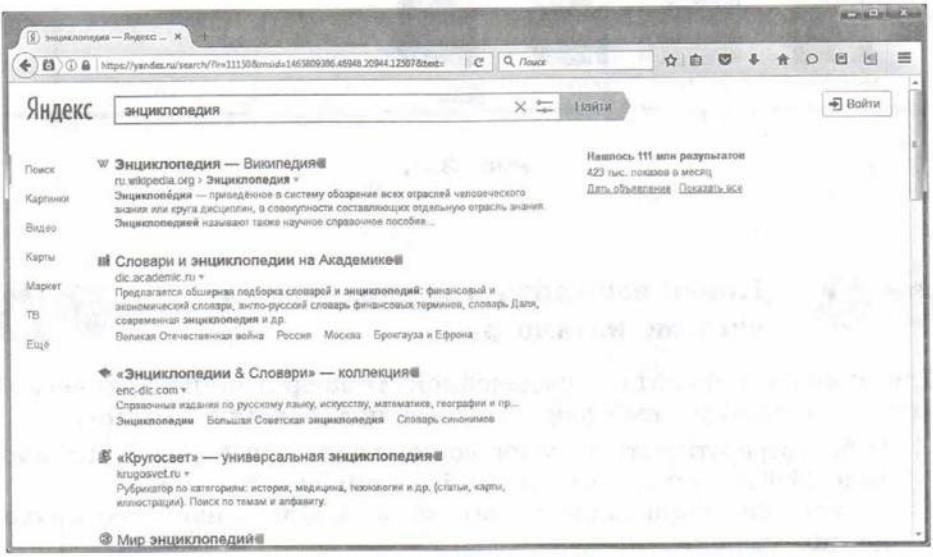


Рис. 3.40

3. Для поиска в системе Mail.ru в адресной строке браузера ввести адрес **http://go.mail.ru**.
   В поле поиска ввести ключевое слово: «энциклопедия».
   Через некоторое время будет выведен список ссылок на найденные веб-сайты (рис. 3.41).

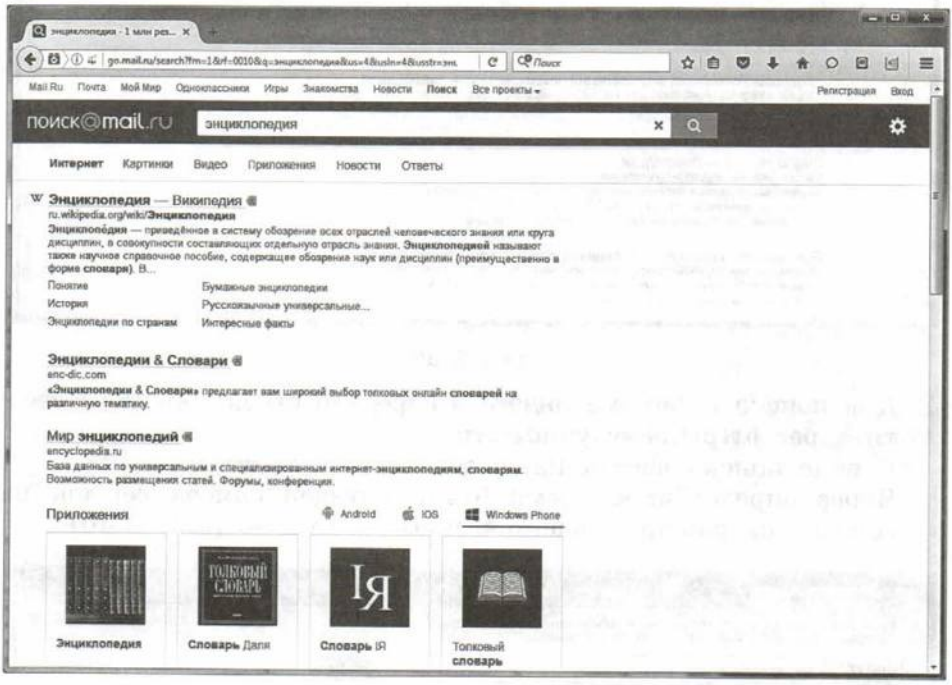


Рис. 3.41

Поиск веб-сайта в иерархической системе каталогов

Для поиска веб-сайта определённой тематики в иерархической системе каталогов выберем наиболее подходящий каталог.

1. В браузере открыть каталог социальной сети Мой мир на портале Mail.ru по адресу **http://list.mail.ru** (рис. 3.42).
   В иерархической системе каталогов выбрать наиболее подходящий каталог.

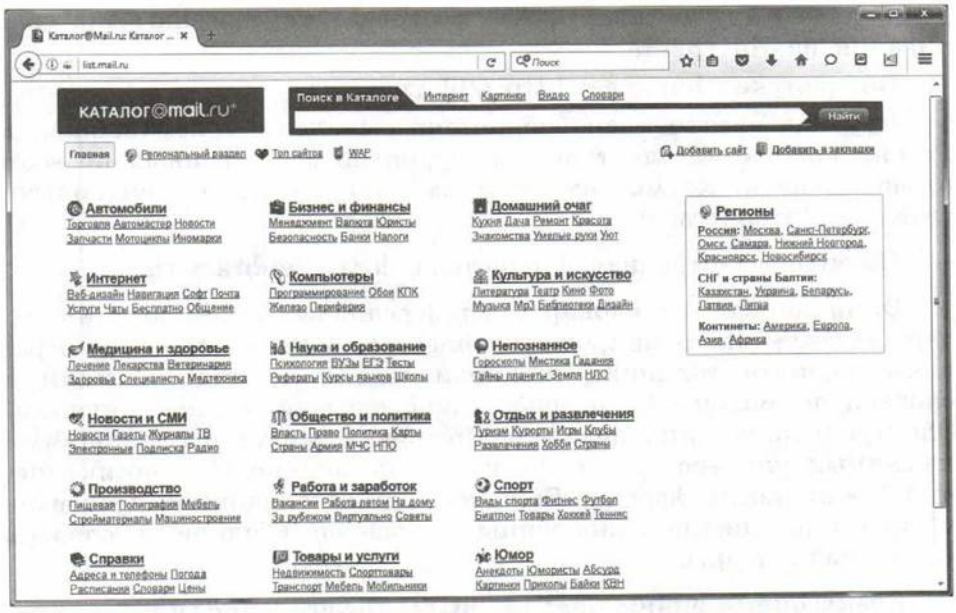


Рис. 3.42

3.11. Библиотеки, энциклопедии и словари в Интернете

Библиотеки. Электронные библиотеки в Интернете содержат электронные (цифровые) копии печатных книг, диссертаций и других документов. Для этого наиболее часто используются форматы файлов PDF, DjVu, FB2; реже — текстовые форматы TXT, RTF, DOC или формат веб-страниц HTML.

Публичная Российская государственная библиотека хранит электронные версии наиболее значительных произведений мировой и русской литературы. Фонд включает электронные копии книг, журналов, карт, нот, изобразительных материалов, диссертаций и авторефератов диссертаций (с разрешения авторов) по различным отраслям знаний. Документы находятся в свободном доступе через Интернет.

Российская государственная библиотека: **http://www.rsl.ru**

Библиотека Максима Мошкова — крупнейшая и старейшая в России электронная библиотека художественной литературы (с ноября 1994 г.). Она хранит прозу, поэзию, фантастику,

детективы и другие жанры. Все произведения доступны для скачивания из Интернета.

Библиотека Мошкова: **http://lib.ru**

Научная электронная библиотека elibrary.ru — крупнейшая в России коллекция электронных журналов и баз данных по всем отраслям наук. Ко многим ресурсам доступ открыт всем пользователям Интернета.

Научная электронная библиотека: **http://elibrary.ru**

Энциклопедии и словари. Универсальные энциклопедии содержат сведения о природе и обществе, а также по всем отраслям науки и техники. Специализированные энциклопедии и словари посвящены какой-либо одной отрасли науки и техники. Электронные энциклопедии в Интернете могут быть копиями известных универсальных печатных энциклопедий (Энциклопедия Брокгауза и Эфрона, Большой Энциклопедический словарь и др.) или специализированных словарей (Толковый словарь В. И. Даля и др.).

Электронные энциклопедии: **http://dic.academic.ru**

Существуют также оригинальные электронные энциклопедии. Википедия — это проект свободной многоязычной энциклопедии, в которой каждый может изменить или дополнить любую статью или создать новую.

Википедия: **http://ru.wikipedia.org**

Современная энциклопедия «Кругосвет» содержит новейшие знания по всем отраслям науки и техники. Много интересного в ней можно найти и по проблемам информатики.

Энциклопедия «Кругосвет»: **http://www.krugosvet.ru**

Переводчики и словари. Онлайновые мультиязычные переводчики позволяют переводить тексты, набранные в окне перевода или скопированные из буфера обмена, либо даже целые веб-страницы, включая гиперссылки, с сохранением исходного форматирования, и электронные письма.

Онлайновый переводчик ПРОМТ: **http://www.translate.ru**

Интернет-версии электронных словарей позволяют получить точный и достоверный перевод слов с английского, немецкого, французского, итальянского и испанского языков на русский, и обратно. Словарные статьи содержат транскрипцию (для английского языка), все варианты перевода, примеры использования и устойчивые словосочетания. Онлайновые словари включают как

общие, так и тематические словари, которые помогут подобрать адекватный перевод для специализированных терминов.

Онлайновый словарь Lingvo: **http://www.lingvo.ru**.

Вопросы и задания

1. Чем отличается электронная библиотека от электронной энциклопедии?
2. В чём особенность ресурса Википедия?

3.12. Электронная коммерция в Интернете

Электронная коммерция в Интернете — это коммерческая деятельность в сфере рекламы и распространения товаров и услуг посредством использования сети Интернет. В настоящее время электронная коммерция быстро развивается.

Хостинг. Одной из самых быстроразвивающихся областей электронной коммерции является хостинг (от английского слова *host* — сервер), т. е. услуги по размещению информации во Всемирной паутине. Хостинг включает в себя выделение дискового пространства для размещения веб-сайтов на веб-сервере, предоставления к ним доступа по каналу связи с определённой пропускной способностью, а также прав администрирования сайта.

Реклама. Важной составляющей электронной коммерции является информационно-рекламная деятельность.

Многие фирмы размещают на своих веб-сайтах в Интернете важную для потребителя информацию (описание товаров и услуг, их стоимость, адрес фирмы, телефон и e-mail, по которым можно сделать заказ и др.). Существуют специализированные серверы, предоставляющие потребителю систематизированную (по видам товара, производителям, ценам и др.) информацию об определённой группе товаров.

Реклама в Интернете реализуется с помощью баннеров (от английского слова *banner* — «рекламный заголовок»). В Интернете баннер представляет собой небольшую прямоугольную картинку, на которой размещается реклама продукта или веб-сайта.

Баннеры могут быть как статическими (показывается одна и та же картинка), так и динамическими (картинки постоянно меняются). Щелчок по баннеру мышью приводит к переходу на соответствующий сайт или страницу, где можно более подробно узнать о том, что рекламирует баннер.

Баннеры размещаются на сайтах либо на платной основе, либо путём обмена. Использование системы обмена баннерами, которая связывает между собой множество сайтов и позволяет им рекламировать друг друга, повышает посещаемость каждого из них.

Доски объявлений. Широкое распространение в Интернете получила электронная торговля. Простейшим её вариантом является виртуальная доска объявлений, где продавцы и покупатели просто обмениваются информацией о предлагаемом товаре (аналог газеты «Из рук в руки»). Пример: **http://www.avito.ru** (рис. 3.43).

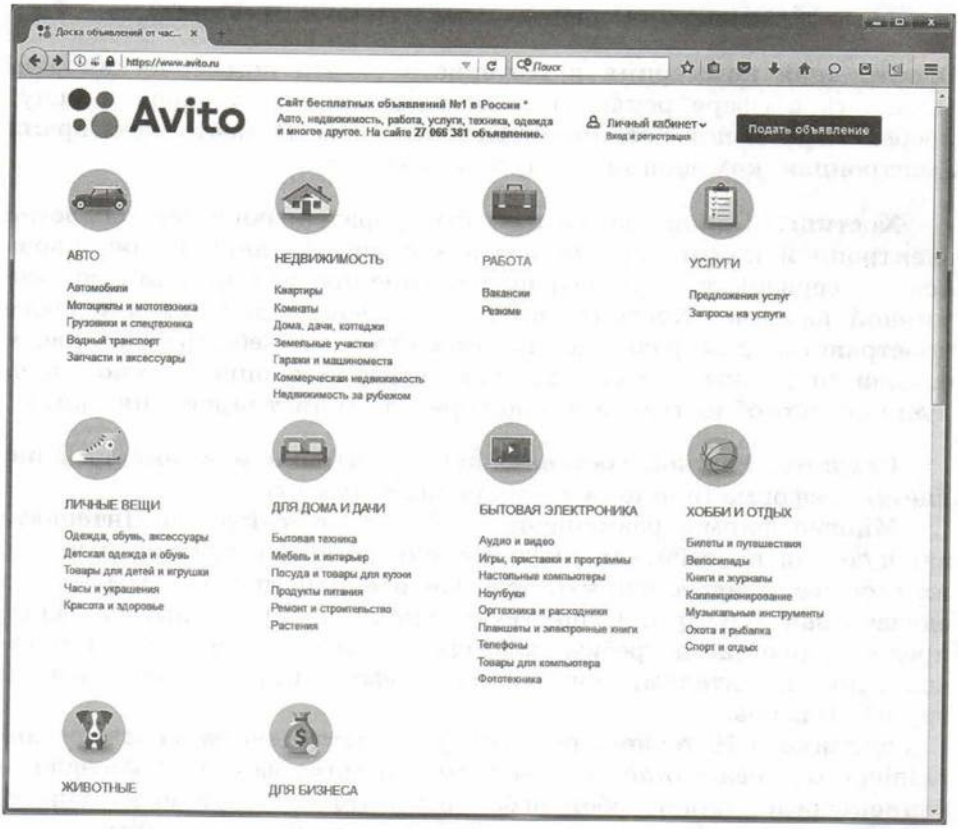


Рис. 3.43

Интернет-аукционы. Интересной формой электронной торговли являются интернет-аукционы. На такие аукционы выставляются самые разные товары: произведения искусства, компью-

терная техника, автомобили и т. д. Пример — интернет-аукцион **http://meshok.ru** (рис. 3.44).

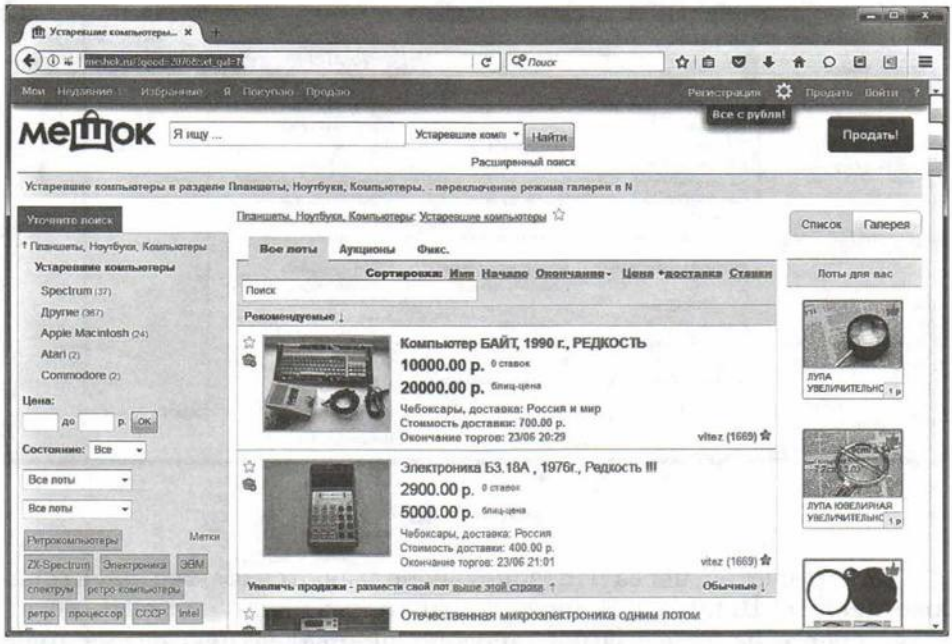


Рис. 3.44

Интернет-магазины. Самой удобной для покупателя формой электронной торговли являются интернет-магазины. В российском Интернете существует множество магазинов, в которых можно купить практически всё: компьютеры и программы, книги и CD, продукты питания и др. Для заказа товара его нужно выбрать в каталоге и затем поместить в «корзину» покупателя, щёлкнув на соответствующей пиктограмме. Далее следует заполнить поля в форме заказа и выбрать способ доставки товара. Пример — интернет-магазин **http://www.ozon.ru** (рис. 3.45).

Покупатель в интернет-магазине имеет возможность ознакомиться с товаром (техническими характеристиками, внешним видом товара и т. д.), а также с его ценой. Выбрав товар, потребитель может сделать непосредственно из Интернета заказ на его покупку, в котором указывается форма оплаты, время и место доставки и т. д. Оплата производится либо наличными деньгами после доставки товара, либо по кредитным карточкам.

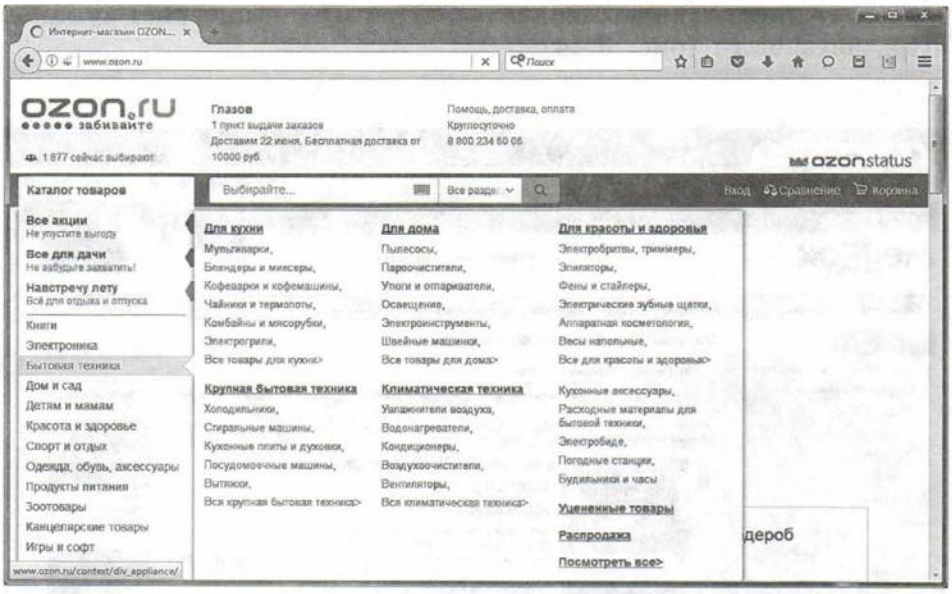


Рис. 3.45

Для расчётов через Интернет используются также **электронные деньги**. Покупатель перечисляет определённую сумму обычных денег в банк, а взамен получает эквивалентную сумму цифровых денег, которые существуют только в электронном виде и хранятся в «кошельке» (с использованием специальной программы) на компьютере покупателя. При расчётах через Интернет цифровые деньги поступают к продавцу, который переводит их в банк, а взамен получает обычные деньги.

Вопросы и задания

Перечислите и опишите формы электронной коммерции в Интернете.

3.13. Основы языка разметки гипертекста

Структура HTML-кода веб-страницы. Веб-страницы создаются с использованием языка разметки гипертекстовых документов **HTML** (*Hyper Text Markup Language*). В обычный текстовый документ вставляются управляющие символы — **HTML-теги**, которые определяют вид веб-страницы при её просмотре в браузере.

Теги заключаются в угловые скобки и могут быть одиночными или парными. Парный тег содержит открывающий и

закрывающий теги: такая пара тегов называется **контейнером**. Закрывающий тег содержит прямой слэш (/) перед обозначением тега. Теги могут записываться как прописными, так и строчными буквами. Некоторые теги имеют атрибуты, которые являются именами свойств и могут принимать определённые значения.

Веб-страница помещается в контейнер <HTML></HTML> и состоит из двух частей: заголовка и отображаемого в браузере содержания.

Заголовок страницы помещается в контейнер <HEAD></HEAD>.

Заголовок содержит название страницы, которое помещается в контейнер <TITLE></TITLE> и при просмотре отображается в верхней строке окна браузера. В раздел заголовка веб-страницы могут быть добавлены информационные одиночные теги <META>, имеющие атрибуты. Так, атрибут charset информирует браузер о кодировке веб-страницы.

Отображаемое в браузере содержание страницы помещается в контейнер <BODY></BODY>.

Пример:

```
<HTML>
<HEAD>
<TITLE>Заголовок страницы</TITLE>
<META ="text/html; charset=windows-1251"
http-equiv="content-type">
</HEAD>
<BODY>

Текст на веб-странице

</BODY>
</HTML>
```

Веб-страницу необходимо сохранить в виде файла. Начальную страницу сайта сохраняют обычно под именем index.htm. В качестве расширения имени файла веб-страницы можно также использовать html. Рекомендуется создать для размещения сайта специальную папку и сохранять все файлы разрабатываемого сайта в этой папке.

Шрифт. Размер шрифта заголовка задаётся парами тегов от <H1></H1> (самый крупный) до <H6></H6> (самый мелкий).

С помощью тега FONT и его атрибутов можно задать параметры форматирования шрифта. Атрибут FACE позволяет задать гарнитуру шрифта (например, FACE="Arial"), а атрибут SIZE — размер шрифта (например, SIZE=4). Атрибут COLOR позволяет

задавать цвет шрифта (например, COLOR="blue"). Значение атрибута COLOR можно задать либо названием цвета (например, "red", "green", "blue" и т. д.), либо его шестнадцатеричным значением.

Шестнадцатеричное представление цвета в HTML использует RGB-формат "#RRGGBB", где две первые шестнадцатеричные цифры задают интенсивность красного (red), две следующие — интенсивность зелёного (green) и две последние — интенсивность синего (blue) цветов. Минимальная интенсивность цвета задаётся шестнадцатеричным числом 00, а максимальная — FF. Например, синий цвет задаётся значением "#0000FF".

Форматирование текста. Разделение текста на абзацы производится с помощью контейнера <P></P>. При просмотре в браузере абзацы отделяются друг от друга интервалами.

Для каждого абзаца можно задать определённый тип выравнивания с помощью атрибута ALIGN. Выравнивание по левой границе: ALIGN="left", выравнивание по правой границе: ALIGN="right", выравнивание по центру: ALIGN="center", выравнивание по ширине: ALIGN="justify".

Заголовки целесообразно отделять от остального содержания страницы горизонтальной линией с помощью одиночного тега <HR>, а для перевода строки в пределах одного абзаца используется одиночный тег
.

Вставка изображений. На веб-страницы можно помещать изображения, хранящиеся в графических файлах форматов GIF, JPEG и PNG.

Для вставки изображения используется тег <IMG> с атрибутом SRC, который указывает место хранения файла на локальном компьютере или в Интернете. Если графический файл находится на локальном компьютере в той же папке, что и файл веб-страницы, то в качестве значения атрибута SRC достаточно указать только имя файла. Например:

```
<IMG SRC="file.gif">
```

Если файл находится в другой папке на данном локальном компьютере, то значением атрибута должно быть полное имя файла, включая путь к нему в иерархической файловой системе. Например:

```
<IMG SRC="D:\path\file.gif">
```

Если файл находится на удалённом сервере в Интернете, то должно быть указано доменное имя этого файла. Например:

```
<IMG SRC="http://www.server.ru/file.gif">
```

Расположить рисунок относительно текста различными способами позволяет атрибут ALIGN, который может принимать пять различных значений: top (верх), middle (середина), bottom (низ), left (слева) и right (справа). Например, чтобы рисунок располагался по правому краю текста, тег вставки изображения должен принять следующий вид:

```
<IMG SRC="file.gif" ALIGN="right">
```

Пользователи иногда в целях экономии времени отключают в браузере загрузку графических изображений и читают только тексты. Чтобы не терялся смысл страницы, вместо рисунка должен выводиться альтернативный текст.

Альтернативный текст выводится с помощью атрибута ALT, значением которого является текст, поясняющий, что должен был бы увидеть пользователь на рисунке. Пример:

```
<IMG SRC="file.gif" ALIGN="right" ALT="Текст">
```

Гиперссылки. Гиперссылки, размещённые на веб-странице, позволяют загружать в браузер другие веб-страницы, хранящиеся на локальном компьютере или в Интернете. Гиперссылка состоит из двух частей: адреса и указателя ссылки.

Гиперссылка создаётся с помощью универсального тега A и его атрибута HREF, указывающего, в каком файле хранится загружаемая веб-страница. Пример:

```
<A HREF="Адрес">Указатель ссылки</A>
```

Если загружаемая в браузер веб-страница размещена на локальном компьютере в той же папке, то вместо адреса указывается просто имя файла, например:

```
<A HREF="file.htm">Указатель ссылки</A>
```

Если загружаемая в браузер веб-страница размещена в Интернете, то в качестве адреса указывается её доменное имя, например:

```
<A HREF="http://www.server.ru/веб-сайт/file.htm">
Указатель ссылки</A>
```

Чтобы посетители сайта могли не только просматривать информацию, но и отправлять сведения для обработки и использования на сервер, на страницах сайта размещают интерактивные формы. Формы включают в себя элементы управления различных типов: текстовые поля, раскрывающиеся списки, флажки, переключатели и т. д.

Пусть мы хотим разместить на странице «Анкета» анкету для посетителей, чтобы выяснить, кто из них, с какими целями

и с помощью каких программ получает и использует информацию из сети Интернет, а также узнать, какую информацию они хотели бы видеть на нашем сайте.

Вся форма заключается в контейнер <FORM></FORM>.

Текстовые поля. В первую очередь выясним имя посетителя нашего сайта и его электронный адрес, чтобы иметь возможность ответить ему на замечания и поблагодарить за посещение сайта. Для получения этих данных разместим в форме два однострочных текстовых поля для ввода информации.

Текстовые поля создаются с помощью тега <INPUT> со значением атрибута TYPE="text". Атрибут NAME является обязательным и служит для идентификации полученной информации. Значением атрибута SIZE является число, задающее длину поля ввода в символах.

Чтобы анкета «читалась», необходимо разделить строки с помощью тега перевода строки
.

Переключатели. Далее, пусть мы хотим выяснить, к какой группе пользователей относит себя посетитель. Предложим выбрать ему один из нескольких вариантов: «школьник», «студент», «учитель».

Для этого необходимо создать **группу переключателей** (радиокнопок). Такая группа создаётся с помощью тега <INPUT> со значением атрибута TYPE="radio". Все элементы в группе должны иметь одинаковые значения атрибута NAME. Например, NAME="group". Ещё одним обязательным атрибутом является VALUE, которому присвоим значения "школьник", "студент" и "учитель". Значение атрибута VALUE должно быть уникальным для каждой радиокнопки, так как при её выборе именно оно передаётся серверу.

Флажки. Далее, пусть мы хотим узнать, какими сервисами Интернета наш посетитель пользуется наиболее часто. Здесь из предложенного перечня он может выбрать одновременно несколько вариантов, пометив их флажками.

Флажки создаются в теге <INPUT> со значением атрибута TYPE="checkbox". Флажки, объединённые в группу, могут иметь различные значения атрибута NAME. Например, NAME="box1", NAME="box2" и т. д. Ещё одним обязательным атрибутом является VALUE, которому присвоим значения "WWW", "e-mail" и "FTP". Значение атрибута VALUE должно быть уникальным для каждого флажка, так как при его выборе именно оно передаётся серверу.

Поля списков. Теперь выясним, какой из браузеров предпочитает посетитель сайта. Перечень браузеров представим в виде

раскрывающегося списка, из которого можно выбрать только один вариант.

Для реализации раскрывающегося списка используется контейнер <SELECT></SELECT>, в котором каждый элемент списка определяется тегом <OPTION>.

В переключателях, флажках и списках выбранный по умолчанию элемент задаётся с помощью атрибута SELECTED.

Текстовая область. В заключение поинтересуемся, что хотел бы видеть посетитель на страницах нашего сайта, какую информацию следовало бы в них добавить. Так как мы не можем знать заранее, насколько обширным будет ответ читателя, отведём для него текстовую область с линейкой прокрутки. В такое поле можно ввести достаточно длинный текст.

Текстовая область создаётся с помощью тега <TEXTAREA> с обязательными атрибутами: NAME — задаёт имя области, ROWS — определяет число строк и COLS — определяет число символов в строке.

Отправка данных из формы. Отправка введённой в форму информации осуществляется с помощью щелчка мышью по кнопке. Такая кнопка создаётся с помощью тега <INPUT>. Атрибуту TYPE необходимо присвоить значение "submit", а атрибуту VALUE, который задаёт надпись на кнопке, присвоить значение "Отправить".

Щелчком по кнопке *Отправить* можно отправить данные из формы на определённый адрес электронной почты. Для этого атрибуту ACTION контейнера <FORM> надо присвоить значение адреса электронной почты. Кроме того, в атрибутах METOD и ENCTYPE необходимо указать метод и форму передачи данных:

```
<FORM ACTION="mailto:Ugrinovich@LBZ.ru" METHOD="POST"
ENCTYPE="text/plain">
```

Создание веб-страниц в веб-редакторах. HTML-код веб-страниц можно создавать не вручную, а с использованием специальных программ — веб-редакторов. В них код элементов управления интерактивной формы создаётся автоматически.

Вопросы и задания

1. Какие теги (контейнеры) должны присутствовать в HTML-документе обязательно? Какова логическая структура веб-страницы?
2. Какие теги (контейнеры) используются для ввода заголовков? Форматирования шрифта? Ввода абзацев?
3. Какой тег и его атрибуты используются для вставки изображений в веб-страницы?

Практическая работа 3.8

Разработка сайта с использованием веб-редактора

Задание. Создать веб-страницу «Рост Интернета» с помощью веб-редактора.

Варианты выполнения работы:

- использование различных веб-редакторов;
- использование различных тематик веб-страниц.

Создание веб-страницы «Рост Интернета» с помощью визуального html-редактора KompoZer

Электронное приложение к главе 3.

1. В операционной системе Windows или Linux запустить визуальный html-редактор KompoZer.
   Выбрать вкладку *Код* с html-кодом заготовки веб-страницы (рис. 3.46).

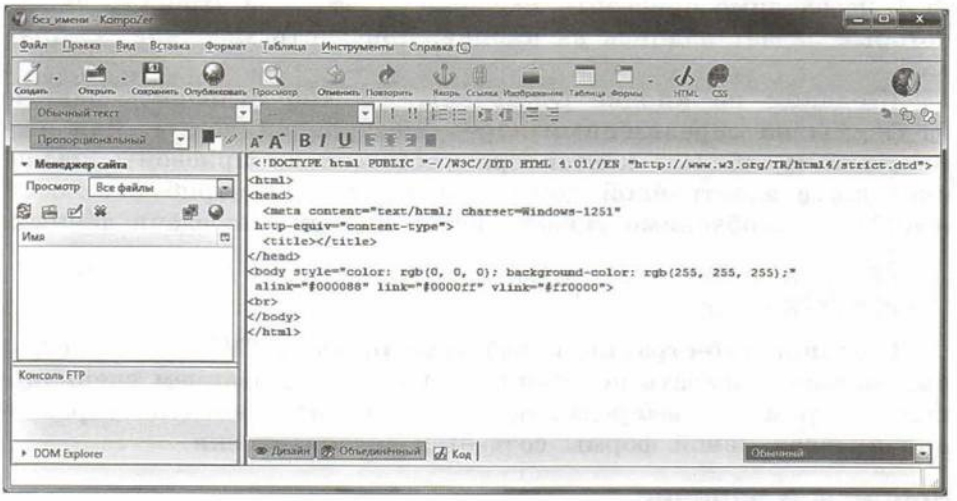


Рис. 3.46

2. Перейти на вкладку *Дизайн*, ввести заголовок «Рост Интернета» и отформатировать его.
   Ввести разделительную линию командой [*Вставка—Разделитель*].
   Результат показан на рис. 3.47.

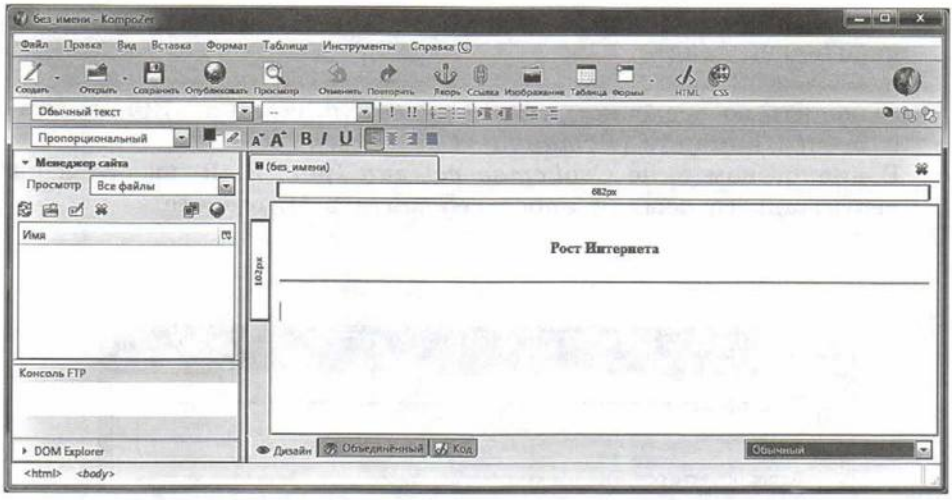


Рис. 3.47

3. Ввести текст: «На сайте Википедии — свободной энциклопедии можно познакомиться с историей и ключевыми принципами Интернета, рассматриваются области его использования».
4. Вставить изображение диаграммы роста Интернета командой [*Вставка—Изображение...*].
   В появившемся диалоговом окне *Свойства изображения* на вкладке *Адрес* (рис. 3.48) выбрать графический файл Internet.jpg и ввести в поле *Альтернативный текст* слова «Рост Интернета».

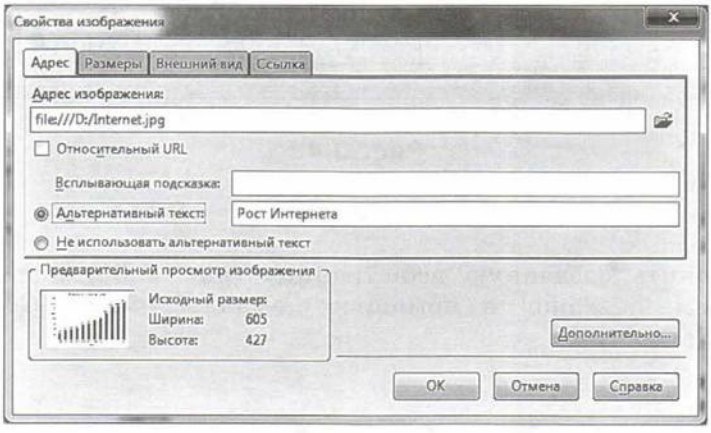


Рис. 3.48

5. На вкладке *Внешний вид* выбрать расположение рисунка относительно текста.

6. Выделить во введённом тексте указатель ссылки «Википедии» и ввести команду [*Вставка—Ссылка...*].
   В диалоговом окне *Свойства ссылки* (рис. 3.49) ввести в качестве адреса ссылки адрес веб-сайта в Интернете.

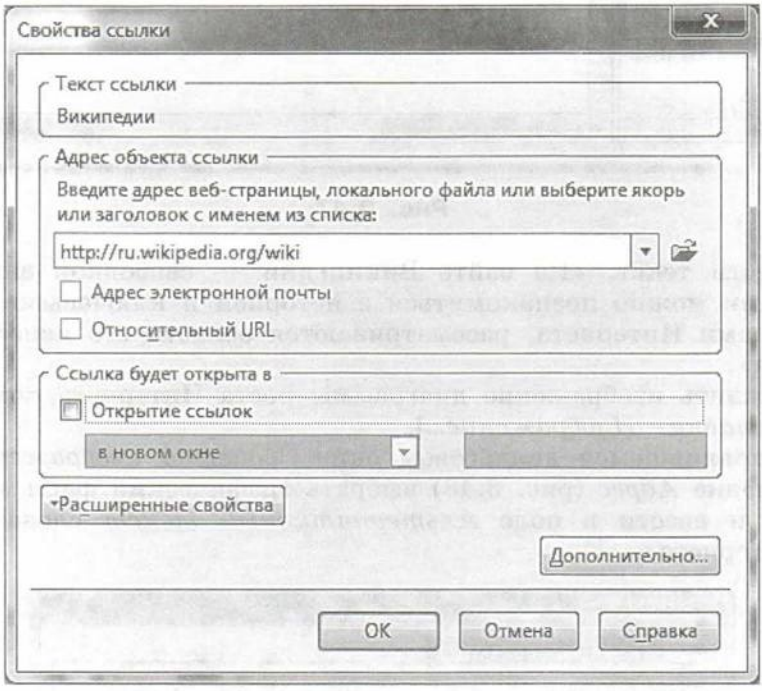


Рис. 3.49

7. Сохранить созданную веб-страницу (рис. 3.50) в файле под именем index.html с помощью команды [*Файл—Сохранить как...*].

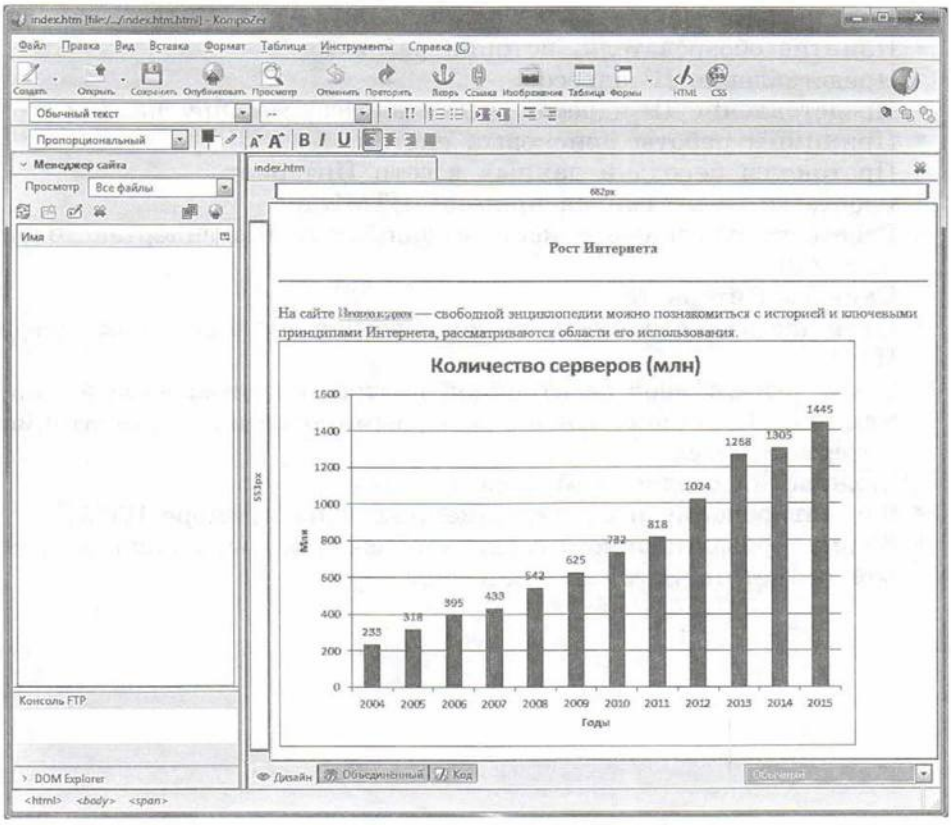


Рис. 3.50

ЭОР к главе 3 на сайте ФЦИОР (http://fcior.edu.ru)

- Архитектура Интернет
- Вставка графических объектов с использованием языка HTML
- Глобальные компьютерные сети
- История создания и развития сети Интернет
- Организация и протоколы, используемые в сети Интернет
- Основные определения и понятия языка HTML. Структура и логика языка разметки HTML
- Основные определения и понятия языка HTML. Структура и логика языка разметки HTML. Понятие тега
- Основные теги HTML
- Поиск информации в Интернете
- Поисковые системы в сети Интернет и принципы их работы

- Пользование основными обозревателями
- Понятие обозревателя, история развития
- Представление IP адресов
- Представление IP адресов, части адреса, маршрутизация
- Принципы работы поисковых систем
- Протоколы передачи данных в сети Интернет
- Работа со ссылками на примере HTML
- Работа со ссылками с использованием языка гипертекстовой разметки
- Службы Интернета
- Создание веб-страницы с использованием основных тегов HTML
- Технологии обмена электронной почтой, представление информации в Интернет, языки программирования, эксплуатация интернет-систем
- Технология создания Web-сайта
- Форматирование и оформление текста на примере HTML
- Форматирование текста с использованием языка гипертекстовой разметки. Заголовки. Абзацы

Глава 4
АЛГОРИТМИЗАЦИЯ И ОСНОВЫ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ

В процессе изучения данной главы рекомендуется установить следующее программное обеспечение:

для операционной системы Windows:

- систему объектно-ориентированного программирования Microsoft Visual Studio;

для операционных систем Windows и Linux:

- систему объектно-ориентированного программирования Lazarus.

4.1. Алгоритм и кодирование основных алгоритмических структур

4.1.1. Алгоритм и его свойства

Алгоритмы могут описывать процессы преобразования самых разных объектов. Широкое распространение получили вычислительные алгоритмы, которые описывают преобразование числовых данных. Само слово «алгоритм» происходит от *algorithmi* — латинской формы написания имени выдающегося математика IX века аль-Хорезми, который сформулировал правила выполнения арифметических операций.

Алгоритм обладает следующими свойствами.

Результативность. Выполнение алгоритма должно завершиться за конечное число шагов.

Дискретность. Алгоритм должен обеспечивать преобразование объекта из начального состояния в конечное состояние за определённое число дискретных шагов.

Массовость. Один и тот же алгоритм может применяться к большому количеству однотипных объектов.

Детерминированность (определённость). Исполнитель должен выполнять команды алгоритма в строго определённой последовательности. Каждое действие алгоритма должно быть строго определено, не допускать разночтений. Алгоритм должен выдавать один и тот же результат для разных исходных данных.

Понятность. Алгоритм должен содержать только команды, входящие в систему команд исполнителя и записанные на понятном для исполнителя языке.

Алгоритм — это строго детерминированная последовательность действий, описывающая процесс преобразования объекта из начального состояния в конечное (исходных данных в результат), записанная с помощью понятных исполнителю команд.

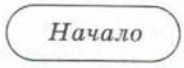
Существуют разные способы записи алгоритмов: словесный (на естественных языках), графический (блок-схемы), описание на языках программирования.

Блок-схемы алгоритмов. Блок-схема представляет собой графическую форму записи алгоритмов. Она позволяет сделать алгоритм более наглядным и выделить в нём основные алгоритмические структуры (линейная, ветвление и цикл). Если исполнителем алгоритма является человек, то он может по блок-схеме легко проследить выполнение алгоритма, так как элементы блок-схем соединены стрелками, указывающими шаги выполнения алгоритма.

Элементы алгоритма изображаются на блок-схеме с помощью различных геометрических фигур, внутри которых записывается программный код (табл. 4.1).

Таблица 4.1

Элементы блок-схем

Элемент блок-схемы	Назначение элемента блок-схемы
	Прямоугольник с закруглёнными углами применяется для обозначения начала или конца алгоритма
	Параллелограмм предназначен для описания ввода или вывода данных, имеет один вход вверху и один выход внизу
	Прямоугольник применяется для описания линейной последовательности команд, имеет один вход вверху и один выход внизу
	Ромб служит для обозначения условий в алгоритмических структурах «ветвление» и «выбор», имеет один вход верху и два выхода (налево, если условие истинно, и направо, если условие ложно). Применяется также для составления алгоритмической структуры «цикл»

Элемент блок-схемы	Назначение элемента блок-схемы
Счетчик	Шестиугольник используется в алгоритмической конструкции «цикл со счётчиком»
Подпрограмма	Прямоугольник в прямоугольнике применяется для вызова отдельно описанного алгоритма (подпрограммы)
Объявление переменных	Прямоугольник со срезанным углом применяется для объявления переменных или ввода комментариев

Вопросы и задания

Какие из нижеперечисленных правил являются алгоритмами:
- орфографические правила;
- правила выполнения арифметических операций;
- правила техники безопасности;
- правила перевода чисел из одной системы счисления в другую?

Ответ обоснуйте.

4.1.2. Алгоритмические структуры «ветвление» и «цикл»

Алгоритмическая структура ветвление. В отличие от линейных алгоритмов, в которых команды выполняются последовательно одна за другой, в алгоритмическую структуру ветвление входит условие: в случае истинности этого условия реализуется одна последовательность команд, а в случае ложности — другая.

В **алгоритмической структуре ветвление** одна или другая серия команд выполняется в зависимости от истинности **условия**.

Алгоритмическая структура ветвление может быть зафиксирована графически с помощью блок-схемы (рис. 4.1). В блок-схеме на рис. 4.1 альтернативные последовательности команд обозначены словами *Серия 1* и *Серия 2*.

На языках объектно-ориентированного программирования, которые мы будем рассматривать в этой главе, алгоритмическая структура ветвление

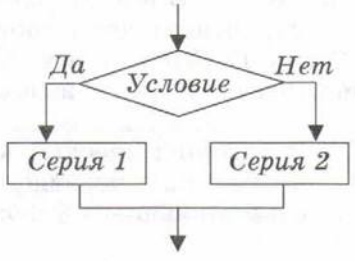


Рис. 4.1

кодируется с использованием оператора **If**. После первого ключевого слова **If** должно быть записано условие. После ключевого слова **Then** (в языке Visual C# оно отсутствует) идёт последовательность команд (*Серия 1*), которая должна выполняться, если условие принимает значение «истина». После ключевого слова **Else** размещается последовательность команд (*Серия 2*), которая должна выполняться, если условие принимает значение «ложь».

В сокращённой форме оператора ключевое слово **Else** отсутствует. Тогда, если условие ложно, выполнение оператора условного перехода заканчивается и выполняется следующая строка программы.

Для реализации ветвления со многими вариантами серий команд используется алгоритмическая конструкция выбор. Конструкция «выбор» может быть зафиксирована графически с помощью блок-схемы (рис. 4.2).

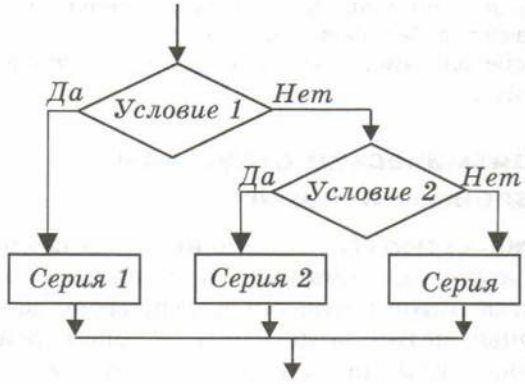


Рис. 4.2

В структуру выбора входят несколько **условий**, проверка которых осуществляется по порядку их записи в структуре выбора. При истинности одного из условий (*Условие 1*, *Условие 2* и т. д.) выполняется соответствующая последовательность команд (*Серия 1*, *Серия 2* и т. д.). Если ни одно из условий не является истиным, то будет выполнена последовательность команд *Серия*.

В алгоритмической конструкции **выбор** выполняется одна из нескольких последовательностей команд при истинности соответствующего условия.

На языках объектно-ориентированного программирования конструкция выбор кодируется с использованием оператора

выбора. На языке программирования Visual Basic .NET оператор выбора начинается с ключевых слов **Select Case**, на языке Visual C# — с ключевого слова **switch**, а на языке Lazarus — с ключевого слова **Case**.

После ключевого слова записывается выражение (переменная или арифметическое выражение). Заданное выражение сравнивается с определёнными значениями. При истинности одного из условий начинает выполняться соответствующая серия команд. Если ни одно из условий не истинно, то выполняется серия команд после ключевого слова **Else** (в языках Visual Basic .NET и Lazarus) или ключевого слова **default** (в языке Visual C#).

В сокращённой форме оператора ключевое слово **Else** (**default**) отсутствует. Тогда, если все условия ложны, выполнение оператора выбора заканчивается и выполняется следующая строка программы.

Алгоритмическая структура цикл. В алгоритмическую структуру цикл входит серия команд, выполняемая многократно. Такая последовательность команд называется **телом цикла**.

В **алгоритмической структуре цикл** серия команд (**тело цикла**) выполняется многократно.

Циклические алгоритмические структуры бывают двух типов:

- циклы со счётчиком, в которых тело цикла выполняется определённое количество раз;
- циклы по условию, в которых тело цикла выполняется, пока истинно (или ложно) заданное условие.

Алгоритмическая структура цикл может быть описана графически с помощью блок-схемы (рис. 4.3).

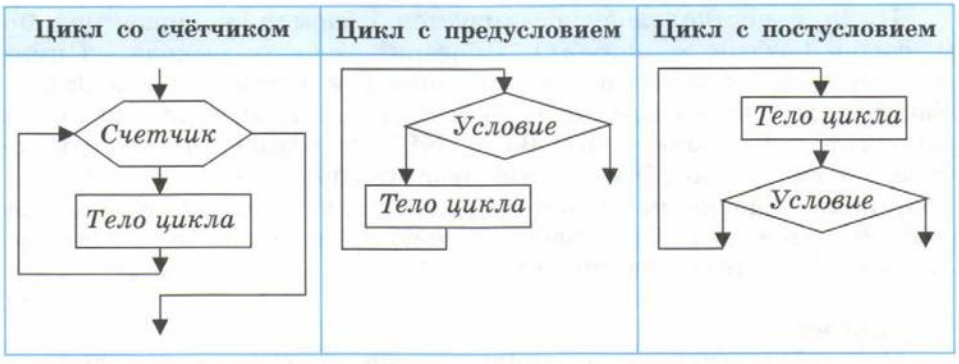


Рис. 4.3

Цикл со счётчиком используется, когда заранее известно, какое количество повторений тела цикла необходимо выполнить. Количество повторений задаётся с помощью счётчика.

Цикл со счётчиком реализуется при помощи оператора **For**. В заголовке цикла устанавливается начальное значение переменной **Счётчик**, определяется величина её конечного значения и величина изменения значения за один шаг. Затем располагаются многократно выполняемые операторы тела цикла.

Цикл с условием используется, когда заранее неизвестно, какое количество раз должно повториться тело цикла. В таких случаях количество повторений зависит от некоторого условия.

Цикл называется **циклом с предусловием**, если условие выхода из цикла стоит в начале, перед телом цикла. В случае ложности условия цикл с предусловием не выполнится ни разу.

В языках объектно-ориентированного программирования цикл с предусловием реализуется с помощью оператора **While** (в языке Visual Basic .NET — **Do While**). Проверка условия выхода из цикла проводится до начала цикла с помощью ключевого слова **While**, которое обеспечивает выполнение цикла, пока условие истинно. Как только условие примет значение «ложь», выполнение цикла закончится.

Цикл называется **циклом с постусловием**, если условие выхода из цикла стоит в конце, после тела цикла. Цикл с постусловием выполняется обязательно, как минимум один раз, независимо от того, истинно условие или нет.

Цикл с постусловием реализуется с помощью оператора **Do** (в языке Lazarus — **Repeat**). Проверка условия выхода из цикла производится после цикла с помощью ключевого слова **Until**. Как только условие примет значение «истина», выполнение цикла закончится. В языке Visual Basic .NET используется также ключевое слово **While**. Оно обеспечивает выполнение цикла до тех пор, пока условие не станет ложным, т. е. пока условие имеет значение «истина». Как только условие примет значение «ложь», выполнение цикла закончится.

Пример

Данный пример демонстрирует использование алгоритмических структур.

Рассмотрим *алгоритм перевода целых десятичных чисел в двоичную систему счисления на естественном языке*:

1) Ввести десятичное целое число.
2) В цикле с предусловием, пока исходное целое десятичное число или целое частное больше 0, выполнить вычисления:
   2.1) Вычислить остаток от деления исходного целого десятичного числа или целого частного на основание новой системы (на 2).
   2.2) Выполнить целочисленное деление целого десятичного числа или целого частного на основание новой системы (на 2).
   2.3) Записать полученный остаток от деления слева от двоичного числа (остатки, записанные в обратном порядке, образуют двоичное число).
3) Вывести двоичное целое число.

На рисунке 4.4 изображена блок-схема этого алгоритма. Команды в блоках записаны на языке Visual Basic .NET.

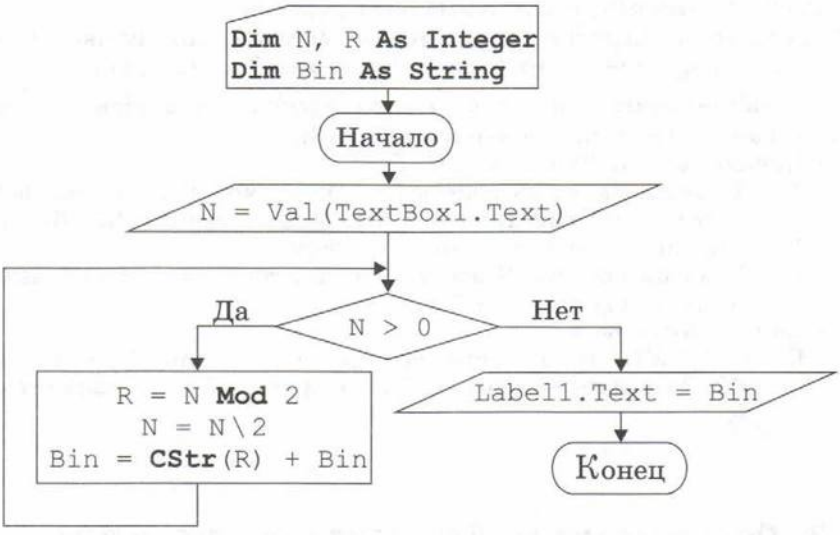


Рис. 4.4

Вопросы и задания

1. Какие типы алгоритмических конструкций использованы в приведённом в параграфе алгоритме перевода десятичных чисел в двоичное представление?

2. Опишите алгоритм перевода чисел из двоичной системы счисления в десятичную. Оформите ответ в форме блок-схемы для числа 1011.
3. Как работает автомат для покупки газет и журналов? Объясните его работу с использованием алгоритмической конструкции проверки условия при выборе издания и выдаче сдачи автоматом. Оформите ответ с помощью блок-схемы.
4. Приведите примеры использования алгоритмической конструкции проверки условия в различных приборах. Какой алгоритм управляет датчиком включения и отключения света в подъезде дома («умный свет») или автоматическими дверями в магазине? Оформите свой пример с помощью блок-схемы.
5. В различных социальных сетях и журналах часто используются опросы или тесты. Опишите с помощью блок-схемы алгоритм автоматического тестирования при условии, что в тесте предлагается пять вопросов и на каждый вопрос — три возможных ответа. Тестируемый может выбрать только один из предложенных вариантов ответа. Каждый ответ имеет свой балл, который учитывается в суммарном балле тестируемого. По итогам прохождения теста набранный балл сравнивается со шкалой результатов, и для тестируемого на экран выводится сообщение в соответствии с тем диапазоном баллов, в который попал суммарный результат.

   Используйте описание алгоритма на естественном языке, предложенное ниже, для построения блок-схемы «Тестирование».

   i — счётчик цикла обработки пяти вопросов (i меняется от 1 до 5).
   n — номер ответа (n меняется от 1 до 3).
   1) **Начало цикла:** Для i от 1 до 5:
      1.1) Вывести на экран вопрос i и три возможных ответа, перенумерованных как 1, 2, 3 и баллы для ответов ($B1$, $B2$, $B3$).
      1.2) Ввести с клавиатуры номер ответа n.
      1.3) В ячейку суммы S добавить балл, соответствующий выбранному ответу ($S = S + Bn$).
   2) **Конец цикла по i**
   3) Если $S < X1$, то вывести сообщение *1*, если $S >= X1$ *AND* $S < X2$, то вывести сообщение *2*, если $S >= X2$, то вывести сообщение *3*.

4.1.3. Подпрограммы. Рекурсивные алгоритмы

Подпрограммы. Во многих алгоритмах те или иные действия могут повторяться для различных исходных данных. Например, пусть нужно составить алгоритм для вычисления площади круговой пластины с круглыми отверстиями (рис. 4.5). Очевидно, для этого нужно вычислить площадь круга, соответствующего внешнему контуру пластины, и вычесть из нее сумму площадей круговых отверстий.

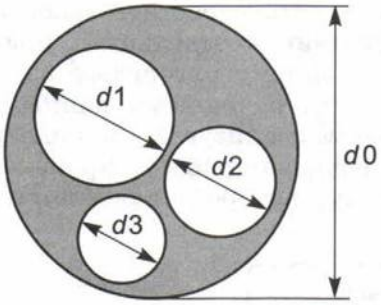


Рис. 4.5

В алгоритме для вычисления площади пластины потребуется записать четыре одинаковых вычислительных блока, в которых рассчитывается площадь круга диаметром, соответственно, $d0$, $d1$, $d2$, $d3$ (рис. 4.6).

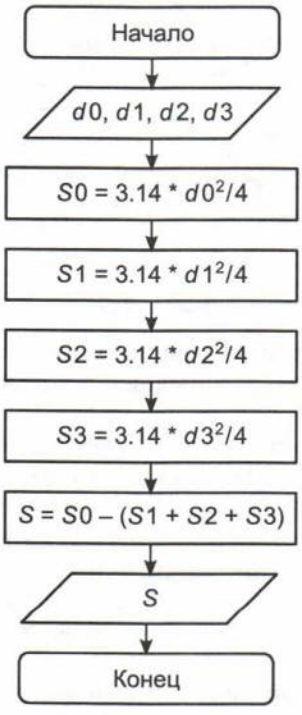


Рис. 4.6

Существует возможность упростить такой алгоритм, вынеся из него однотипные действия в отдельный, подчинённый алгоритм и обращаясь к подчинённому алгоритму из основного (главного) алгоритма с требуемыми исходными данными (рис. 4.7). Такой подчинённый алгоритм реализуется в виде **подпрограммы**. Когда подпрограмма завершает работу, происходит *возврат* из неё в основную программу, которой передаются результаты работы подпрограммы.

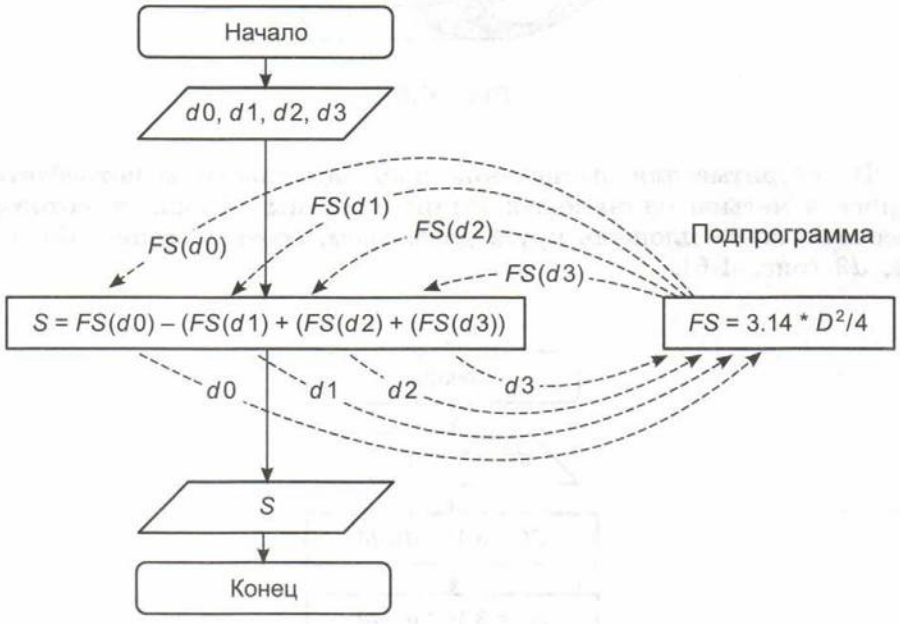


Рис. 4.7

Процедуры и функции. Возможны два вида подпрограмм, различающиеся принципами их вызова и количеством возвращаемых ими значений.

Функция — это подпрограмма, которая, принимая одно или несколько исходных данных, возвращает единственное значение, являющееся результатом работы этой функции. Такая подпрограмма аналогична математическим функциям (например, тригонометрическим, которые по заданному значению угла вычисляют его синус, косинус, тангенс и т. д.). Обращение к функции (вызов функции) обычно записывается так же, как это делается в математике, например:

`X = SIN(A)` — вызов функции, вычисляющей значение синуса угла, величина которого задана в переменной *A*; результат, возвращенный этой функцией, записывается в переменную *X*;

`Y = SQRT(1 - COS(A)*COS(A))` — реализация вычисления значения синуса угла по формуле $\sqrt{1-\cos^2(A)}$: здесь результаты, возвращаемые функциями вычисления косинуса угла A, сразу же используются в арифметическом выражении, значение которого, в свою очередь, затем используется в качестве исходных данных для функции вычисления квадратного корня (SQRT).

Обычно при написании алгоритма подпрограммы-функции после выполнения всех вычислений полученные результаты нужно записать в специальную переменную. Именно её значение возвращается в основной алгоритм в качестве результата работы функции.

В современных языках программирования предусматривается большое количество различных готовых функций (**стандартных**, или **библиотечных**), которые достаточно вызывать из основного алгоритма. Однако при необходимости можно создавать свои собственные (**пользовательские**) функции, реализующие любой алгоритм.

Процедура — подпрограмма, которая может принимать любое количество исходных данных и возвращать любое количество значений в качестве результатов своей работы. Обращение к процедуре в основном алгоритме записывается отдельной командой: и исходные данные, и переменные, в которые должны быть записаны результаты работы процедуры, записываются в скобках списком через запятую (сначала перечисляются исходные данные, а затем — переменные-результаты). Вызовы процедур не могут использоваться в арифметических выражениях (в отличие от функций).

Формальные и фактические параметры. Для удобства и большей универсальности при написании подпрограмм переменные, используемые в алгоритме подпрограммы, обычно полностью независимы от переменных, используемых в основном алгоритме (даже если совпадают имена этих переменных). Принято считать, что подпрограмма использует отдельное *адресное пространство*. Соответственно, переменные, используемые внутри подпрограммы, называют **локальными**.

Чтобы обеспечить передачу данных из основного алгоритма в подпрограмму и возврат из подпрограммы результатов её работы, используется механизм формальных и фактических параметров.

- При написании подпрограммы в её заголовке записывается «шаблон» вызова этой подпрограммы. Он включает имя подпрограммы и записанный в скобках список локальных переменных, используемых внутри подпрограммы. Эти переменные представляют собой передаваемые в подпрограмму *исходные* (*входные*) данные, а для процедуры — также список локальных переменных, в которые в процедуре записываются резуль-

таты вычислений. Все эти переменные, записанные в заголовке подпрограммы, называют **формальными параметрами.**

- При вызове подпрограммы из основного алгоритма в строке обращения к подпрограмме записывается её имя, а в скобках — список передаваемых ей реальных исходных значений и (для процедуры) список переменных, в которые после выполнения процедуры будут записаны результаты её работы. В качестве исходных данных могут быть записаны константы, переменные, арифметические выражения или подпрограммы-функции. Все эти исходные данные и переменные для записи результатов называют **фактическими параметрами.**

Для правильной работы подпрограммы необходимо соблюдать правило: *количество, порядок записи и типы формальных и фактических параметров должны совпадать*[1].

Например:

	Имя подпрограммы	1-е исходное данное (целое число)	2-е исходное данное (вещественное число)	3-е исходное данное (текстовый символ)	1-й результат (число)	2-й результат (текст)
Заголовок подпрограммы:	PODPROG	(A,	X,	T,	Y,	S)
Правильный вызов подпрограммы:	PODPROG	(5,	2.3,	"@",	Z,	TXT)
Неправильный вызов подпрограммы:	PODPROG	("$",	3,	2.5,	"&",	7)
Суть ошибки:		Формальный параметр — целое число, а фактически задан символ	Формальный параметр — вещественное число, а фактически задано целое число	Формальный параметр — текстовый символ, а фактически задано вещественное число	В качестве фактического параметра должна быть записана переменная	В качестве фактического параметра должна быть записана переменная

1) Кроме особых случаев, когда часть фактических параметров может быть пропущена. Такая возможность реализуется особыми приёмами программирования. Вместо отсутствующих фактических параметров в подпрограмме тогда используются некие заранее оговоренные и занесённые в алгоритм подпрограммы значения «по умолчанию».

Процедуры — обработчики событий. Современные программы для компьютеров обычно пишутся по объектному принципу. Рассматривается некий набор **объектов** — как отображаемых визуально (изображаемые на экране окна, кнопки и другие элементы интерфейса), так и не имеющих визуального представления (например, аудиозапись). Каждый такой объект может иметь свой набор **свойств**, значения которых можно задать при создании объекта, а затем изменять при выполнении программы. Кроме того, рассматривается некоторый набор возможных **событий** — ситуаций, как возникающих при работе самого компьютера и его периферийных устройств (скажем, замятие бумаги в принтере), так и инициированных пользователем (например, в качестве события может выступать щелчок кнопкой мыши или нажатие определённой клавиши на клавиатуре).

Для каждого объекта и каждого возможного события, совершаемого над ним, разрабатывается отдельная процедура — алгоритм, определяющий действия компьютера при возникновении того или иного события. Такая процедура называется **обработчиком события**.

При отсутствии событий компьютер находится в «ждущем» режиме — не выполняет никаких действий, которые можно было бы наблюдать со стороны. Если же для какого-то объекта произойдёт то или иное событие, для которого была предусмотрена процедура-обработчик, то компьютер запустит в работу именно её. Если произойдёт несколько событий для одного и того же или для разных объектов (одновременно или поочерёдно, когда новое событие происходит до завершения обработки предыдущего), то компьютер может запустить несколько соответствующих обработчиков, — если только их выполнение не противоречит друг другу.

Рекурсивные алгоритмы. Возможна ситуация, когда в ходе выполнения подпрограммы её алгоритмом предусматривается вызов самой этой же подпрограммы. Такой приём называют **рекурсией**, а подобные алгоритмы с «самовызовом» называют рекурсивными алгоритмами.

Пример использования рекурсии — вычисление факториала.

Факториалом натурального числа называют значение, равное произведению этого числа на все числа натурального ряда, меньшие данного числа. Факториал обозначается восклицательным знаком, записанным после исходного числа. Например, факториал числа 4 вычисляется так: $4! = 4 \cdot 3 \cdot 2 \cdot 1$.

Создадим для решения этой задачи рекурсивный алгоритм в виде подпрограммы, которой в качестве исходного данного будет передаваться значение 4.

Предположим, что мы уже знаем, чему равен факториал предыдущего натурального числа. Тогда для вычисления 4! достаточно умножить это уже известное значение 3! на число 4.

Однако мы пока не знаем, чему равен 3!. Вычислить его мы можем... вызвав эту же самую подпрограмму, которую мы пишем для вычисления факториала числа 4. Но теперь в эту подпрограмму в качестве исходного данного передается число 3. И вычисление значения 3! производится по тому же принципу: число 3 умножается на 2!.

Поскольку величина 2! нам тоже пока неизвестна, её мы вычислим снова вызовом этой же подпрограммы, которой будет передано уже число 2. И здесь вычисление производится тоже умножением числа 2 на факториал предыдущего числа 1 (и тоже вызовом нашей же подпрограммы с передачей ей исходного данного — числа 1).

Но вот чему равен факториал единицы, мы знаем точно: он равен единице. И этот факт можно отразить в нашей подпрограмме при помощи оператора ветвления типа: «ЕСЛИ переданное число равно 1, то результат равен 1».

Запишем теперь (на псевдокоде; это условный язык — смесь языка программирования и естественного (русского, английского) языка) соответствующую подпрограмму:

```
FAKTOR(N) // N - формальный параметр, соответствующий
          // заданному числу
    ЕСЛИ N = 1 ТО РЕЗУЛЬТАТ = 1
    ИНАЧЕ РЕЗУЛЬТАТ = N * FAKTOR(N-1)
                                  // рекурсивный вызов
КОНЕЦ ПОДПРОГРАММЫ
```

А теперь посмотрим, как работает этот алгоритм, например, для числа 4 (из основного алгоритма производится вызов подпрограммы в виде `F = FAKTOR(4)`). Для этого будем отслеживать изменение значения переменной N (рис. 4.8).

Как видим, в процессе работы подпрограммы сначала формируется цепочка её *вложенных вызовов* — до тех пор, пока не сработает **условие завершения рекурсии** (в данном случае $N = 1$). После этого в каждом из выполненных вложенных вызовов — начиная с самого последнего и в обратном порядке — производится вычисление результата и возврат в предыдущий вложенный вызов подпрограммы. Такая обратная цепочка рекурсивных возвратов выполняется, пока не произойдёт возврат в самый первый вызов подпрограммы. А когда в ней будет вычислен окончательный результат, происходит возврат в основной алгоритм.

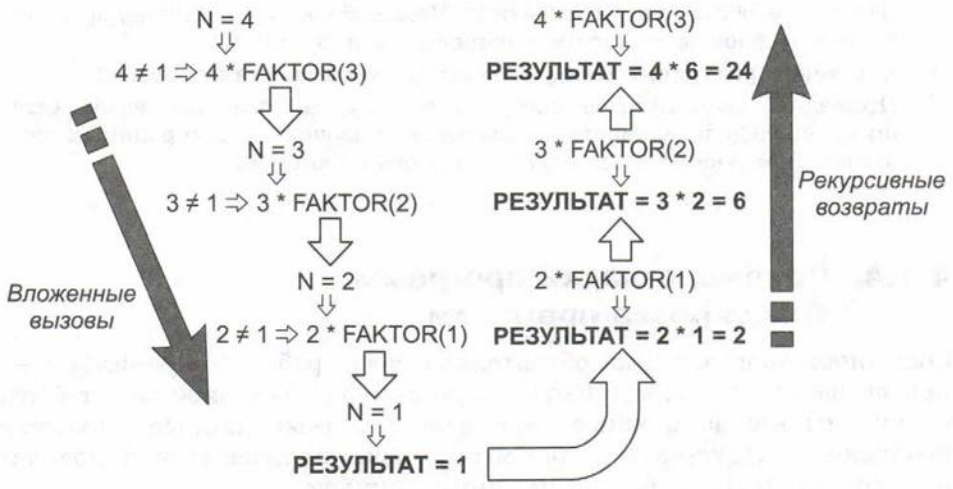


Рис. 4.8

Важно помнить, что использовать рекурсию нужно оптимально. Некоторые задачи более просто и изящно решаются при помощи рекурсии. Но во многих случаях гораздо быстрее и проще разработать обычный, нерекурсивный алгоритм. Например, для вычисления факториала можно использовать обычный цикл (ниже приведен фрагмент алгоритма):

```
F = 1
ПОКА N > 1 выполнять:
  F = F * N
  N = N - 1
КОНЕЦ ПОКА
```

Вопросы и задания

1. Что такое подпрограмма? Для чего используются подпрограммы?
2. Сформулируйте основные различия между подпрограммой-функцией и подпрограммой-процедурой. В каких случаях предпочтительно использовать тот или иной из этих двух видов подпрограмм? Приведите примеры.
3. Что понимается под стандартными функциями? Как вы считаете, почему их называют стандартными? Найдите в Интернете информацию о том, почему такие функции называют библиотечными.
4. Как работает механизм передачи данных между основным алгоритмом и подпрограммой при помощи формальных и фактических параметров? Какое основное правило при этом должно соблюдаться?

5. Что такое обработчики событий? Чем работа таких процедур отличается от работы обычных подпрограмм-процедур?
6. Что такое рекурсия? Как работают рекурсивные алгоритмы?
*7. Приведите свой пример построения рекурсивного алгоритма. Опишите последовательность вложенных вызовов и рекурсивных возвратов при работе вашего рекурсивного алгоритма.

4.1.4. Приёмы отладки программ. Трассировка программ

Созданная программа не обязательно сразу работает правильно — при её разработке могут быть допущены ошибки либо могут быть не учтены все возможные варианты исходных данных. Поэтому необходима проверка программы, поиск, выявление и исправление допущенных в ней возможных ошибок.

Поиск и исправление ошибок в программе называют её **отладкой.**

Первый шаг отладки — проверка синтаксиса (правильности написания текста программы по созданному алгоритму). Обычно эта проверка выполняется автоматически средой программирования. Обнаруженные возможные ошибки компьютер выделяет визуально (цветом, подчёркиванием и т. д.), чтобы разработчик алгоритма обратил на них своё внимание.

Важно помнить, что одна допущенная ошибка может стать причиной «обнаружения» других предполагаемых ошибок (например, если в начале алгоритма вы забыли объявить тип используемой переменной, то возникнут ошибки при её последующем использовании). Поэтому ошибки надо исправлять с начала алгоритма и каждый раз, исправив очередную ошибку, повторять проверку программы заново: возможно, какие-то из ранее обнаруженных ошибок исчезнут сами собой.

Второй шаг отладки — проверка правильности работы программы на каком-то одном типовом примере. Цель этой проверки — убедиться, что в программе не допущено семантических (смысловых) ошибок.

Такую проверку можно выполнить путём **трассировки программы** — её пробного выполнения вручную для конкретных исходных данных, с контролем получаемых значений переменных на каждом шаге. Эти значения переменных удобно записывать в виде таблицы, которую называют **таблицей трассировки.**

Приведём *пример трассировки программы* вычисления факториала натурального числа:

```
НАЧАЛО
ВВОД N
F = 1
ПОКА N > 1 выполнять:
  F = F * N
  N = N - 1
КОНЕЦ ПОКА
ВЫВОД F
КОНЕЦ
```

Проведём трассировку программы для значения N, равного 6. Моменты изменения значений переменных будем для наглядности выделять серым фоном. Заметим, что в остальных случаях в строках таблицы повторяются те значения переменных, которые были получены ранее и в данный момент сохраняются в них. Если же переменная ещё не имеет никакого значения, в соответствующей ячейке таблицы ставится прочерк.

	N	F
ВВОД N	6	–
F = 1	6	1
ПОКА N > 1	6 > 1 — цикл выполняется	1
F = F * N	6	1 · 6 = 6
N = N - 1	6 – 1 = 5	6
ПОКА N > 1	5 > 1 — цикл выполняется	6
F = F * N	5	6 · 5 = 30
N = N - 1	5 – 1 = 4	30
ПОКА N > 1	4 > 1 — цикл выполняется	30
F = F * N	4	30 · 4 = 120
N = N - 1	4 – 1 = 3	120
ПОКА N > 1	3 > 1 — цикл выполняется	120
F = F * N	3	120 · 3 = 360
N = N - 1	3 – 1 = 2	360
ПОКА N > 1	2 > 1 — цикл выполняется	360
F = F * N	2	360 · 2 = 720
N = N - 1	2 – 1 = 1	720
ПОКА N > 1	1 = 1 — цикл прекращается	720
ВЫВОД F	1	720

Нетрудно видеть, что для команды ветвления или для цикла с условием в таблице трассировки записывается соответствующее условие, и, в зависимости от его истинности, записывается соответствующий вывод, согласно которому далее выполняются те или иные команды алгоритма. Для цикла в таблице трассировки соответствующие команды из тела цикла повторяются столько раз, сколько реально выполняется этот цикл при заданных исходных данных (соответствующие блоки для наглядности разделены двойной линией границы).

В последней строке таблицы трассировки, соответствующей команде вывода результата, для соответствующей переменной (F) мы видим полученное значение (720).

Проверяем его правильность, зная математическую формулу расчета факториала:

$$6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720.$$

Совпадение результата, полученного при трассировке программы, со значением результата, вычисленным по исходной математической формуле, показывает, что алгоритм для выбранного исходного значения работает правильно. Однако это не гарантирует его правильную работу при всех возможных исходных данных. Поэтому необходим **третий шаг отладки программы — её тестирование**.

Для выполнения тестирования алгоритма необходимо тщательно продумать и подготовить **набор тестов** — такой набор возможных исходных данных, который исчерпывает все возможные ситуации.

Так, для программы, связанной с математическими расчётами, необходимо предусмотреть следующие варианты исходных данных:

- типичные для решаемой задачи значения исходных данных (например, при решении задачи на движение автомобилей это будут значения скоростей порядка десятков километров в час, а для задачи на движение космических кораблей — значения скоростей в несколько километров в секунду);
- граничные значения данных (если заранее известны диапазоны их изменения);
- значения данных вне границ допустимого диапазона (например, при работе с натуральными числами — нулевые или отрицательные значения) — это необходимо для того, чтобы выявить, как алгоритм реагирует на заведомые ошибки[1)].

1) Как правило, в профессионально разработанных программах должен быть предусмотрен контроль вводимых значений исходных данных, обнаружение их несоответствия корректным диапазонам и либо исправление этих данных (если ситуация однозначна), либо выдача пользователю соответствующего сообщения об ошибке и запрос на повторный ввод данных.

Если программа предполагает ввод нескольких исходных данных, то необходимо предусмотреть в наборе тестов *все* возможные их комбинации.

Разработка корректного и полного набора тестов для каждой отлаживаемой программы представляет собой отдельную, достаточно сложную задачу. Однако сделать это необходимо. Иначе варианты исходных данных, приводящие к сбою в работе программы, могут выявиться уже при работе с ней её пользователей.

При отладке программ важно помнить следующее правило: *чем на более раннем этапе разработки будет обнаружена ошибка, тем легче и дешевле её исправить*. Так, обнаружение и исправление ошибки при разработке программы потребуют лишь повторной её отладки. Но если ошибка обнаружится уже при коммерческой эксплуатации созданной программы, то это приведёт к значительным потерям — как моральным (страдает репутация разработчика как создателя высококачественных программ), так и финансовым (необходимость информирования пользователей программы об ошибке, выпуска и распространения по всем пользователям исправленной версии программы и т. д.),

Вопросы и задания

1. Что такое отладка программы? Для чего она нужна?
2. Перечислите основные этапы отладки программы. Опишите назначение каждого этапа и характер обнаруживаемых с его помощью недочётов программы.
3. Что такое таблица трассировки? Как она составляется? Приведите свой пример программы и составленной для неё таблицы трассировки.
4. Как формируется набор тестов для тестирования программы? Какие варианты исходных данных рекомендуется включать в этот набор тестов?

*5. Приходилось ли вам при работе на компьютере сталкиваться со сбоями в работе программ? Как вы считаете — вызвано ли это недостаточно тщательным тестированием этих программ при их разработке? Предположите, какие меры разработчику этих программ потребовалось (или потребуется) предпринять для исправления этих ошибок и оцените примерные затраты на это исправление.

4.1.5. Типовые алгоритмы

Некоторые алгоритмы используются настолько часто, что считаются типовыми. Рекомендуется знать принципы построения этих алгоритмов, их принципы работы и уметь опознавать эти алгоритмы (уметь определять по виду алгоритма, какую задачу он решает).

Алгоритм нахождения наибольшего из нескольких чисел

Задано два, три, четыре или более исходных чисел. Требуется определить (и, например, вывести на экран) максимальное из них.

Принцип решения задачи

Исходные числа при их вводе записываются в соответствующие переменные (например, *a*, *b*, *c*).

Определить наибольшее из двух чисел позволяет оператор ветвления, условие в котором сводится к сравнению этих двух чисел:

```
ЕСЛИ a>b ТО Max = a ИНАЧЕ Max = b
```

Если требуется найти максимальное из трёх чисел, то проще всего использовать несколько операторов ветвления с соответствующими сложными условиями[1]:

```
ЕСЛИ (a>=b) И (a>=c) ТО Max = a
ЕСЛИ (b>=a) И (b>=c) ТО Max = b
ЕСЛИ (c>=a) И (c>=b) ТО Max = c
```

Возможен и алгоритм, использующий вложенные операторы ветвления (рис. 4.9).

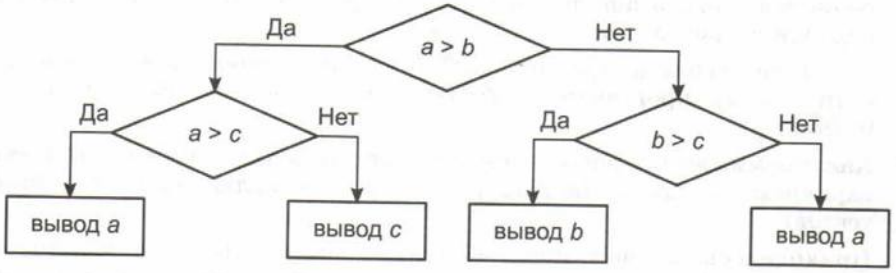


Рис. 4.9

1) В условии используется сравнение «больше или равно», чтобы обеспечить корректную работу алгоритма в случае равенства чисел (очевидно, для равных чисел в качестве «наибольшего» можно выбрать любое из них).

Если требуется найти максимальное из четырёх и более чисел, то используется прием последовательного сравнения чисел с «предполагаемым максимумом». Сначала мы предполагаем, что максимальным является первое число. Далее каждое число по очереди сравнивается с текущим значением предполагаемого максимума: если это число больше, чем предполагаемый максимум, то в качестве предполагаемого максимума берётся уже это число. В результате после проверки всех чисел в переменной, в которой хранится предполагаемый максимум, окажется наибольшее число.

```
Max = a
ЕСЛИ b>=Max ТО Max = b
ЕСЛИ c>=Max ТО Max = c
ЕСЛИ d>=Max ТО Max = d
...
```

Выполним трассировку алгоритма для исходных чисел 2, 3, 1, 5:

	Условие	*a*	*b*	*c*	*d*	*Max*
`Max = a`		2	3	1	5	2
ЕСЛИ `b >= Max` **ТО** `Max = b`	да	2	3	1	5	3
ЕСЛИ `c >= Max` **ТО** `Max = c`	нет	2	3	1	5	3
ЕСЛИ `d >= Max` **ТО** `Max = d`	да	2	3	1	5	5

Алгоритм поиска наименьшего из двух, трёх, четырёх и более чисел реализуется аналогично, но в условиях операторов ветвления используется сравнение «меньше» (либо «меньше или равно»).

Подобный алгоритм обладает *масштабируемостью*: его легко дополнять для обработки практически любого количества чисел. Однако для поиска максимума или минимума из множества числовых значений удобнее использовать массивы и циклы.

Анализ записи числа в позиционной системе счисления (разбор числа на цифры)

Задано десятичное число. Требуется выделять и обрабатывать его отдельные цифры (от младших разрядов к старшим).

Принцип решения задачи

Для разбора числа на цифры используется пара операций, как правило, имеющихся в любом языке программирования:

- поиск остатка от деления (операция MOD);
- целочисленное деление (операция DIV).

Очевидно, что операция вычисления остатка от деления исходного числа на 10 приводит к выделению последней цифры этого числа. Например: `12345 MOD 10 = 5`.

Операция целочисленного деления исходного числа на 10 приводит к «отсечению» последней цифры этого числа. Например: `12345 DIV 10 = 1234`.

Соответственно, для разбора исходного числа X на отдельные цифры можно использовать цикл с предусловием:

```
ПОКА X > 0 ВЫПОЛНЯТЬ
C = X MOD 10  // в переменную C выделяется
              // очередная цифра
  // обработка выделенной цифры
  X = X DIV 10  // от числа отсекается
                // обработанная цифра
КОНЕЦ ЦИКЛА
```

Если использовать операции MOD и DIV для деления не на 10, а на другое число, то алгоритм будет выполнять разбор исходного числа на цифры соответствующей системы счисления. Например, использование операций MOD 2 и DIV 2 позволяет выделять цифры записи числа в двоичной системе счисления.

Поиск НОД заданных натуральных чисел

Заданы два натуральных числа. Требуется вычислить их наибольший общий делитель (НОД).

Принцип решения задачи

Операция поиска **наибольшего общего делителя (НОД)** двух заданных натуральных чисел обычно выполняется по **алгоритму Евклида**, который был впервые описан греческим математиком Евклидом около 300 лет до нашей эры. Идея алгоритма Евклида очень проста: из большего числа вычитается меньшее до тех пор, пока числа не станут равны. Полученное значение — это и есть НОД двух исходных чисел (рис. 4.10).

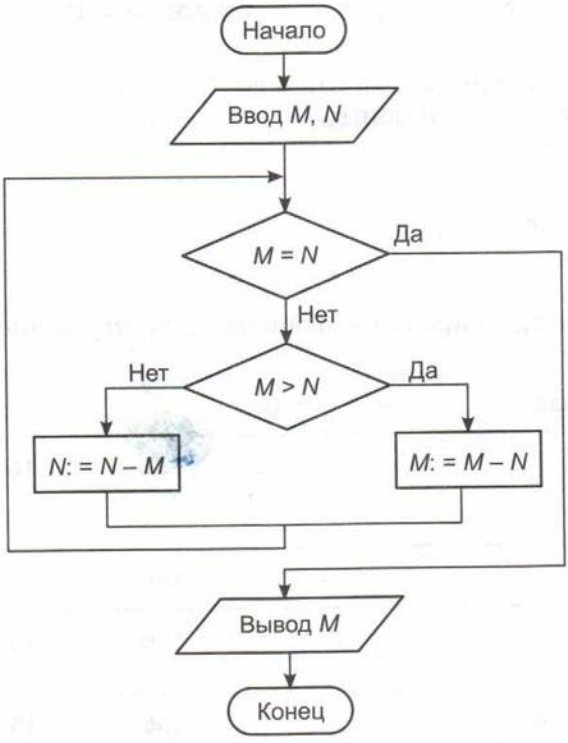


Рис. 4.10

Возможен также **модифицированный алгоритм Евклида**, работающий быстрее традиционного. Смысл модификации в том, что вместо последовательности нескольких одинаковых вычитаний меньшего числа из большего можно использовать операцию вычисления остатка от деления большего числа на меньшее. Например, для пары чисел 65 и 18:

Традиционный алгоритм Евклида	Модифицированный алгоритм Евклида
65 – 18 = 47	65 MOD 18 = 11
47 – 18 = 29	
29 – 18 = 11	

Для реализации модифицированного алгоритма Евклида используются две вспомогательные переменные. Кроме того, перед началом работы алгоритма необходимо обеспечить, чтобы первое из чисел обязательно было больше второго.

```
ЕСЛИ M > N ТО A = M, B = N ИНАЧЕ A = N, B = M
NOD = A
// теперь гарантировано, что A >B
ПОКА NOD <> 0 ВЫПОЛНЯТЬ
NOD = A MOD B
A = B, B = NOD
КОНЕЦ ЦИКЛА
ВЫВЕСТИ A
```

Выполним трассировку алгоритма для пары чисел 156 и 15:

Команда	NOD <> 0	A	B	NOD
ЕСЛИ M > N **ТО** A = M, B = N **ИНАЧЕ** A = N, B = M	–	156	15	–
NOD = A	–	156	15	156
ПОКА NOD <> 0 **ВЫПОЛНЯТЬ**	Да	156	15	156
NOD = A MOD B		156	15	6
A = B, B = NOD		15	6	6
ПОКА NOD <> 0 **ВЫПОЛНЯТЬ**	Да	15	6	6
NOD = A MOD B		15	6	3
A = B, B = NOD		6	3	3
ПОКА NOD <> 0 **ВЫПОЛНЯТЬ**	Да	6	3	3
NOD = A MOD B		6	3	0
A = B, B = NOD		3	0	0
ПОКА NOD <> 0 **ВЫПОЛНЯТЬ**	Нет	3	0	0
ВЫВЕСТИ A		3	0	0

Вы можете самостоятельно оценить выгодность модифицированного алгоритма Евклида, заменив операции A MOD B на соответствующее количество операций вычитания числа B из числа A и определив их количество.

Проверка числа на простоту

Задано натуральное число. Требуется определить, является ли это число простым.

Простое число — это натуральное число, которое делится нацело только на единицу или на само себя.

Принцип решения задачи

Обычно такая задача решается **методом перебора делителей** — берётся исходное число N, и производятся попытки разделить его нацело на все натуральные числа от 2 до $N - 1$. При этом заранее предусматривается отдельная *переменная-флаг*:

- изначально ей присваивается значение, которое будет обозначать, что деление нацело невозможно (например, значение 0);
- в цикле имеется оператор ветвления, условие которого проверяет, делится ли исходное число нацело на очередной проверяемый делитель; если да, то в переменную-флаг записывается другое значение (например, 1).

После завершения работы цикла отдельный оператор ветвления проверяет значение переменной-флага: если в ней осталось значение 0, то можно сделать вывод, что исходное число простое (ни один проверяемый делитель не подошёл), а если её значение изменилось на 1, это означает, что исходное число не простое.

Ниже приведен фрагмент алгоритма определения простоты числа N. Для корректного решения задачи нужно также добавить перед этим фрагментом операторы ветвления, контролирующие возможную ошибкуввода отрицательного или нулевого числа, а также числа, равного 1 (которое, как известно из математики, хотя и удовлетворяет условию «делится только на 1 и на само себя», но простым числом не является).

```
FLAG = 0 // исходное значение переменной-флага
ЦИКЛ I ОТ 2 ДО N - 1 С ШАГОМ 1
ЕСЛИ N MOD I = 0 ТО FLAG = 1
  // если остаток от деления отсутствует,
  // то число N делится нацело, и оно не простое
КОНЕЦ ЦИКЛА
ЕСЛИ FLAG = 1 ТО ВЫВЕСТИ "Число составное"
              ИНАЧЕ ВЫВЕСТИ "Число простое"
```

Можно сделать этот алгоритм более оптимальным. Вспомним, что составное число можно представить как произведение меньшего сомножителя на больший, например:

$$12 = \underbrace{2 \cdot 6 = 3 \cdot 4} = \underbrace{4 \cdot 3 = 6 \cdot 2}.$$

Обратите внимание на группы умножений, выделенные скобками: они «зеркально симметричны». А значит если деление исходного числа нацело обнаружено для какого-то из меньших сомножителей (в первой группе умножений они записаны первыми), то проверять соответствующие им бóльшие сомножители уже не требуется. «Граница» между этими «симметричными» группами умножений примерно соответствует квадратному корню из исходного числа, поэтому в описанном выше алгоритме достаточно выполнять цикл не до $N - 1$, а до ближайшего натурального значения, большего $\sqrt{N}$. При этом используются две стандартные подпрограммы-функции, которые есть в любом языке программирования:

SQRT() — вычисление квадратного корня от значения, заданного в скобках;
INT() — определение целой части вещественного значения, заданного в скобках (в отличие от математической операции округления, эта функция просто отбрасывает дробную часть заданного числа, поэтому для округления в бóльшую сторону нужно к полученному целому значению дополнительно прибавить единицу).

```
FLAG = 0 // исходное значение переменной-флага
ЦИКЛ ДЛЯ I ОТ 2 ДО INT(SQRT(N))+1 С ШАГОМ 1
ЕСЛИ N MOD I = 0 ТО FLAG = 1
  // если остаток от деления отсутствует,
  // то число Nделится нацело, и оно не простое
КОНЕЦ ЦИКЛА
ЕСЛИ FLAG = 1 ТО ВЫВЕСТИ "Число составное"
              ИНАЧЕ ВЫВЕСТИ "Число простое"
```

Вопросы и задания

1. Опишите основную идею алгоритмов поиска наибольшего/наименьшего из нескольких заданных чисел. Объясните принцип работы алгоритма поиска максимального/минимального значения путём последовательного сравнения чисел с предполагаемым максимумом/минимумом.

2. Объясните принцип работы алгоритма разбора десятичной записи числа на отдельные цифры.
*3. Напишите на основе наразбора записи числа на отдельные цифры свой алгоритм перевода десятичного числа в его запись в двоичной системе счисления.
4. Объясните принцип работы модифицированного алгоритма Евклида. На примере определения НОД двух конкретных выбранных вами чисел продемонстрируйте и оцените выгодность модифицированного алгоритма Евклида по сравнению с традиционным (считайте для простоты, что каждая команда алгоритма выполняется в течение 0,1 секунды и сравните соответствующие затраты времени).
5. Как выполняется проверка заданного числа на простоту? Объясните суть идеи оптимизированного алгоритма проверки числа на простоту.

4.2. История развития языков программирования

Выполнение алгоритма может быть автоматически реализовано техническими устройствами, среди которых особое место занимает компьютер. При этом говорят, что компьютер исполняет программу (последовательность команд), реализующую алгоритм.

Алгоритм, записанный на «понятном» компьютеру языке программирования, называется **программой.**

Машинный язык. На заре компьютерной эры, в 40–50-е годы XX века, программы писались на машинном языке и представляли собой очень длинные последовательности нулей и единиц. Составление и отладка таких программ были чрезвычайно трудоёмким делом. Программы на машинных языках были машинно зависимыми, т. е. для каждой ЭВМ необходимо было создавать свою собственную программу, так как в ней в явной форме учитывались аппаратные ресурсы ЭВМ.

Ассемблер. В начале 1950-х годов были созданы языки программирования ассемблеры. Вместо одних только нулей и единиц программисты теперь могли пользоваться операторами (`MOV`, `ADD`, `SUB` и т. д.), которые были похожи на слова английского языка. Для преобразования текста программы на ассемблере в «понятный» компьютеру машинный код использовался компилятор, который загружался в оперативную память ЭВМ.

Программы на ассемблере были также машинно зависимыми, т. е. ассемблеры для различных процессоров существенно различались между собой.

Первые языки программирования высокого уровня. С середины 1950-х годов начали создаваться первые языки программирования высокого уровня. Эти языки были машинно независимыми, так как использовали универсальную компьютерную логику и не были привязаны к типу ЭВМ. Однако для каждого языка и каждого типа ЭВМ требовалась разработка собственных компиляторов, которые загружались в оперативную память.

Языки программирования высокого уровня создавались и использовались для решения разных задач:

- язык FORTRAN (*FORmula TRANslator* — транслятор формул) был предназначен для научных и технических расчётов;
- язык COBOL (*Common Business-Oriented Language* — стандартный язык для делового применения) в основном предназначался для коммерческих приложений, обрабатывавших большие объёмы нечисловых данных;
- язык BASIC (*Beginner's All-Purpose Symbolic Instruction Code* — универсальный язык символьных инструкций для начинающих) отличается простотой изучения и был ориентирован на начинающих программистов.

Алгоритмические языки программирования. С начала 1980-х годов начали создаваться алгоритмические языки программирования, которые позволили программистам перейти к структурному программированию. Отличительной чертой этих языков было использование операторов ветвления, выбора и цикла и отказ от хаотического использования оператора `goto`. Наибольшее влияние на переход к структурному алгоритмическому программированию оказали:

- язык Pascal (назван его создателем Никлаусом Виртом в честь Блеза Паскаля) — алгоритмический язык, который позволяет легко кодировать основные алгоритмические структуры;
- язык C (произносится «Си»), позволяющий создавать быстро и эффективно выполняющийся программный код.

Языки объектно-ориентированного программирования. С 1990-х годов начали создаваться объектно-ориентированные языки программирования. В основу этих языков были положены программные объекты, которые объединяли данные и методы их обработки. Необходимо подчеркнуть, что в языках объектно-ориентированного программирования сохранялся алгоритмический

стиль программирования. С течением времени для этих языков были разработаны интегрированные среды разработки, позволяющие визуально конструировать графический интерфейс приложений:

- язык C++ является прямым потомком алгоритмического языка C;
- язык Object Pascal был разработан компанией Borland на основе алгоритмического языка Pascal. После создания интегрированной среды разработки система программирования получила название Delphi, а кроссплатформенная свободно распространяемая версия — Lazarus;
- язык Visual Basic был создан корпорацией Microsoft на основе языка QBasic для разработки приложений с графическим интерфейсом в среде операционной системы Windows.

Языки программирования для компьютерных сетей. В 1990-е годы в связи с бурным развитием Интернета были созданы языки, обеспечивающие межплатформенную совместимость:

- язык Java, полноценный объектно-ориентированный язык, был разработан фирмой Sun Microsystems для создания сетевого программного обеспечения;
- язык JavaScript, язык сценариев веб-страниц, разработан компанией Netscape.

На подключённых к Интернету компьютерах с различными операционными системами (Windows, Linux, macOS и др.) могли выполняться одни и те же программы. Исходная программа на таких языках компилируется в промежуточный код, который исполняется на компьютере встроенной в браузер виртуальной машиной.

Языки программирования на платформе .NET. В настоящее время используют интегрированную систему программирования Visual Studio на платформе .NET Framework, разработанную корпорацией Microsoft. Эта платформа предоставляет возможность создавать приложения в различных системах объектно-ориентированного программирования, в которых для создания программного кода используются объектно-ориентированные языки программирования, в том числе:

- на языке Visual Basic .NET, созданном на основе языка Visual Basic и сохраняющем простоту своих предшественников;
- на языке Visual C# (читается «С-шарп»), созданном на основе языков C++ и Java.

Вопросы и задания

1. В чём состоит преимущество языков высокого уровня перед машинным языком и ассемблерами?
2. Каковы характерные особенности алгоритмических языков программирования? Объектно-ориентированных языков программирования?
3. Какими причинами было обусловлено появление языков Java и JavaScript?
4. На основе каких языков возникли языки программирования на платформе .NET?

4.3. Введение в объектно-ориентированное программирование

4.3.1. Объекты: свойства и методы

Объекты (`Objects`). Основной единицей в объектно-ориентированном программировании является **программный объект**, который объединяет в себе как описывающие его **свойства**, так и действия объекта (процедуры) — **методы**. Если говорить образно, то объекты — это «существительные», свойства объекта — это «прилагательные», а методы объекта — это «глаголы».

Программные объекты обладают **свойствами**, имеют **методы** и для них можно описать реакцию на **события**.

Классы объектов являются «шаблонами», определяющими наборы свойств, методов и событий, по которым создаются объекты. Основными классами объектов являются объекты, реализующие графический интерфейс проектов.

Объект, созданный по «шаблону» класса объектов, является **экземпляром класса** и **наследует** весь набор свойств, методов и событий данного класса. Каждый экземпляр класса имеет уникальное для данного класса имя. Различные экземпляры класса обладают одинаковым набором свойств, однако значения этих свойств у них могут различаться.

Свойства объектов (`Properties`). Каждый объект обладает определённым набором свойств. Существует несколько основных ситуаций, в которых можно менять свойства объектов. Во время разработки проекта `[design]` можно установить первоначальные значения свойств объекта.

В режиме выполнения проекта [**run**] можно устанавливать или менять значения свойств объекта в ходе исполнения программного кода. Для присваивания свойству объекта нового значения необходимо указать в левой части строки программного кода имя объекта, а затем — название свойства. В правой части строки необходимо записать конкретное значение свойства. Например, программный код вывода в поле с именем `Label` текста в различных языках программирования будет выглядеть следующим образом.

Язык Visual Basic .NET:[1)]
```
Label1.Text = "Текст"
```

Язык Visual C#:
```
label1.Text = "Текст";
```

Язык Lazarus:
```
Label1.Caption := 'Текст';
```

В программном коде для доступа к свойствам и методам используется **точечная нотация (dot-запись)**, при которой имена объектов, свойств и методов отделяются друг от друга знаком точки «.».

Методы объектов (Methods). Чтобы объект выполнил какую-либо операцию, необходимо применить метод, которым он обладает. Многие методы имеют аргументы, которые позволяют задать параметры выполняемых действий. Обратиться к методу объекта можно тоже с использованием точечной нотации, причём аргументы метода заключаются в скобки. Например, для добавления элемента в список в языках объектно-ориентированного программирования используется метод **Add()**.

Язык Visual Basic .NET:
```
ListBox1.Items.Add("Элемент списка")
```

Язык Visual C#:
```
listBox1.Items.Add("Элемент списка");
```

Язык Lazarus:
```
ListBox1.Items.Add('Элемент списка');
```

1) Набранные тексты этой и следующих программ размещены в электронном приложении к главе 4.

Вопросы и задания

1. Чем различаются понятия «класс объектов» и «экземпляр класса»?
2. В экземпляре класса можно изменить набор свойств? Набор методов? Значения свойств?

4.3.2. События

События (Events). Событие представляет собой изменение некоторого состояния, распознаваемое объектом. Для реакции на это изменение могут быть описаны те или иные методы — обработчики, обрабатывающие события в программном коде. Событие может создаваться пользователем (например, щелчок мышью или нажатие клавиши) или быть результатом воздействия других программных объектов. Каждый элемент управления может реагировать на различные события, однако есть события (например, `Click` — щелчок мышью), на которые реагирует большинство типов элементов управления.

Обработчик события. Для каждого события можно запрограммировать обработчик события (*событийную процедуру*). Если пользователь производит какое-либо воздействие на элемент графического интерфейса (например, щелчок мышью), в качестве обработчика выполняется некоторая последовательность действий в форме процедуры.

Каждый обработчик события представляет собой процедуру, которая реализует определённый алгоритм. Создание программного кода обработчика события производится с использованием алгоритмических структур различных типов (линейная, ветвление или цикл).

Обработчик события представляет собой процедуру, которая начинает выполняться после реализации определённого события.

Имя обработчика события (событийной процедуры) включает в себя имя объекта и имя события. После имени событийной процедуры в скобках указываются параметры, которые позволяют правильно обработать событие. В событийной процедуре на языках .NET существуют два параметра, а в языке Lazarus — один параметр.

Далее на трёх языках программирования показан пустой обработчик события `Click` элемента управления «кнопка» — `Button1`.

Язык Visual Basic .NET:

```
Private Sub Button1_Click(sender As Object,
e As EventArgs) Handles Button1.Click
End Sub
```

Язык Visual C#:

```
private void button1_Click(object sender,
EventArgs e)
{
}
```

Язык Lazarus:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
end;
```

Первый параметр, sender, предоставляет ссылку на объект, который вызывает событие. Например, при щелчке мышью по кнопке наступает событие **Click** данной кнопки и её адрес передаётся обработчику события и сохраняется в аргументе sender.

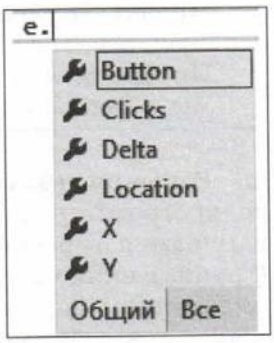


Рис. 4.11

В языках программирования на платформе .NET используется и второй параметр e, который передаёт данные, характерные для обрабатываемого события. Этот параметр обычно имеет тип **EventArgs**, однако существуют события, которые требуют особого типа данных. Например, если обрабатываются события мыши, то используется параметр MouseEventArgs. С помощью этого параметра можно получить сведения о координатах мыши, какая была нажата кнопка и сколько было сделано щелчков. Для вывода всех свойств аргумента e (рис. 4.11) достаточно ввести в процедуре обработчика события: e.

Вопросы и задания

В чём состоит различие между обработчиками событий в языках программирования на платформе .NET и в языке Lazarus?

4.3.3. Проекты и приложения

Проект (Project). С одной стороны, система объектно-ориентированного визуального программирования является системой про-

граммирования, так как позволяет кодировать алгоритмы на данном языке. С другой стороны, система объектно-ориентированного визуального программирования является средой проектирования, так как позволяет осуществлять визуальное конструирование графического интерфейса.

Результатом процессов программирования и проектирования является проект, который объединяет в себе программный код и графический интерфейс.

В объектно-ориентированном программировании проект может включать несколько **форм**, причём каждой форме, с помощью которой реализуется графический интерфейс проекта, соответствует свой программный модуль формы. Подробно о графическом интерфейсе проекта рассказано в параграфе 4.7.

Кроме того, в состав проекта могут входить отдельные самостоятельные программные модули.

Проект включает в себя **программные модули форм** и самостоятельные **программные модули** в виде отдельных файлов. Проект может быть запущен на выполнение только из системы объектно-ориентированного программирования.

Решения (Solution). В системах объектно-ориентированного программирования Visual Basic .NET и Visual C# проекты объединяются в решения, а в системе Lazarus — в группы. Решение (группа) включает один или несколько проектов, которые в упорядоченном виде в системах Visual Basic .NET и Visual C# отображаются в *Обозревателе решений*, а системе Lazarus — в окне *Обозреватель кода*. Решение (группа) создаётся автоматически при создании нового проекта, а при необходимости к решению можно добавлять новые проекты. Решения (группы) позволяют работать с несколькими проектами в пределах одного экземпляра системы объектно-ориентированного программирования.

Интерпретаторы и компиляторы. Чтобы процессор мог выполнить программу, эта программа и данные, с которыми она работает, должны быть загружены в оперативную память.

Итак, мы создали программу на языке программирования (некоторый текст) и загрузили её в оперативную память. Теперь мы хотим, чтобы процессор её выполнил, однако процессор «понимает» команды на машинном языке, а наша программа написана на языке программирования. Как быть?

Необходимо, чтобы в оперативной памяти находилась программа-переводчик **(транслятор)**, автоматически переводящая

нашу программу с языка программирования на машинный язык. Компьютер может выполнять программы, написанные только на том языке программирования, транслятор которого размещён в оперативной памяти компьютера.

Трансляторы языков программирования бывают двух типов: интерпретаторы и компиляторы.

Интерпретатор — это программа, которая обеспечивает последовательный «перевод» инструкций программы на машинный язык с одновременным их выполнением. Поэтому при каждом запуске программы на выполнение эта процедура повторяется. Достоинством интерпретаторов является удобство отладки программы (поиска в ней ошибок), так как возможно пошаговое её исполнение, а недостатком — сравнительно малая скорость выполнения.

Компилятор действует иначе. Он переводит весь текст программы на машинный язык и сохраняет его в исполняемом файле (обычно с расширением exe). Затем этот уже готовый к исполнению файл, записанный на машинном языке, можно запускать на исполнение многократно. Достоинством компиляторов является большая скорость выполнения программы, а недостатком — трудоёмкость отладки, так как невозможно пошаговое выполнение программы.

Системы объектно-ориентированного программирования позволяют программисту контролировать в интегрированной среде выполнение программ с помощью **отладчика**. Это даёт возможность отлаживать программу пошагово.

Итак, как мы отмечали ранее, сохранённый проект может выполняться только в самой системе программирования. Чтобы преобразовать проект в *приложение*, которое может выполняться непосредственно в среде операционной системы, необходимо выполнить компиляцию проекта, в процессе которой приложение сохраняется в исполняемом файле (с расширением exe).

Приложение интегрирует программный код и графический интерфейс в одном исполняемом файле, который может запускаться непосредственно в операционной системе.

Этапы разработки проектов. Создание проектов и приложений в системах объектно-ориентированного программирования можно условно разделить на несколько этапов.

1. *Создание графического интерфейса проекта.* На форму помещаются элементы управления, которые должны обеспечить взаимодействие проекта с пользователем.

2. *Установка значений свойств объектов графического интерфейса.* В режиме конструирования задаются значения свойств формы и элементов управления, помещённых ранее на форму.
3. *Создание и редактирование программного кода.* Создаются заготовки обработчиков событий (двойной щелчок мышью по элементу управления вызывает заготовку обработчика события, которое для данного элемента управления используется наиболее часто). Затем в редакторе программного кода производится ввод и редактирование программного кода обработчиков событий.
4. *Сохранение проекта.* Так как проекты включают в себя несколько файлов, рекомендуется для каждого проекта создать отдельную папку на диске. Сохранение проекта производится с помощью пунктов меню *Файл*.
5. *Компиляция проекта в приложение.* Создаётся приложение — исполняемый файл (exe).

Вопросы и задания

1. В чём состоит различие между интерпретаторами и компиляторами?
2. В чём состоит различие между проектом и приложением?

4.4. Система объектно-ориентированного программирования Microsoft Visual Studio

4.4.1. Интегрированная среда разработки языков Visual Basic .NET и Visual C#

Состав Visual Studio. Microsoft Visual Studio — это инструмент разработки программ, позволяющий писать программы на нескольких языках программирования .NET. В состав Visual Studio[1)] входят следующие языки программирования (рис. 4.12):

- Visual Basic .NET;
- Visual C# (произносится «Си-шарп»);
- Visual C++ (произносится «С плюс плюс»).

1) Здесь и далее в зависимости от используемых версий Visual Studio и Lazarus вид диалоговых окон может несколько различаться, но смысл имеющихся на них надписей и диалоговых элементов (кнопок, меню и пр.) остаётся одним и тем же.

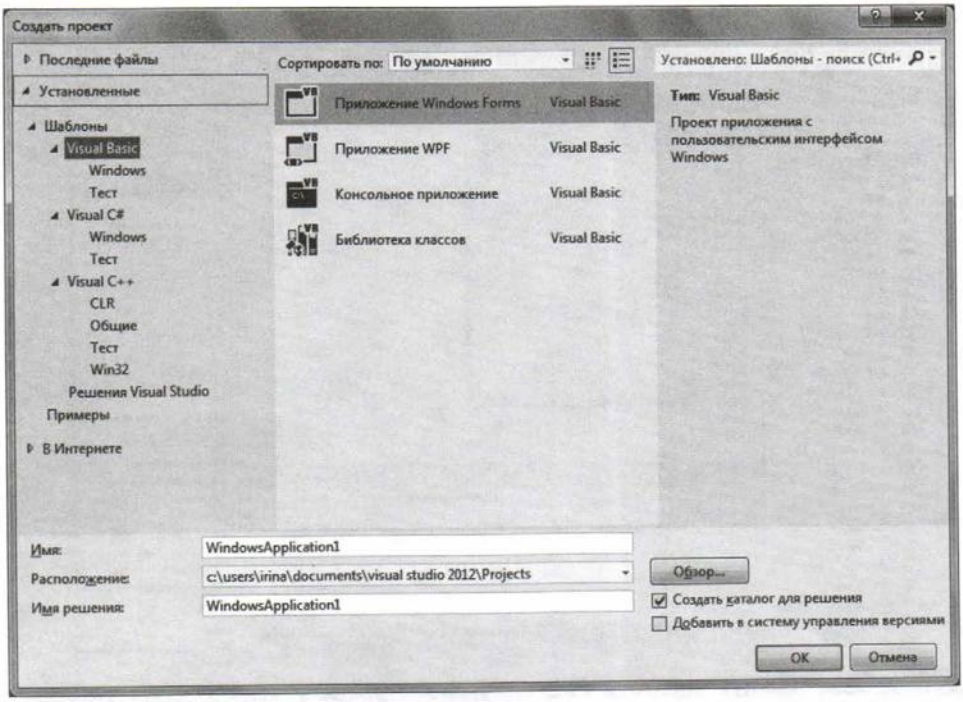


Рис. 4.12

Интегрированная среда разработки. Visual Studio является системой визуального объектно-ориентированного программирования на платформе .NET Framework. Система программирования Visual Studio предоставляет пользователю для работы со всеми языками .NET удобный, причём одинаковый, графический интерфейс **IDE** (*Integrated Development Environment* — интегрированная среда разработки) — см. на примере Visual Basic (рис. 4.13).

Визуальное конструирование графического интерфейса проекта выполняется в **конструкторе форм** (*Windows Forms Designer*), который по умолчанию размещается на вкладке *Form1.vb [Конструктор]*. Конструктор форм располагается в центре окна интегрированной среды разработки и содержит **форму** (по умолчанию **Form1**), являющуюся основой графического интерфейса проекта. На **форму** можно поместить различные **элементы управления**: кнопки (`Button`), текстовые поля (`TextBox`), надписи (`Label`) и т. д.

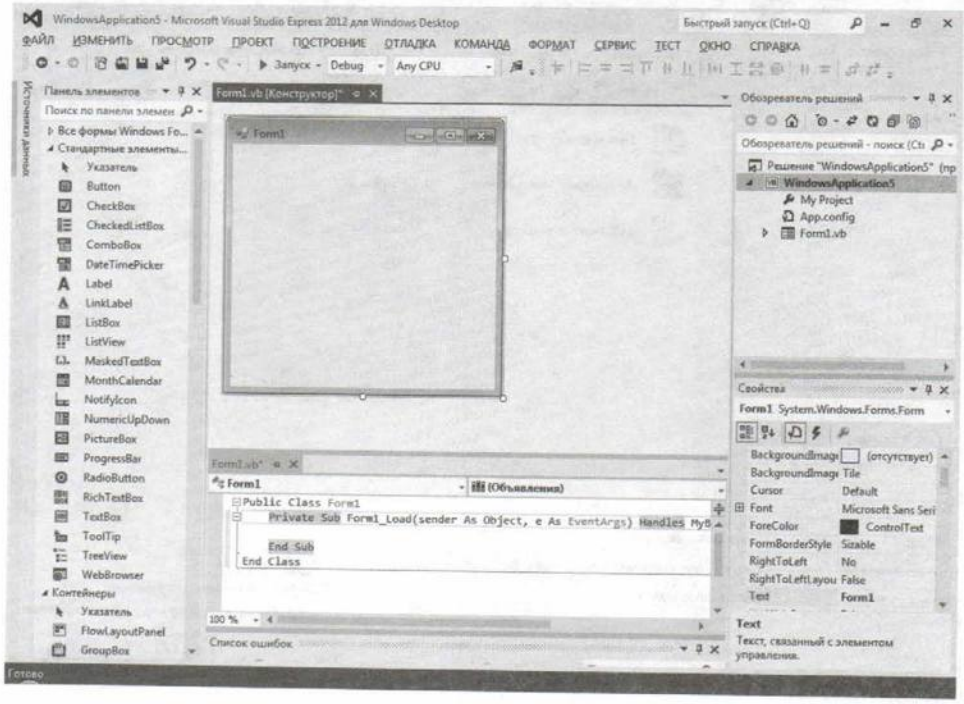


Рис. 4.13

Пиктограммы элементов управления располагаются на панели *Панель элементов*, которая размещается в левой части окна интегрированной среды разработки.

С формой связан программный код проекта, для ввода и редактирования которого служит **редактор программного кода** (по умолчанию размещается на вкладке *Form1.vb*). Редактор программного кода поддерживает язык проекта, т. е. предлагает подсказки при вводе программы, выделяет синтаксические ошибки, структурирует программный код отступами и т. д.

Для перехода в окно программного кода можно просто щёлкнуть по ярлыку вкладки *Form1.vb* или ввести команду [*Просмотр—Код*]. Для обратного перехода в конструктор форм можно щёлкнуть по ярлыку вкладки *Form1.vb* [*Конструктор*] или ввести команду [*Просмотр—Конструктор*].

Первые части названий окна конструктора форм и окна редактора программного кода совпадают не случайно, так как в них создаётся и редактируется один и тот же файл `Form1.vb`. Этот файл и другие файлы проекта можно увидеть в окне *Обозреватель решений*, которое размещается в правом верхнем углу интегрированной среды разработки.

Справа внизу располагается окно *Свойства*. Оно содержит список свойств, относящихся к выбранному объекту (форме или элементу управления на форме). В левом столбце находятся названия свойств, а в правом — их значения. Установленные по умолчанию значения могут быть изменены.

Настройка интегрированной среды разработки. Рекомендуется провести настройку интегрированной среды разработки. Для этого необходимо ввести команду [*Сервис—Параметры...*] и в появившемся диалоговом окне *Параметры* установить нужные параметры для элементов интерфейса: *Конструктор Windows Form*, *Отладка* и др.

В процессе создания графического интерфейса проекта важно упорядоченно разместить на форме элементы управления. Для этого необходимо в левой части диалогового окна *Параметры* выбрать пункт *Конструктор Windows Form*, а в правой части установить шаг сетки на форме и её видимость (рис. 4.14).

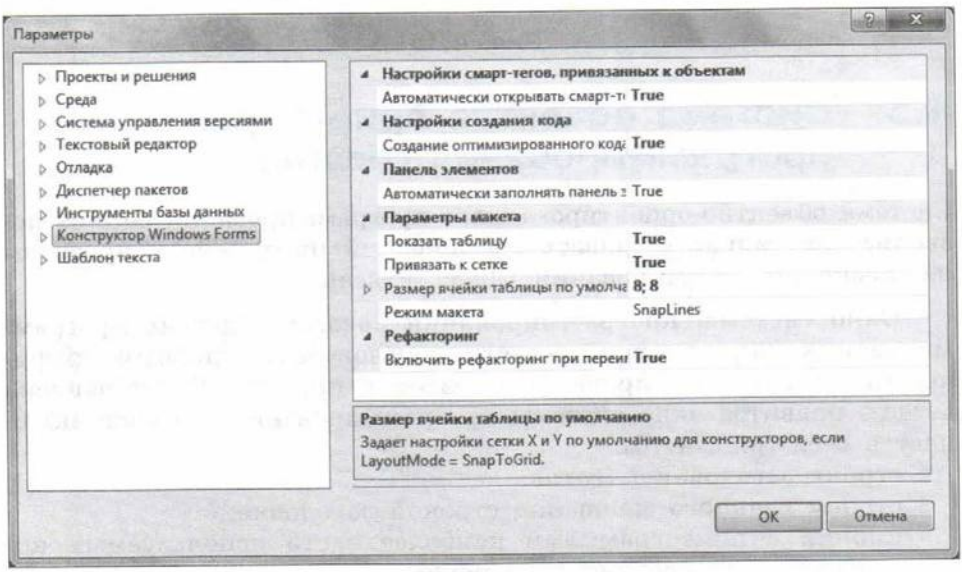


Рис. 4.14

В процессе создания, редактирования и отладки программного кода проекта целесообразно пронумеровать строки программы. Для этого необходимо в левой части диалогового окна *Параметры* выбрать пункт [*Текстовый редактор—Все языки*], а в правой части установить флажок *Номера строк* (рис. 4.15).

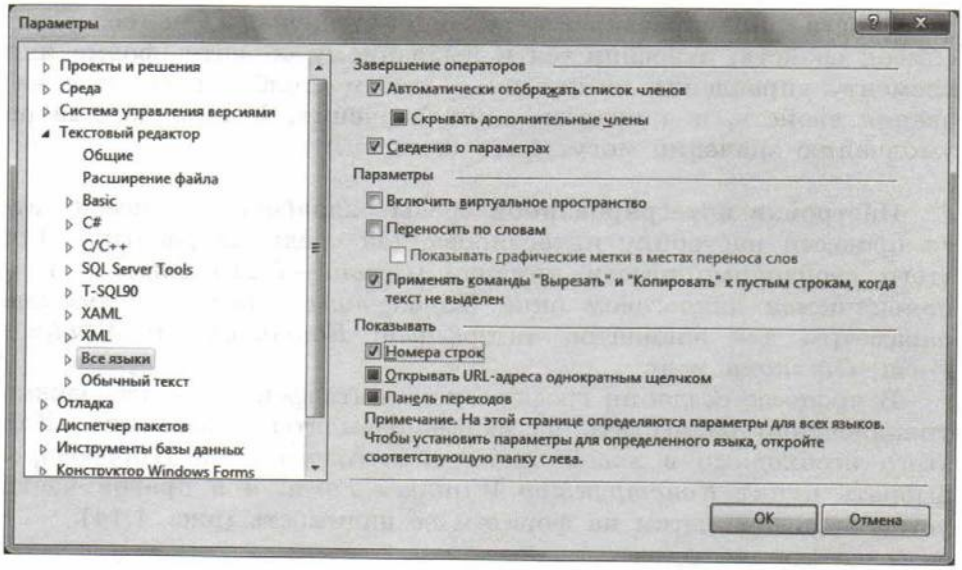


Рис. 4.15

4.5. Система объектно-ориентированного программирования Lazarus

Система объектно-ориентированного программирования Lazarus позволяет визуально создавать графический интерфейс к проектам на языке программирования Object Pascal.

Окно системы программирования Lazarus. Система программирования Lazarus предоставляет пользователю удобный графический интерфейс в процессе разработки проекта. После запуска Lazarus появится окно системы программирования, которое включает в себя (рис. 4.16):

- строку заголовка *Lazarus*;
- строку **главного меню** под строкой заголовка;
- кнопки с пиктограммами наиболее часто используемых команд под строкой главного меню.

Окно *Конструктор форм*. В центре располагается окно *Конструктор форм*. Для создания нового проекта необходимо ввести команду [*Проект—Создать проект*]. Окно представляет собой **форму** (в данном случае *Form1*), на которой происходит визуальное конструирование графического интерфейса разрабатываемого проекта. Размеры формы можно менять, перетаскивая мышью правую или нижнюю границу формы.

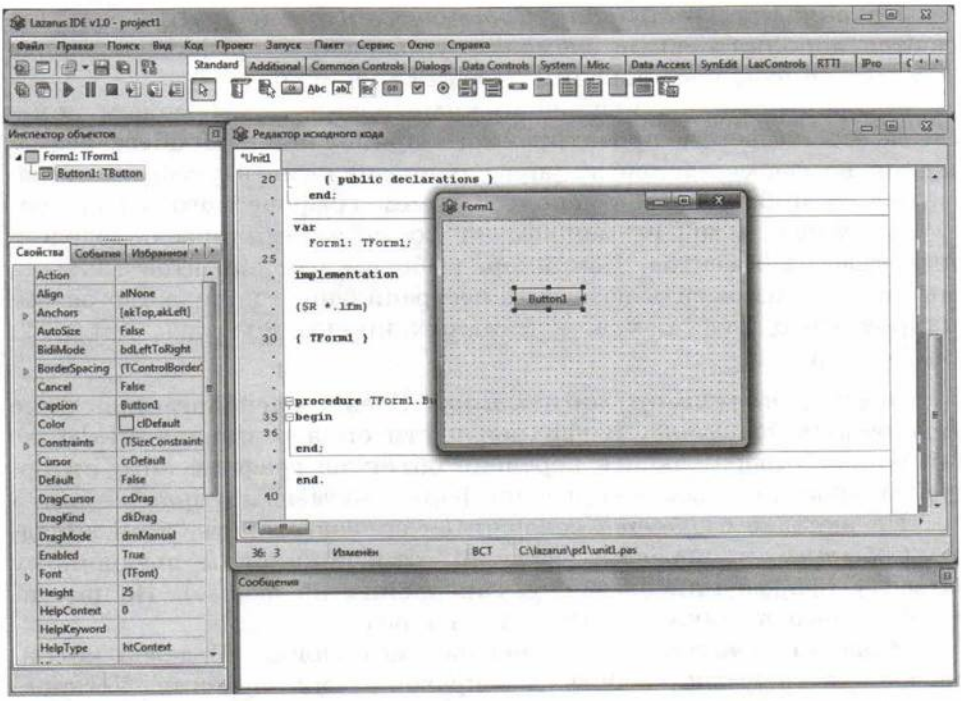


Рис. 4.16

Окно *Конструктор форм* вызывается командой [*Вид—Переключить форму/модуль*].

Первоначально форма пуста, в дальнейшем в процессе создания графического интерфейса проекта на ней размещаются **элементы управления.**

Окно *Редактор исходного кода*. С формой связан программный модуль, содержащий программные коды процедур. Для ввода и редактирования текста программы служит окно *Редактор исходного кода* (в данном случае *Unit1.pas*), которое вызывается командой [*Вид—Редактор исходного кода*].

Вкладки элементов управления. Эти вкладки содержат пиктограммы элементов управления. Стандартный набор элементов управления размещается на вкладке *Standard* и включает 18 классов объектов — это командная кнопка `TButton`, текстовое поле `TEdit`, надпись `TLabel` и т. д.

На вкладках *Additional*, *Common Controls* и других располагаются дополнительные элементы управления: графическое поле `TImage`, список изображений `TImageList` и др.

Выбрав щелчком мышью нужный элемент, мы можем поместить его на форму проектируемого приложения. Процесс размещения на форме элементов управления аналогичен рисованию графических примитивов с использованием графического редактора.

Фактически мы размещаем на форме экземпляры определённых классов объектов. Например, выбрав класс объектов `TButton`, мы можем разместить на форме неограниченное количество экземпляров этого класса, т. е. командных кнопок `Button1`, `Button2`, `Button3` и т. д.

Окно *Инспектор объектов*. Слева располагается окно *Инспектор объектов*. В верхней части окна размещается *дерево объектов*, отображающее перечень объектов графического интерфейса проекта (размещаемые на форме элементы управления).

На вкладке *Свойства* содержится список свойств, а на вкладке *События* — перечень событий, относящихся к выбранному объекту (форме или элементу управления на форме). На рисунке 4.10 выбран объект `Button1` из класса `TButton`.

Вкладка *Свойства* разделена на две колонки. В левой колонке находятся имена свойств, а в правой — их значения. Установленные по умолчанию значения могут быть изменены. Свойством объекта является количественная или качественная характеристика этого объекта (размеры, цвет, шрифт и др.). Для некоторых свойств предусмотрена возможность выбора из раскрывающегося списка значений, например, из списка можно выбрать значение цвета фона элемента управления (свойство `Color`).

Вкладка *События* также разделена на две колонки. В левой колонке находятся имена событий, а в правой можно ввести их значения.

Окно *Инспектор объектов* вызывается командой [*Вид—Инспектор объектов*].

Практическая работа 4.1

Создание проекта «Консольное приложение»

Задание. В системе объектно-ориентированного программирования Visual Studio создать консольное приложение, т. е. приложение, не имеющее графического интерфейса. Приложение должно выводить название языка программирования в окне командной строки.

Электронное приложение к главе 4.

Создание проекта «Консольное приложение» на языках программирования Visual Basic .NET и Visual C#

1. Запустить систему Microsoft Visual Studio.
2. Создать новый проект командой [*Файл—Создать проект...*]. В появившемся диалоговом окне *Создать проект* выбрать язык программирования Visual Basic .NET или Visual C# и тип приложения *Консольное приложение*. В окне *Код* появится заготовка программы.

Создание программного кода на языке Visual Basic .NET

3. Ввести программный код пользователя между служебными словами:

```
Sub Main()
  ...
End Sub
```

Для вывода текста в окно программной строки использовать оператор `Console.WriteLine()`
Чтобы консольное приложение не закрылось и можно было увидеть результаты работы программы, в конце программы разместить оператор `Console.Read()`.

```
Sub Main()
  Console.WriteLine ("Язык программирования Visual
                      Basic")
  Console.Read()
End Sub
```

Теперь можно запустить консольное приложение.

4. Ввести команду [*Отладка—Начать отладку*]. В окне командной строки будет выведено (рис. 4.17):

```
Язык программирования Visual Basic
```

Рис. 4.17

Создание программного кода на языке Visual C#

3. На языке Visual C# программный код пишется между фигурными скобками после служебных слов.

```
static void Main(string[] args)
{
  ...
}
```

Ввести программный код:

```
static void Main(string[] args)
{
  Console.WriteLine("Язык программирования C#");
  Console.Read();
}
```

Теперь можно запустить консольное приложение.

4. Ввести команду [*Отладка—Начать отладку*]. В окне командной строки будет выведено (рис. 4.18):

```
Язык программирования C#
```

Рис. 4.18

Создание проекта «Консольное приложение» на языке программирования Lazarus

1. Запустить систему программирования Lazarus.
2. Создать новый проект командой [*Проект—Создать проект...*]. В появившемся диалоговом окне *Создать новый проект* выбрать тип приложения *Консольное приложение* (рис. 4.19).

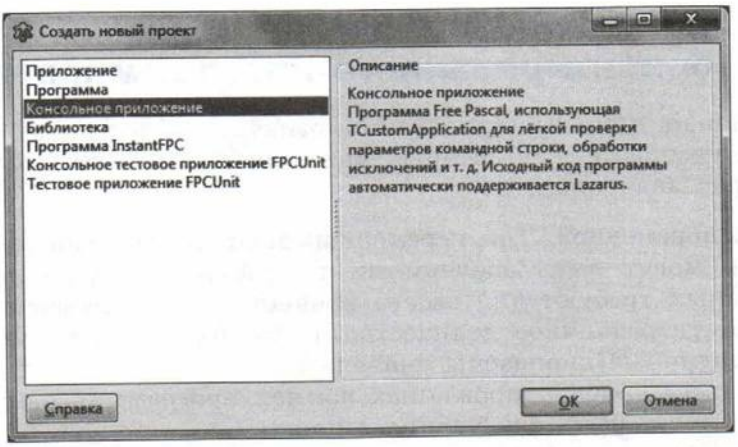


Рис. 4.19

3. В окне *Редактор исходного кода* появится заготовка программы.
   Ввести программный код пользователя между служебными словами **begin** и **end.**

Для вывода текста в окно программной строки использовать оператор `writeln()`.

```
begin
  writeln('Language Lazarus');
  readln;
end.
```

4. Сохранить программный код консольного приложения командой [*Файл—Сохранить как...*].

Теперь можно запустить консольное приложение.

5. Ввести команду [*Запуск—Запустить*].
   В окне командной строки будет выведено (рис. 4.20):
   `Language Lazarus`

Рис. 4.20

4.6. Переменные в языках объектно-ориентированного программирования

Переменная в программировании означает область оперативной памяти компьютера, в которую может быть занесено и храниться некоторое значение.

Тип переменной. Тип переменной определяется типом данных, которые могут быть значениями переменной. Различные типы переменных требуют для своего хранения в оперативной памяти компьютера различное количество ячеек (байтов) и могут принимать различные диапазоны значений.

В объектно-ориентированных языках программирования переменные могут быть следующих типов:

- **Byte**, **Short** (**SmallInt** в Lazarus), **Integer** (**Int** в Visual C#) и **Long** (**LongInt** в Lazarus) — значениями являются целые числа;
- **Single** (**float** в Visual C#, **real** в Lazarus), **Double** и **Decimal** (в Visual Basic .NET и Visual C#) — значениями являются вещественные числа;
- **Char** и **String** — значениями являются символы и последовательности символов;
- **Boolean** (**bool** в Visual C#) — переменные принимают логические значения **True** (истина) или **False** (ложь).

Имя переменной. Имя переменной обозначает адрес ячейки пямяти, где хранится значение переменной.

Имя каждой переменной (идентификатор) уникально и не может меняться в процессе выполнения программы. Имя переменной:

- должно начинаться с буквенного символа или с подчёркивания «_»;
- может содержать только буквенные символы, десятичные цифры и подчёркивания;
- должно содержать, по крайней мере, один буквенный или цифровой символ, если оно начинается с подчёркивания.

Рекомендуется для большей понятности текстов программ для программиста в имена переменных включать приставку, которая обозначает тип переменной. Тогда имена целочисленных переменных будут начинаться с приставок **byt**, **int** и **lng**, содержащих вещественные числа — **sng** и **dbl**, строковых — **chr** и **str**, логических — **bln**.

Объявление переменной. Важно, чтобы исполнитель программы (компьютер) «понимал», переменные какого типа используют-

ся в программе. При объявлении переменной используется ключевое слово и указывается её тип.

Область действия переменной. Область действия переменной, т. е. область в программе, где она доступна для использования, может быть локальной или глобальной.

Локальная переменная доступна только внутри процедуры или программного модуля, и к ней невозможно обращение из другой процедуры или модуля. Если переменная определена внутри процедуры, то она может быть вызвана только в этой процедуре, если она определена внутри программного модуля, то она может быть вызвана только в этом модуле.

К **глобальным переменным** может быть произведено обращение из всех программных модулей проекта.

Присваивание переменным значений. Переменная может получить или изменить значение с помощью **оператора присваивания.** В языках объектно-ориентированного программирования Visual Basic .NET и Visual C# в качестве оператора присваивания используется знак «=», а в языке Lazarus — знак «:=».

При выполнении оператора присваивания переменная, имя которой указано слева от знака равенства, получает значение, равное значению правой части. Это может быть, как в алгоритмических языках, выражение (арифметическое, строковое или логическое), а также значение свойства объекта или результат выполнения метода.

Вопросы и задания

1. В чём состоит разница между типом, именем и значением переменной?
2. Что происходит в оперативной памяти компьютера в процессе присваивания значения переменной?
3. Для чего нужно объявлять переменные в программе?

4.7. Графический интерфейс

Система объектно-ориентированного программирования позволяет визуализировать процесс создания графического интерфейса разрабатываемого проекта. Графический интерфейс необходим для реализации интерактивного диалога пользователя с исполняемым проектом.

Форма. Основой для создания графического интерфейса разрабатываемого проекта является **форма**, представляющая собой окно, в котором размещаются элементы управления. Необходимо отметить, что графический интерфейс проекта может включать в себя несколько форм.

Форма — это объект, представляющий собой окно на экране, в котором размещаются элементы управления.

Элементы управления. Визуальное конструирование графического интерфейса проекта состоит в том, что на форму с помощью мыши помещаются и «рисуются» те или иные **элементы управления**.

Перечислим классы элементов управления и их назначение в графическом интерфейсе проекта:

- текстовые поля `TextBox` (`Edit` в языке Lazarus), надписи `Label` и списки `ListBox` и `ComboBox` для ввода и вывода данных;
- графические поля `PictureBox` (`Image` в языке Lazarus) для вывода графики;
- командные кнопки `Button`, переключатели `RadioButton` и флажки `CheckBox` для организации интерактивного диалога пользователя с проектом;
- главное меню `MainMenu` для создания меню формы;
- панель инструментов `ToolBar` для создания панели инструментов формы;
- коллекция изображений `ImageList` для хранения изображений;
- диалоги `ColorDialog` и `FontDialog` для выбора цвета и шрифта;
- диалоги `OpenFileDialog` и `SaveFileDialog` для выбора файла при открытии и сохранении.

На форму может быть помещено несколько экземпляров одного класса элементов управления. Например, несколько кнопок, каждая из которых обладает индивидуальными значениями свойств (надпись, размеры и др.).

Элементы управления — это объекты, являющиеся элементами графического интерфейса проекта и реагирующие на события, производимые пользователем или другими программными объектами.

Форма и каждый класс элементов управления обладают определённым набором свойств, методов и событий. Однако есть

свойства, методы и события, которыми обладают формы и большинство элементов управления. С помощью свойств `Width` и `Height` устанавливается размер формы или элемента управления; с помощью метода `Show()` можно показать форму или элемент управления; на событие `Click` они реагируют.

Системы объектно-ориентированного программирования Visual Basic .NET и Visual C# имеют одинаковый набор элементов управления, который практически совпадает с набором элементов управления языка Lazarus (табл. 4.2). Это позволяет конструировать практически одинаковые графические интерфейсы проектов во всех вышеперечисленных системах объектно-ориентированного программирования.

Таблица 4.2

Некоторые элементы управления в языках Visual Basic .NET, Visual C# и Lazarus

Элемент управления	Visual Basic .NET	Visual C#	Lazarus
Текстовое поле	`TextBox1`	`textBox1`	`Edit1`
Надпись	`Label1`	`label1`	`Label1`
Список	`ListBox1`	`listBox1`	`ListBox1`
Поле со списком	`ComboBox1`	`comboBox1`	`ComboBox1`
Счётчик	`NumericUpDown1`	`numericUpDown1`	`SpinEdit1`
Ползунок	`TrackBar1`	`trackBar1`	`TrackBar1`
Переключатель	`RadioButton1`	`radioButton1`	`RadioButton1`

Автоматическая генерация кода элементов графического интерфейса. При размещении на форме элементов управления в системах объектно-ориентированного программирования производится автоматическая генерация программного кода. Устанавливаются значения свойств формы и элементов управления, которые определяют местоположение элемента графического интерфейса, его размеры, цвет, параметры шрифта и др.

В языках программирования Visual Basic .NET и Visual C# этот программный код сохраняется в файле программного кода

формы в разделе *Код, автоматически созданный конструктором форм Windows*. В языке Lazarus этот программный код сохраняется в отдельном файле project1.lpr, связанном с формой.

Вопросы и задания

1. Какие элементы управления целесообразно использовать при конструировании графического интерфейса проекта, если необходимо: вводить и выводить данные? Организовать диалог с пользователем?
2. Что происходит в системе программирования после помещения на форму элемента управления?

Практическая работа 4.2

Создание проекта «Переменные»

Задание. Создать в системе объектно-ориентированного программирования Visual Studio или Lazarus проект, который позволит продемонстрировать использование различных типов переменных, арифметических, строковых и логических выражений и операции присваивания. Каждый тип переменных разместить на отдельной вкладке.

Электронное приложение к главе 4.

Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Lazarus

Для создания вкладок разместить на форме элемент управления `TabControl1` (`PageControl1` в Lazarus).

Создание вкладок на языках Visual Basic .NET, и Visual C#

1. Выделить элемент управления `TabControl1` и в окне *Свойства* у свойства *TabPages* активизировать значение (*Коллекция*).
2. В появившемся окне *Редактор коллекции TabPage* (рис. 4.21) создать три вкладки на панели инструментов, нажав три раза кнопку *Добавить*.
   В списке *Свойства TabPage:* в свойстве *Text* ввести названия вкладок: *Числовые*, *Строковые*, *Логические*.

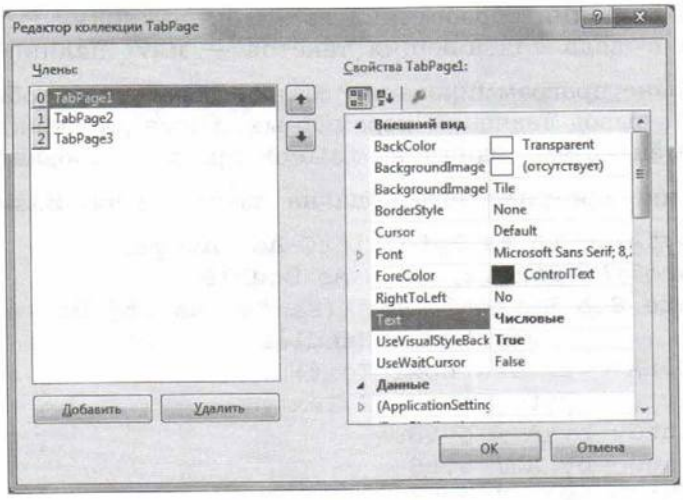


Рис. 4.21

Создание вкладок на языке Lazarus

1. Выделить элемент управления `PageControl1`, щёлкнуть правой кнопкой мыши и в контекстном меню выбрать пункт *Добавить страницу*. Выполнить эти действия три раза.
2. В окне *Инспектор объектов* в свойстве *Caption* ввести названия вкладок: *Числовые*, *Строковые*, *Логические*.

Настройка графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Lazarus

3. На вкладке *TabPage1* (*TabSheet1* в Lazarus) разместить:
   - для ввода чисел — два текстовых поля `TextBox1` (`Edit1` в Lazarus) и `TextBox2` (`Edit2` в Lazarus);
   - для вывода чисел — надписи `Label1`, `Label2` и `Label3`;
   - для создания обработчика события — кнопку `Button1`;
   - для вывода поясняющих текстов — пять надписей.
4. На вкладке *TabPage2* (*TabSheet2* в Lazarus) разместить:
   - для ввода символа и строки — два текстовых поля `TextBox3` (`Edit3` в Lazarus) и `TextBox4` (`Edit4` в Lazarus);
   - для вывода строки — надпись `Label9`;
   - для создания обработчика события — кнопку `Button2`;
   - для вывода поясняющих текстов — три надписи.
5. На вкладке *TabPage3* (*TabSheet3* в Lazarus) разместить:
   - для вывода логического значения — надпись `Label13`;

- для создания обработчика события — кнопку Button3;
- для вывода поясняющих текстов — пять надписей.

На языке программирования создать обработчик события, реализующий вывод значений переменных. Ниже для каждой вкладки приведён код на одном из языков программирования.

Создание программного кода на языке Visual Basic .NET

```
Dim bytA, bytB As Byte, intC As Integer,
    sngD As Single, dblE As Double
Private Sub Button1_Click(sender As Object, e As
            EventArgs) Handles Button1.Click
    bytA = Val(TextBox1.Text)
    bytB = CByte(TextBox2.Text)
    intC = bytA / bytB
    sngD = bytA / bytB
    dblE = bytA / bytB
    Label1.Text = intC
    Label2.Text = sngD
    Label3.Text = dblE
  End Sub
```

Создание программного кода на языке программирования Visual C#

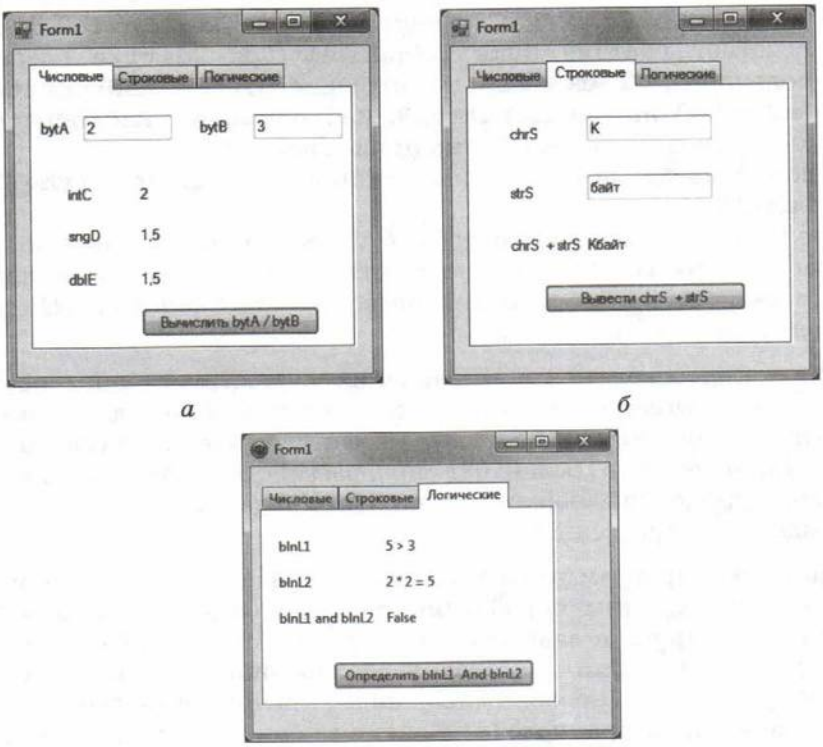
```
char chrS;
String strS;
private void button2_Click(object sender,
                           EventArgs e)
{chrS = Convert.ToChar(textBox3.Text);
  strS = textBox4.Text;
  label9.Text = chrS + strS;
}
```

Создание программного кода на языке Lazarus

```
var
blnL1: boolean;
blnL2: boolean;
blnL3: boolean;
procedure TForm1.Button3Click(Sender: TObject);
begin
  blnL1 := 5 > 3;
  blnL2 := 2 * 2 = 5;
  blnL3 := blnL1 and blnL2;
  Label13.Caption := BoolToStr(blnL3,True);
end;
```

Запуск проекта на языках Visual Basic .NET, Visual C# и Lazarus

1. Ввести команду [*Отладка—Начать*] (на языке Lazarus [*Запуск—Запустить*]) или щёлкнуть по кнопке ▶ на панели инструментов окна интегрированной среды разработки.
2. На вкладке *Числовые* в текстовые поля ввести числа и щёлкнуть по кнопке *Вычислить bytA/bytB*. На надписи будут выведены значения числовых переменных различных типов (рис. 4.22, *а*).
   На вкладке *Строковые* в текстовые поля ввести символ и строку и щёлкнуть по кнопке *Вывести chrS + strS*. На надпись будет выведена строка (рис. 4.22, *б*).
   На вкладке *Логические* щёлкнуть по кнопке *Определить lblL1 And lblL2*. На надпись будет выведено значение логического выражения (рис. 4.22, *в*).



а *б* *в*

Рис. 4.22

Практическая работа 4.3

Создание проекта «Отметка»

Задание. Создать в системе объектно-ориентированного программирования Visual Studio или Lazarus проект, который в зависимости от количества ошибок, введённых с использованием различных элементов управления (списка, поля со списком, текстового поля, счётчика, ползунка и переключателей), выводит на надпись отметку.

Электронное приложение к главе 4.

Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Lazarus

1. Разместить на форме (рис. 4.23):
   - для ввода количества ошибок — список **ListBox1**, поле со списком **ComboBox1**, текстовое поле **TextBox1** (**Edit1** в Lazarus), счётчик **NumericUpDown1** (**SpinEdit1** в Lazarus), ползунок **TrackBar1** и переключатели **RadioButton1**, **RadioButton2**, **RadioButton3**, **RadioButton4**, **RadioButton5**;
   - для вывода отметки — надпись **Label1**;
   - для вывода поясняющих текстов — надписи **Label2** и **Label3**.
2. Сделать графический интерфейс более привлекательным. Для этого присвоить форме и элементам управления новые значения свойств с помощью диалогового окна *Свойства* (*Инспектор объектов* в Lazarus).

Для каждого элемента управления существует одно событие (событие по умолчанию), чаще всего возникающее для данного элемента управления (табл. 4.3). После двойного щелчка по элементу управления в редакторе программного кода открывается заготовка процедуры обработки такого события.

3. Создать программный код.

На языке программирования Visual Basic .NET объявим переменную и создадим обработчик события, реализующий вывод отметки на надпись в зависимости от количества ошибок, выбранных в списке. Создадим заготовку обработчика события щелчком по списку и введём программный код. Для преобразования количества ошибок, выбранного как элемент списка строкового типа, в целое число используем функцию **CByte()**. Для реализации выбора отметки используем оператор **Select Case**.

Таблица 4.3

События по умолчанию некоторых элементов управления в языках Visual Basic .NET, Visual C# и Lazarus

Элемент управления	Visual Basic .NET	Visual C#	Lazarus
Текстовое поле	TextChanged	TextChanged	Change
Надпись	Click	Click	Click
Список	SelectedIndexChanged	SelectedIndexChanged	Click
Поле со списком	SelectedIndexChanged	SelectedIndexChanged	Change
Счётчик	ValueChanged	ValueChanged	Click
Ползунок	Scroll	Scroll	Change
Переключатель	CheckedChanged	CheckedChanged	Click

Создание программного кода на языке программирования Visual Basic .NET

```
Dim N As Byte
Private Sub ListBox1_SelectedIndexChanged
            (sender As Object, e As EventArgs)
Handles ListBox1.SelectedIndexChanged
N = CByte(ListBox1.Text)
Select Case N
   Case 0
   Label1.Text = "Отлично"
   Case 1
   Label1.Text = "Хорошо"
   Case 2
   Label1.Text = "Удовлетворительно"
   Case 3
   Label1.Text = "Плохо"
   Case Else
   Label1.Text = "Очень плохо"
End Select
End Sub
```

На языке программирования Lazarus объявим переменную и создадим обработчик события, реализующий вывод отметки на надпись в зависимости от количества ошибок, введённых в текстовое поле. Для преобразования введённого в текстовое поле количества ошибок строкового типа в целое число используем функцию **StrToInt()**. Для реализации выбора отметки используем оператор **Case**.

Создание программного кода на языке программирования Lazarus

```
var
N: byte;
procedure TForm1.Edit1Change(Sender: TObject);
begin
N := StrToInt(Edit1.Text);
Case N Of
  0: Label1.Caption := 'Отлично';
  1: Label1.Caption := 'Хорошо';
  2: Label1.Caption := 'Удовлетворительно';
  3: Label1.Caption := 'Плохо';
  Else Label1.Caption := 'Очень плохо';
end;
end;
```

На языке программирования Visual C# объявим переменную и создадим обработчик события, реализующий вывод отметки на надпись в зависимости от количества ошибок, введённых с помощью ползунка. Так как свойство ползунка **Value** имеет тип **int**, объявим переменную такого же типа. Для реализации выбора отметки используем оператор **switch**.

Создание программного кода на языке программирования Visual C#

```
int n;
private void trackBar1_Scroll(object sender,
                              EventArgs e)
{n = trackBar1.Value;
  switch (n)
  {case 0:
  label1.Text = "Отлично";
  break;
  case 1:
  label1.Text = "Хорошо";
  break;
  case 2:
  label1.Text = "Удовлетворительно";
  break;
  case 3:
  label1.Text = "Плохо";
  break;
  default:
  label1.Text = "Очень плохо";
  break;
  }
}
```

Запуск проекта на языках Visual Basic .NET, Visual C# и Lazarus

Загрузка проекта в систему объектно-ориентированного программирования производится путём активизации в папке проекта основного файла проекта:

- язык Visual Basic .NET — файл с расширением vbproj;
- язык Visual C# — файл с расширением csproj;
- язык Lazarus — файл с расширением lpr.

Запустим проект, после этого система программирования перейдёт в режим выполнения проекта [*Отладка*], в котором редактирование графического интерфейса или программного кода невозможно.

1. Ввести команду [*Отладка—Начать отладку*] ([*Запуск—Запустить*] на языке Lazarus) или щёлкнуть по кнопке ▶ на панели инструментов окна интегрированной среды разработки.
2. Ввести количество ошибок (например, 2) с использованием элементов управления:
   - списка;
   - поля со списком;
   - текстового поля;
   - счётчика;
   - ползунка;
   - одного из переключателей.

   На надпись будет выведена отметка (в данном случае «Удовлетворительно») (см. рис. 4.23).

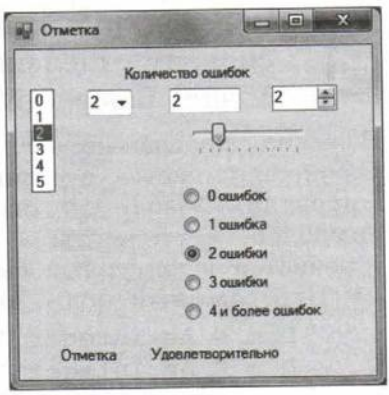


Рис. 4.23

Практическая работа 4.4

Создание проекта «Перевод целых чисел»

Задание. Создать проект, реализующий перевод целых десятичных чисел в двоичную систему счисления.

Электронное приложение к главе 4.

Создание графического интерфейса проекта на языках Visual Basic .NET, Visual C# и Lazarus

1. Разместить на форме (рис. 4.24):
   - текстовое поле TextBox1 (Edit1 в Lazarus) для ввода целого числа в десятичной системе счисления;
   - надпись Label1 для вывода целого двоичного числа;
   - кнопку Button1 для создания обработчика события, реализующего перевод целых чисел в двоичную систему счисления;
   - надписи Label2 и Label3 для вывода поясняющей информации.

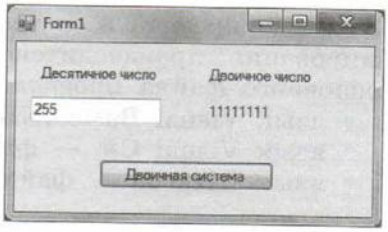


Рис. 4.24

2. Сделать графический интерфейс более привлекательным. Присвоить элементам управления новые значения свойств с помощью диалогового окна *Свойства*.

Создание обработчика события, реализующего перевод целых десятичных чисел в двоичную систему счисления, на языке программирования Visual Basic .NET

В языке программирования Visual Basic .NET комментарий в строке начинается с символа апострофа «'». В языке программирования Visual C# строки комментария начинаются с двух символов косой черты «//»; в языке Lazarus комментарий заключается в фигурные скобки «{}».

1. Программный код с комментариями:

```
Dim N As Integer 'десятичное число
Dim R As Integer 'остаток от деления исходного
                 'целого десятичного числа или
                 'целого частного на основание
                 'новой системы
Dim Bin As String 'двоичное число в строковой форме
Private Sub Button1_Click(sender As Object,
e As EventArgs) Handles Button1.Click
  '1.Ввести десятичное целое число и другие
  'начальные значения
     N = Val(TextBox1.Text)
     Label1.Text = ""
```

```
        Bin = ""
'2.В цикле с предусловием, пока исходное целое
'десятичное число или целое частное больше 0,
'выполнить вычисления:
        Do While N > 0
'2.1.Вычислить остаток от деления исходного целого
'десятичного числа или целого частного на основание
'новой системы (на 2).
          R = N Mod 2
'2.2.Выполнить целочисленное деление целого
'десятичного числа или целого частного на основание
'новой системы (на 2).
          N = N \ 2
'2.3. Записать полученный остаток от деления слева
'от двоичного числа (остатки, записанные в обратном
'порядке, образуют двоичное число).
          Bin = CStr(R) + Bin
        Loop
'3.Вывести двоичное целое число
        Label1.Text = Bin
        End Sub
```


Создание обработчика события, реализующего перевод целых десятичных чисел в двоичную систему счисления, на языке программирования Visual C#

Программный код:

```
private void button1_Click(object sender, EventArgs e)
        {
            int N;
            int R;
            string Oct;
                N = Convert.ToInt32(textBox1.Text);
                label1.Text = "";
                Oct = "";
                while (N > 0)
                    {R = N % 2;
                     N = Convert.ToInt32(N/2);
                     Oct = Convert.ToString(R)+ Oct;
                    }
                label1.Text = Oct;
        }
```

Создание обработчика события, реализующего перевод целых десятичных чисел в двоичную систему счисления, на языке программирования Lazarus

Программный код:

```
procedure TForm1.Button1Click(Sender: TObject);
var N, R: integer;
       Bin: string;
begin
     N := StrToInt(Edit1.Text);
     Label1.Caption := '';
     Bin := '';
     while N > 0 Do
         begin
             R := N mod 2;
             N := N div 2;
             Bin := FloatToStr(R) + Bin;
         end;
     Label1.Caption := Bin;
end;
```

Запуск проекта на языках Visual Basic .NET, Visual C# и Lazarus

Запустить проект на выполнение и ввести в текстовое поле десятичное число (например, 255).

Щёлкнуть по кнопке — на надписи будет выведено двоичное число (см. рис. 4.18).

ЭОР к главе 4 на сайте ФЦИОР (http://fcior.edu.ru)

- Алгоритмы. Ветвление и выбор
- Блок-схемы. Циклический алгоритм
- Идентификаторы, ключевые слова, литералы
- Общий обзор .NET и синтаксиса C#
- Понятие алгоритма, его свойства. Линейный алгоритм
- Теория алгоритмов. Основные понятия

ОГЛАВЛЕНИЕ