



11



Н. Д. Угринович

ИНФОРМАТИКА

БАЗОВЫЙ УРОВЕНЬ



ИЗДАТЕЛЬСТВО

БИНОМ

Н. Д. Угринович

ИНФОРМАТИКА

11 класс

Базовый уровень



Москва
БИНОМ. Лаборатория знаний
2017

УДК 004.9
ББК 32.97
У27

Угринович Н. Д.
У27 Информатика. 11 класс. Базовый уровень / Н. Д. Угринович. — М. : БИНОМ. Лаборатория знаний, 2017. — 272 с. : ил.

ISBN 978-5-9963-3145-1

Учебное издание ориентировано на преподавание информатики на базовом уровне в 11 классе общеобразовательных организаций. Рассматриваются такие темы, как автоматизация информационных процессов, моделирование и формализация, базы данных и СУБД, социальная информатика. 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 10 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

РЕКОМЕНДАЦИИ ПО ИСПОЛЬЗОВАНИЮ УЧЕБНИКА

Учебник обеспечивает изучение курса «Информатика» в 11 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»;
- методическое пособие для учителей.

Учебник содержит практикум.

Компьютерный практикум может проводиться в операционных системах Windows  и Linux .

Файлы для выполнения практических работ предложены 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-ресурс;
-  — проектное или исследовательское задание (по выбору);
-  — практическая работа на компьютере;
-  — межпредметные связи;
-  — групповая работа.

Глава 1

КОМПЬЮТЕР КАК СРЕДСТВО АВТОМАТИЗАЦИИ ИНФОРМАЦИОННЫХ ПРОЦЕССОВ

При изучении данной главы рекомендуется установить следующее программное обеспечение:

для операционной системы Windows:

- антивирус Kaspersky AntiVirus;
- системный анализатор SiSoftware Sandra;

для операционных систем Windows
и Linux:

- браузеры Mozilla FireFox и Opera;
- свободный звуковой редактор Audacity;

для операционной системы Linux:

- антивирус KlamAV;
- просмотр и управление разделами жёсткого диска KDiskFree,
центр управления KDE, управление пакетами Synaptic.



1.1. История развития вычислительной техники

Вычисления в доэлектронную эпоху. Потребность счёта предметов у человека возникла ещё в доисторические времена. Древнейший метод счёта заключался в сопоставлении предметов некоторой группы (например, животных) с предметами другой группы, играющей роль счётного эталона. У большинства народов первым таким эталоном были пальцы (счёт на пальцах).

Расширяющиеся потребности в счёте заставили людей употреблять другие счётные эталоны (зарубки на палочке, узлы на верёвке и т. д.).

Каждый школьник хорошо знаком со счётными палочками, которые использовались в качестве счётного эталона в первом классе.

В Древнем мире при счёте больших количеств предметов для обозначения определённого их количества (у большинства народов — десяти) стали применять новый знак, например зарубку на другой палочке. Первым вычислительным устройством, в котором применялся этот метод, стал **абак**.

Древнегреческий абак представлял собой посыпанную морским песком дощечку. На песке проводились бороздки, на которых камешками обозначались числа. Одна бороздка соответствовала единицам, другая — десяткам и т. д. Если в какой-то бороздке при счёте набиралось более 10 камешков, их снимали и добавляли один камешек в следующий разряд. Римляне усовершенствовали абак, перейдя от песка и камешков к мраморным доскам с выточенными желобками и мраморными шариками (рис. 1.1).

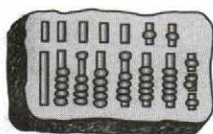


Рис. 1.1

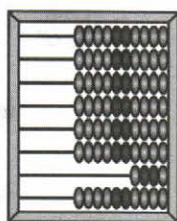


Рис. 1.2



Рис. 1.3

По мере усложнения хозяйственной деятельности и социальных отношений (денежных расчётов, задач измерений расстояний, времени, площадей и т. д.) возникла потребность в арифметических вычислениях. Для выполнения простейших арифметических операций (сложения и вычитания) стали использовать абак, а по прошествии веков — **счёты** (рис. 1.2).

Развитие науки и техники требовало проведения всё более сложных математических расчётов, и были изобретены механические счётные машины — **арифмометры** (рис. 1.3). Арифмометры могли не только складывать, вычитать, умножать и делить числа, но и запоминать промежуточные результаты, печатать результаты вычислений и т. д.

Первый арифмометр создал Готфрид Вильгельм Лейбниц в 1672 г.

В середине XIX века английский математик Чарльз Бэббидж выдвинул идею создания программно управляемой счётной машины, имеющей арифметическое устройство, устройство управления, а также устройства ввода и печати.

Аналитическую машину Бэббиджа (прообраз современных компьютеров) по сохранившимся описаниям и чертежам построили энтузиасты из Лондонского музея науки (рис. 1.4). Аналитическая машина состоит из четырёх тысяч стальных деталей и весит три тонны.

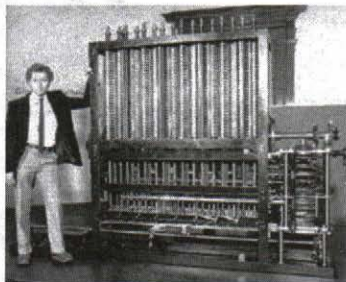


Рис. 1.4

Вычисления производились Аналитической машиной в соответствии с инструкциями (программами), которые разработала леди Ада Лавлейс (дочь английского поэта Джорджа Байрона). Графиню Лавлейс считают первым программистом, и в её честь назван язык программирования Ада.

Первыми носителями информации, которые использовались для хранения программ, были **перфокарты** (рис. 1.5).

Программы записывались на перфокарты путём пробития в определённом

порядке отверстий в плотных бумажных карточках. Затем перфокарты помещались в Аналитическую машину, которая считывала расположение отверстий и выполняла вычислительные операции в соответствии с заданной программой.



Рис. 1.5

1.1.1. Развитие электронной вычислительной техники

ЭВМ первого поколения. В 40-е годы XX века начались работы по созданию первых электронных вычислительных машин, в которых на смену механическим деталям пришли **электронные лампы** (см. таблицу в конце параграфа). ЭВМ первого поколения требовали для своего размещения больших залов, так как в них использовались десятки тысяч электронных ламп. Такие ЭВМ создавались в единичных экземплярах, стоили очень дорого и устанавливались в крупнейших научно-исследовательских центрах.

В 1945 году в США был построен ENIAC (Electronic Numerical Integrator and Computer — электронный числовой интегратор и вычислитель), а в 1950 году в СССР была создана МЭСМ (Малая Электронная Счётная Машина) — рис. 1.6.



Рис. 1.6

ЭВМ первого поколения могли выполнять вычисления со скоростью несколько тысяч операций в секунду, последовательность выполнения которых задавалась программами. Программы писались на машинном языке, алфавит которого состоял из двух знаков: 1 и 0.

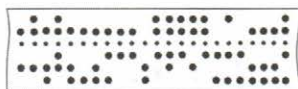


Рис. 1.7

Программы вводились в ЭВМ с помощью перфокарт или перфолент (рис. 1.7), причём наличие отверстия на перфокарте соответствовало знаку 1, а его отсутствие — знаку 0.

Результаты вычислений выводились с помощью печатающих устройств в форме длинных последовательностей нулей и единиц. Писать программы на машинном языке и расшифровывать результаты вычислений могли только высококвалифицированные программисты, понимавшие язык первых ЭВМ.

ЭВМ второго поколения. В начале 60-х годов XX века были созданы ЭВМ второго поколения, основанные на новой элементной базе — **транзисторах** (см. таблицу в конце параграфа), которые имеют в десятки и сотни раз меньшие размеры и массу, более высокую надежность и потребляют значительно меньшую электрическую мощность, чем электронные лампы. Такие ЭВМ производились малыми сериями и устанавливались в крупных научно-исследовательских центрах и ведущих высших учебных заведениях.

В СССР в 1967 году вступила в строй наиболее мощная в Европе ЭВМ второго поколения БЭСМ-6 (Большая Электронная Счётная Машина, рис. 1.8), которая могла выполнять 1 миллион операций в секунду.



Рис. 1.8

В БЭСМ-6 использовалось 60 тысяч транзисторов и 180 тысяч диодов, устройства внешней памяти на магнитных лентах для хранения программ и данных, а также алфавитно-цифровые печатающие устройства для вывода результатов вычислений.

Работа программистов по разработке программ существенно упростилась, так как стала проводиться с использованием языков программирования высокого уровня (Алгол, Фортран и др.).

ЭВМ третьего поколения. В 70-е годы прошлого века в качестве элементной базы стали использовать **интегральные схемы** (см. таблицу в конце параграфа). В интегральной схеме (маленькой полупроводниковой пластине) были плотно упакованы тысячи транзисторов, каждый из которых имеет размеры, сравнимые с толщиной человеческого волоса.

ЭВМ на базе интегральных схем стали гораздо более компактными, быстродействующими и дешёвыми. Такие мини-ЭВМ (рис. 1.9) производились большими сериями и были доступными для большинства научных институтов и высших учебных заведений.

ЭВМ четвёртого поколения имеют две линии развития: персональные компьютеры и суперкомпьютеры.

Персональные компьютеры. Развитие высоких технологий привело к созданию больших интегральных схем — БИС (см. таблицу в конце параграфа), включающих десятки тысяч транзисторов. Это позволило приступить к выпуску компактных персональных компьютеров, доступных для массового пользователя.



Рис. 1.9



Рис. 1.10

Первым персональным компьютером был Apple II (рис. 1.10) («дедушка» современных компьютеров Macintosh), созданный в 1977 году. В 1982 году фирма IBM приступила к изготовлению персональных компьютеров IBM PC («дедушек» современных IBM-совместимых компьютеров).

Современные персональные компьютеры компактны и обладают в тысячи раз большим быстродействием по сравнению с первыми персональными компьютерами (могут выполнять несколько миллиардов операций в секунду). Они доступны по цене для массового потребителя.

Персональные компьютеры могут быть различного конструктивного исполнения: настольные и портативные (рис. 1.11).



Рис. 1.11

Современные суперкомпьютеры. Это многопроцессорные комплексы (рис. 1.12), которые позволяют добиться очень высокой производительности и могут применяться для расчётов в реальном времени в метеорологии, военном деле, науке и т. д.



Рис. 1.12

Вопросы и задания



Используя текст параграфа и таблицу, ответьте на вопросы.

- Почему современные персональные компьютеры в сотни раз меньше, но при этом в сотни тысяч раз быстрее ЭВМ первого поколения?
- Почему современные персональные компьютеры доступны массовому потребителю?

Характеристика	Поколения			
	Первое	Второе	Третье	Четвёртое
Годы использования	40–50 гг. XX в.	50-е гг. XX в.	60-е гг. XX в.	70-е гг. XX в. – настоящее время
Основной элемент	 Электронная лампа	 Транзистор	 Интегральная схема	 Сверх-большая интегральная схема — СВИС (процессор)
Быстродействие (операций в секунду)	Десятки тысяч	Сотни тысяч	Миллионы	Миллиарды
Количество ЭВМ в мире, шт.	Сотни	Тысячи	Сотни тысяч	Около двух миллиардов

Практическая работа 1.1



Виртуальные компьютерные музеи

Задание. Посетить в Интернете виртуальные компьютерные музеи и ознакомиться с историей развития отечественной и зарубежной вычислительной техники.

Варианты выполнения работы:

- в операционной системе Windows или в операционной системе Linux;
- просмотр веб-страниц виртуальных компьютерных музеев в различных браузерах (Mozilla Firefox, Internet Explorer, Chrome, Opera или др.).



Посещение виртуальных компьютерных музеев



Посетим в Интернете Виртуальный музей информатики школы № 444 Москвы.

1. Запустить браузер в операционной системе Windows или в операционной системе Linux и в поле *Адрес:* ввести **<http://informat444.narod.ru/museum/>** (доменное имя и путь к начальной странице Виртуального музея информатики школы № 444 Москвы).
2. На появившейся начальной странице музея активизировать ссылки (рис. 1.13).

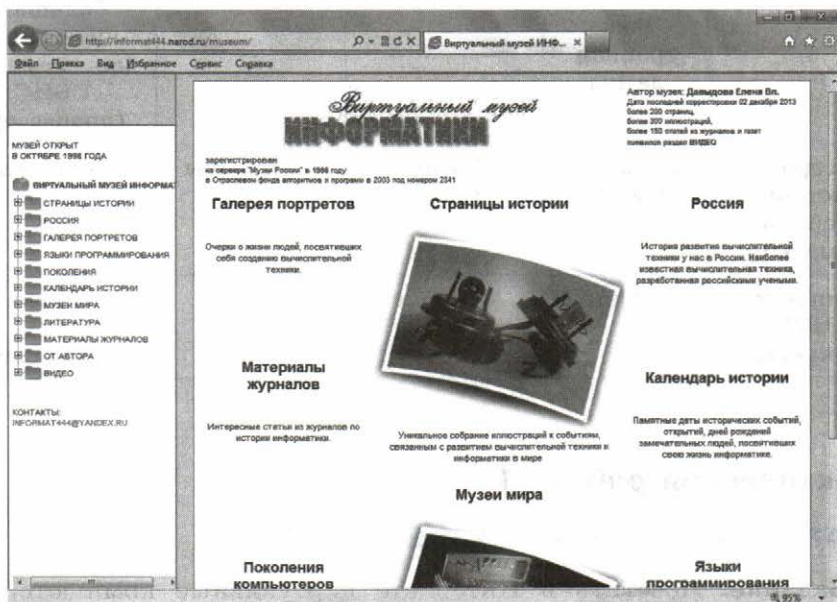


Рис. 1.13

Посетим в Интернете Виртуальный компьютерный музей Эдуарда Пройдакова.

1. Запустить браузер (например, SeaMonkey) в операционной системе Windows или в операционной системе Linux и в поле *Адрес:* ввести **http://www.computer-museum.ru** (доменное имя Виртуального компьютерного музея Эдуарда Пройдакова).
2. На появившейся начальной странице музея активизировать ссылки (рис. 1.14).

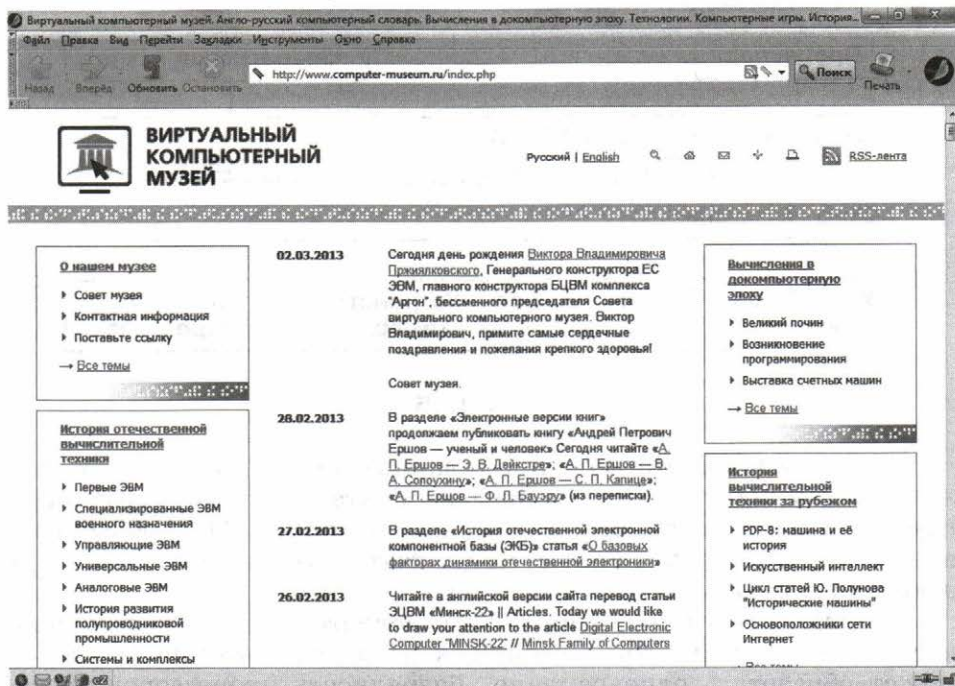


Рис. 1.14

1.2. Архитектура персонального компьютера

В основу архитектуры современных персональных компьютеров положен магистрально-модульный принцип. Этот принцип предусматривает построение компьютера из функциональных блоков, взаимодействующих посредством общего канала (каналов) —

шины. В сочетании с открытой (общеизвестной) архитектурой это позволяет потребителю собирать машину нужной конфигурации.

Магистраль включает в себя **три многоразрядные шины**: шину данных, шину адреса и шину управления, которые представляют собой многопроводные линии (рис. 1.15). К магистрالي подключаются процессор и оперативная память, а также периферийные устройства ввода, вывода и хранения информации, которые обмениваются информацией в форме последовательностей нулей и единиц, реализованных в виде электрических импульсов.

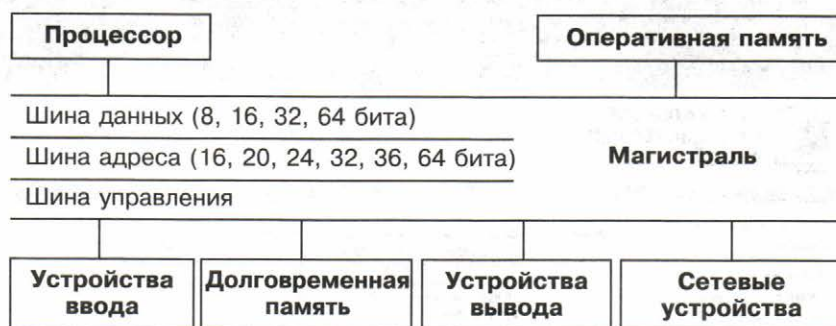


Рис. 1.15

Шина данных. По этой шине данные передаются между различными устройствами. Например, считанные из оперативной памяти данные могут быть переданы процессору для обработки, а затем полученные данные могут быть отправлены обратно в оперативную память для хранения.

Разрядность шины данных определяется разрядностью процессора, т. е. количеством двоичных разрядов, которые процессор может обработать одновременно. Разрядность процессоров постоянно увеличивалась по мере развития компьютерной техники и в настоящее время составляет 64 бита.

Шина адреса. Выбор устройства или ячейки памяти, куда пересылаются или откуда считываются данные по шине данных, производит процессор. Каждое устройство или ячейка оперативной памяти имеет свой адрес. Адрес передаётся по адресной шине, причём сигналы по ней передаются в одном направлении: от процессора к оперативной памяти и устройствам (однонаправленная шина).

Разрядность шины адреса определяет объём адресуемой памяти, т. е. количество ячеек оперативной памяти, которые могут

иметь уникальные адреса. Количество адресуемых ячеек памяти можно рассчитать по формуле:

$$N = 2^I, \text{ где } I — \text{разрядность шины адреса.}$$

Разрядность шины адреса постоянно увеличивалась, но она не обязательно совпадает с заявленной разрядностью процессора. Например, четырёхъядерный процессор Intel Core i7-6700K Skylake (август 2015 года), заявленный как 64-разрядный, на самом деле обеспечивает максимальный объём памяти, равный только 64 Гбайт (т. е. количество адресуемых байтовых ячеек памяти, равно 2^{36} , а не 2^{64}). Таким образом, его шина адреса имеет разрядность 36 бит. Большая разрядность шины адреса усложняет конструкцию процессора, а обеспечиваемый при этом больший объём памяти в современных компьютерах не востребован.

Шина управления. По шине управления передаются сигналы, определяющие характер обмена информацией по магистрали. Сигналы управления определяют, какую операцию — считывание или запись информации из памяти — нужно производить, синхронизируют обмен информацией между устройствами и т. д.

Системная плата. Важнейшим аппаратным компонентом компьютера является системная плата. На рис. 1.16, 1.17 представлены фотография и логическая схема системной платы. На системной плате реализована магистраль обмена информацией, имеются разъёмы для установки процессора, слоты для установки оперативной памяти, а также контроллеров внешних устройств.

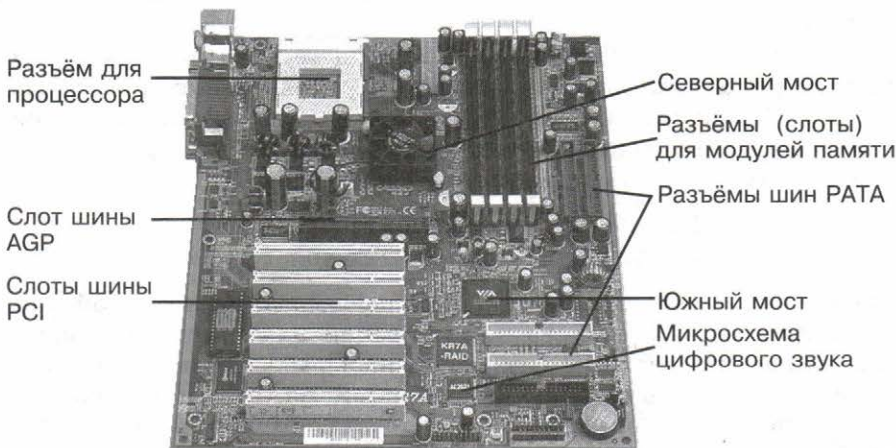


Рис. 1.16



Рис. 1.17

Пропускная способность. Быстродействие устройства зависит от тактовой частоты тактового генератора (обычно измеряется в мегагерцах — МГц) и разрядности, т. е. количества бит данных, которые устройство может обрабатывать или передавать одновременно (измеряется в битах). Дополнительно в устройствах используется внутреннее умножение частоты с разными коэффициентами.

Соответственно, скорость передачи данных (пропускная способность) соединяющих эти устройства шин также должна различаться. Пропускная способность шины данных (измеряется в бит/с) равна произведению разрядности шины (измеряется в битах) и частоты шины (измеряется в Гц = 1/с):

$$\text{Пропускная способность шины} = \text{Разрядность шины} \times \text{Частота шины}.$$

Северный и Южный мосты. Для согласования тактовой частоты и разрядности устройств на системной плате устанавливаются специальные микросхемы (их набор называется чипсетом), включающие в себя контроллер оперативной памяти и видеопамати (так называемый Северный мост) и контроллер периферийных устройств (Южный мост).

Северный мост обеспечивает обмен данными с процессором, оперативной памятью и видеопаматю. В предыдущих моделях

процессоров этот обмен данными обеспечивался при помощи системной шины FSB (*FrontSide Bus*). Частота шины FSB была меньше, чем частота процессора, в некоторое число раз, называемое множителем. Например, если частота шины FSB равна 400 МГц, а множитель равен 14, то рабочая частота процессора составляет $400 \text{ МГц} \cdot 14 \approx 5,5 \text{ ГГц}$. При этом шина FSB имела определённую разрядность и могла передавать несколько порций данных за один рабочий цикл, например системная шина в процессорах Intel Xeon при частоте 400 МГц обеспечивала пропускную способность до 12 800 Мбайт, пропуская за один такт четыре 8-разрядные порции данных (технология *Quad Pumped Bus*).

В современных процессорах вместо FSB применяется другой тип системной шины. Так, в процессорах фирмы AMD используется шина HyperTransport, а в процессорах фирмы Intel — шина QPI (*QuickPath Interconnect*). Конструкции и принципы работы этих шин иные, чем у шины FSB, а скорость передачи данных существенно выше. Например, шина QPI является двусторонней и может обеспечивать передачу данных в обоих направлениях (от процессора и к процессору) с пропускной способностью до 25,6 Гбайт/с в одном соединении, и при этом процессор может иметь несколько соединений по шине QPI.

Шина памяти. Обмен данными между процессором и оперативной памятью производится по шине памяти, частота которой может быть меньше, чем частота шины процессора. Например, частота шины памяти может составлять 533 МГц, т. е. оперативная память получает данные примерно в два раза реже, чем процессор. Так как разрядность шины памяти равна разрядности процессора и составляет 64 бита, пропускная способность шины памяти равна:

$$64 \text{ бита} \cdot 533 \text{ МГц} = 34\,112 \text{ Мбит/с} \approx \\ \approx 34 \text{ Гбит/с} \approx 4 \text{ Гбайт/с}.$$

Шины AGP и PCI Express. По мере усложнения графики приложений требования к быстродействию шины, связывающей видеопамять с процессором и оперативной памятью, возрастают. Для подключения видеоплаты к Северному мосту может использоваться 32-битовая шина AGP (англ. *Accelerated Graphic Port* — ускоренный графический порт). Эта шина работала с частотой 66 МГц и первоначально передавала по одной порции данных за один такт. Позже появились новые версии шины: AGP-4 и AGP-8, которые, соответственно, при той же частоте могли передавать за один такт 4 или 8 порций данных. Можно определить, что, например, пропускная способность шины AGP-8 составляла:

$$32 \text{ бита} \cdot 66 \text{ МГц} \cdot 8 \approx 16\,896 \text{ Мбит/с} \approx 16,9 \text{ Гбит/с} \approx 2 \text{ Гбайт/с}.$$



В настоящее время для подключения видеоплаты к Северному мосту всё большее распространение получает шина PCI Express (англ. *Peripheral Component Interconnect bus Express* — ускоренная шина взаимодействия периферийных устройств). Пропускная способность этой шины значительно выше пропускной способности AGP.

К видеоплате с помощью аналогового разъёма VGA (англ. *Video Graphics Array* — графический видеоадаптер) или цифрового разъёма DVI (англ. *Digital Visual Interface* — цифровой видеоинтерфейс) подключается электронно-лучевой или жидкокристаллический монитор либо проектор.

Шина PCI. К Северному мосту подключается по специальной шине Южный мост, к которому, в свою очередь, подключаются периферийные устройства. Шина PCI (англ. *Peripheral Component Interconnect bus* — шина взаимодействия периферийных устройств) обеспечивает обмен информацией с контроллерами периферийных устройств, которые устанавливаются в слоты расширения системной платы.

Наиболее часто эта шина используется для установки устройств доступа к локальной сети и глобальной сети Интернет (сетевая карта), и беспроводной сети (сетевой адаптер Wi-Fi, произносится «вай-фай», сокр. от англ. *Wireless Fidelity* — протокол и стандарт на оборудование для широкополосной радиосвязи).

Разрядность шины PCI может составлять 32 бита или 64 бита, а частота — 33 МГц или 66 МГц. Таким образом, максимальная пропускная способность шины PCI составляет:

$$64 \text{ бита} \cdot 66 \text{ МГц} \approx 4224 \text{ Мбит/с} \approx 528 \text{ Мбайт/с.}$$

Шина IEEE 1394 (другие названия FireWire, i-Link). Это последовательная высокоскоростная шина, предназначенная для обмена цифровой информацией между компьютером и цифровыми устройствами (цифровыми видеокамерами, DVD-плеерами и др.) без потери качества изображения и звука. (Данную функцию может выполнять также контроллер IEEE 1394, который подключается к шине PCI.) Скорость передачи данных по этой шине может достигать 200 Мбайт/с и более.

Шина ATA. Устройства внешней памяти (жёсткие диски, CD- и DVD-дисководы) подключаются к Южному мосту по шине ATA (англ. *Advanced Technology Attachment* — шина подключения накопителей). Ранее использовалась параллельная шина PATA (англ. *Parallel ATA*), скорость передачи данных по которой может достигать 133 Мбайт/с. С 2003 года широкое распространение получила последовательная шина SATA (англ. *Serial*

ATA), скорость передачи данных по которой может достигать 600 Мбайт/с.

Шина USB. Для подключения принтеров, сканеров, цифровых камер и других периферийных устройств обычно используется шина USB (англ. *Universal Serial Bus* — универсальная последовательная шина). Шина USB прежнего стандарта 2.0 обеспечивала пропускную способность до 60 Мбайт/с. Новый стандарт USB 3.0, появившийся в 2008 году, позволяет увеличить пропускную способность до 600 Мбайт/с, а наиболее современный стандарт USB 3.1 (2013 год) позволит достигнуть пропускной способности 1,25 Гбайт/с. Шина USB при этом обеспечивает подключение к компьютеру одновременно нескольких периферийных устройств (принтер, сканер, цифровая камера, веб-камера, модем и др.).

Клавиатура и мышь. Клавиатура и мышь подключаются с помощью порта PS/2 или шины USB (в том числе с помощью беспроводного адаптера).

Звук. К Южному мосту может подключаться интегрированная в системную плату микросхема, которая обеспечивает обработку цифрового звука (эту функцию может выполнять также звуковая плата, которая подключается к шине PCI). С помощью аудиоразъёмов к системной плате могут подключаться микрофон, колонки или наушники.

Архитектура процессоров. Современные процессоры имеют достаточно сложную архитектуру: они содержат не только традиционные для процессоров компоненты (такие как арифметико-логическое устройство (АЛУ), выполняющее вычисления, и устройство управления (УУ), реализующее управление различными устройствами компьютера), но также собственную высокоскоростную оперативную память (кэш-память нескольких уровней), встроенные модули для обработки графической информации и мультимедийных данных.

Большинство современных процессоров являются **много-ядерными** — они содержат в своей структуре 2, 4 или более типовых вычислительных модулей (ядер), тогда как остальные электронные компоненты процессора являются общими для всех его ядер. Каждое такое ядро может работать независимо от остальных, благодаря чему появляется возможность параллелизации вычислений. Например, различные ядра могут одновременно выполнять соответствующее количество компьютерных программ либо одновременно обрабатывать соответствующие фрагменты большого массива информации. Благодаря много-ядерности процессоров удаётся существенно увеличить быстро-



действие компьютера без необходимости увеличения тактовой частоты. В результате в настоящее время при определении потенциального быстродействия компьютерной системы количество ядер процессора является существенно более важной величиной, чем тактовая частота процессора.

Имеющаяся в современных процессорах встроенная быстродействующая память — **кэш-память** позволяет ускорить вычисления за счёт того, что данные, которые предположительно потребуются в ближайших вычислительных операциях, заранее копируются в кэш-память из ОЗУ. Тем самым непосредственно во время вычислений эти данные доступны процессору сразу, без необходимости обмена информацией с оперативной памятью. Обычно кэш-память делится на несколько уровней, различающихся быстродействием и доступным объёмом; эти уровни нумеруют и обозначают как L1, L2 и L3. (L1 — это наиболее быстрая кэш-память малого объёма, L3 — наиболее объёмная, но медленная кэш-память.)

Реализуемая в современных процессорах **конвейерная архитектура** также позволяет ускорить вычисления за счёт того, что каждая стадия обработки команды программы (извлечение команды из ОЗУ и декодирование; адресация и выборка данных; собственно вычисления; сохранение результата) выполняется отдельным блоком — ступенью конвейера. Как только тот или иной блок выполнит соответствующую операцию для очередной команды (например, декодирует её), он сразу же начинает обработку следующей команды, не дожидаясь, когда текущая команда будет выполнена до конца.

Процессоры различных изготовителей по архитектуре делят на три основных семейства: **CISC** (*Complex instruction set computer*), **RISC** (*Reduced instruction set computer*) и **MISC** (*Minimum instruction set computer*), отражающих богатство реализуемых наборов команд. В **CISC-архитектуре** (характерной для процессоров Intel) разработчики процессора стараются реализовать все возможные команды, которые могут встретиться в выполняемых программах. Недостатком такого подхода является «громоздкость» кодирования команд, из-за чего усложняется структура конвейера и увеличивается время выполнения каждой команды. В отличие от этого **RISC-архитектура** предполагает сокращённый набор из основных команд, которые имеют фиксированную длину кода. Такие команды выполняются быстрее, а конвейер для их обработки имеет более простую структуру; процессоры, построенные с использованием RISC-архитектуры, имеют меньшее энергопотребление и меньшее тепловыделение. Однако всегда возможен риск, что в программе встретится

«неизвестная» процессору команда. В этом случае она реализуется как «микропрограмма» — путём выполнения нескольких имеющихся в процессоре команд, которые в итоге дают эквивалентный результат, — что замедляет работу процессора при большом количестве подобных «неизвестных» ему команд в выполняемой программе. **MISC-архитектура** характеризуется дальнейшим развитием идеи RISC, в ней количество реализованных команд уменьшено до 20–30.

Многопроцессорные и кластерные системы. Как уже было сказано, многоядерные процессоры позволяют реализовать параллелизацию вычислений, благодаря которой существенно повышается быстродействие компьютерной системы. Другая возможность реализации параллельных вычислений — использование в одной компьютерной системе нескольких обычных, одноядерных процессоров.

Возможно два типа многопроцессорных вычислительных систем.

В первом случае небольшое количество процессоров (от 2 до 32), каждый из которых имеет собственную кэш-память, подключены к одной общей шине и работают с одной централизованной общей оперативной памятью (технология **UMA: Uniform Memory Access**). Например, в мощных компьютерах-серверах нередко устанавливались два или четыре процессора на одной системной плате.

Второй способ — объединение в одну вычислительную систему нескольких обычных компьютеров, взаимодействующих друг с другом посредством высокоскоростной локальной сети. При этом каждый компьютер имеет собственные процессор, кэш-память, оперативную память и подсистемы ввода/вывода. Такие компьютеры называют **кластерами**, а созданную на их основе многопроцессорную систему называют **кластерной системой** (другое её название — системы с локальной памятью).

Тенденции развития аппаратного обеспечения компьютеров

Можно попытаться проследить тенденции развития аппаратного обеспечения компьютеров и представить себе приблизительный облик «компьютеров будущего».

Микроминиатюризация и мобильность. Продолжается стремление разработчиков к уменьшению размеров и веса компьютерных устройств при одновременном увеличении их мобильности (увеличение времени автономной работы от аккумулятора, беспроводной доступ к сети Интернет), в том числе ценой снижения вычислительной мощности — что вполне допустимо благодаря развитию *облачных технологий*, когда хранение данных и их обработка осуществляются в основном на интернет-серверах.

Совершенствование процессоров в основном ведётся в направлении уменьшения физических размеров размещённых в процессоре электронных элементов (так называемая *технологическая норма*) и в размещении большего количества ядер. Например, многоядерные процессоры, которые несколько лет назад ещё были «экзотикой» для настольных ПК, сегодня широко используются для смартфонов.

Накопители и носители. Так же, как вслед за перфокартами, перфолентами и магнитными лентами ушли в прошлое дискеты, сегодня начинают терять популярность компакт-диски (CD, CD-R, CD-RW). Оптические диски DVD, DVD-R, DVD+R, DVD-RW пока ещё используются, но уступают место более ёмким дискам Blu-Ray, используемым в качестве носителей для видеофильмов высокой чёткости и 3D-видеофильмов. Однако в качестве средства для переноса информации с одного компьютера на другой и даже в качестве средства резервного хранения оптические диски уже используются всё реже. Вместо них растёт популярность носителей на базе флеш-памяти («флешек» и карт флеш-памяти), ёмкость которых возрастает, а стоимость снижается. Для переноса и хранения информации широко используются переносные (карманные) жёсткие диски, но уже появляются первые признаки грядущего отказа от магнитных жёстких дисков в пользу твердотельных (SSD) накопителей, основанных на той же флеш-памяти.

Устройства ввода и управления. Хотя клавиатура и мышь ещё долго будут сохранять популярность, они начинают уступать место средствам голосового ввода (на базе систем распознавания речи), а особенно — тактильным экранам планшетных компьютеров и смартфонов. Всё шире используются средства видеоввода (веб-камеры высокого разрешения и встроенные видеокамеры смартфонов и планшетов), которые используются не только по прямому назначению (съёмка фото и видео, трансляция видео через Интернет), но и для других целей, например для распознавания штрих-кодов или для обеспечения функционирования систем дополненной реальности (за счёт ПО распознавания образов). Совершенствуются сканеры (появляются роботизированные сканеры для сканирования книг с автоматическим перелистыванием страниц), всё шире используются 3D-сканеры различных конструкций.

Устройства вывода. Совершенствование устройств отображения информации (мониторов) в основном ведётся в направлениях роста диагонали, разрешения и качества получаемого изображения, а также поиска новых технологий формирования изображения (например, активно развиваются экраны на базе «электронной бумаги»). Всплеск популярности 3D-мониторов в настоящее время

несколько проходит, по всей вероятности в связи с появлением шлемов виртуальной реальности на базе смартфонов, которые при более низкой цене обеспечивают гораздо больший эффект присутствия. Одновременно растёт интерес производителей вычислительной техники к проекторам: появляются 3D-проекторы, обычные проекторы становятся более миниатюрными и экономичными за счёт использования в качестве источника света светодиодов и полупроводниковых лазеров, а миниатюрные проекторы уже встраивают в планшетные ПК и смартфоны. Последнее позволяет реализовать вывод с таких устройств изображения с диагональю, сравнимой с диагональю ЖК-мониторов настольных компьютеров. В перспективе разработчики обещают реализацию голографических устройств отображения информации, позволяющих работать на устройствах в форм-факторе наручных часов с изображением достаточно большого размера (10–12 дюймов по диагонали).

Ещё одно направление развития устройств вывода — совершенствование устройств печати: рост экономичности (по стоимости печати одного листа бумаги) лазерных принтеров и возвращение конкурентоспособности с ними струйных принтеров за счёт роста качества струйной печати и появления моделей с легко заполняемыми краской контейнерами большой ёмкости вместо дорогостоящих одноразовых картриджей. Также быстро набирают популярность (за счёт снижения цены) 3D-принтеры, для которых находится всё больше перспективных областей применения как в промышленной сфере, так и у обычных, домашних пользователей.

Новые технологии вычислений и новые интерфейсы. Возможно, что в будущем вообще уйдут в прошлое нынешние электронные вычислительные машины, уступив место вычислителям на базе других архитектур и даже технологий. Уже появляются публикации о создании первых действующих моделей оптических и квантовых компьютеров, а нейросетевые системы находят всё больше областей применения в сферах распознавания образов и речи, прогнозирования, «неалгоритмической» обработки данных (пример — созданная в 2016 году российскими программистами онлайн-система обработки фотографий Prisma, позволяющая в течение нескольких секунд превратить обычное фото в картину, нарисованную в стиле одного из великих художников). Наконец, одно из направлений развития ПК (очень перспективных и многообещающих, но вместе с тем — и наиболее спорных!) — это прямое подключение компьютера к мозгу человека с организацией непосредственного ввода/вывода информации. При этом, очевидно, многие существующие устройства ввода/вывода станут ненужными, компьютер фактически окажется скрыт (имплантирован) в теле каждого пользователя и не только появится большое

количество принципиально новых возможностей, недостижимых при помощи любых существующих «отдельных» компьютеров, но и изменится сама парадигма существования человека и его информационного взаимодействия с обществом и с окружающим миром.

<http://mir-3d-world.w.pw> — бесплатный электронный журнал «Мир 3D / 3D World»: информация о современных технологиях и устройствах 3D-сканирования, 3D-печати и отображения 3D-информации, о технологиях виртуальной и дополненной реальности.

http://www.iiorao.ru/iio/pages/izdat/ison/publication/ison_2011/num_1_2011/%D3%F1%E5%ED%EA%EE%E2%20%C4.%DE.pdf — статья о применении надеваемых и вживлённых вычислительных систем.



Вопросы и задания

1. В чём состоит магистрально-модульный принцип построения компьютера?
2. Какие устройства обмениваются информацией через Северный мост?
3. Какие устройства обмениваются информацией через Южный мост?
4. В каком направлении развивается архитектура процессоров?



Практическая работа 1.2

Сведения об архитектуре компьютера

Задание. С помощью системы тестирования компьютера получить сведения об архитектуре компьютера и процессора.

Варианты выполнения работы:

- в операционной системе Windows с помощью программы тестирования компьютера SiSoftware Sandra получить сведения о разных параметрах архитектуры компьютера, используя различные тестирующие модули программы.



Получение сведений об архитектуре компьютера и процессора

В операционной системе Windows получим сведения об архитектуре компьютера и процессора.

1. В операционной системе Windows запустить программу тестирования компьютера SiSoftware Sandra командой [*Все программы—SiSoftware—SiSoftware Sandra Lite*].

В появившемся диалоговом окне программы на вкладке *Устройства* выберем устройства, о которых хотелось бы получить сведения (рис. 1.18).

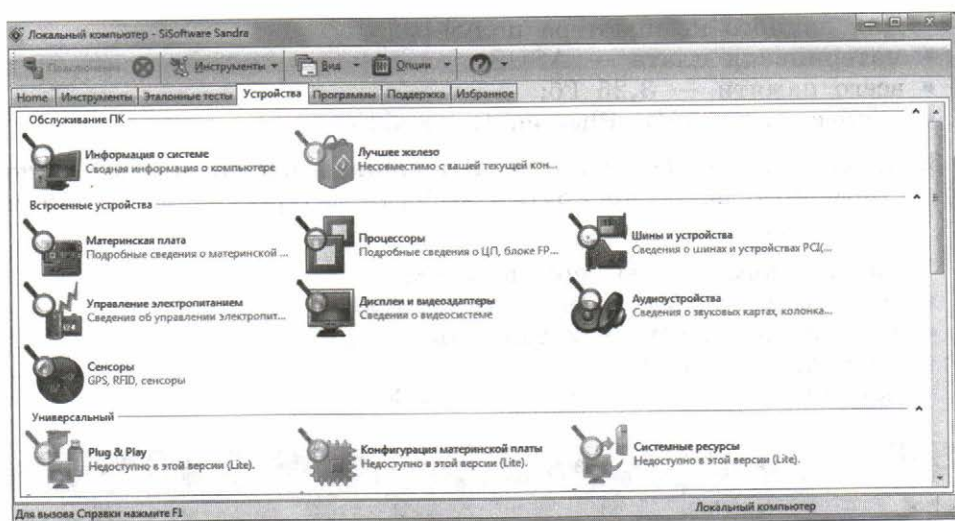


Рис. 1.18

2. Выбрать пункт *Информация о системе*. В появившемся диалоговом окне с помощью полос прокрутки выбрать интересные сведения (1.19).

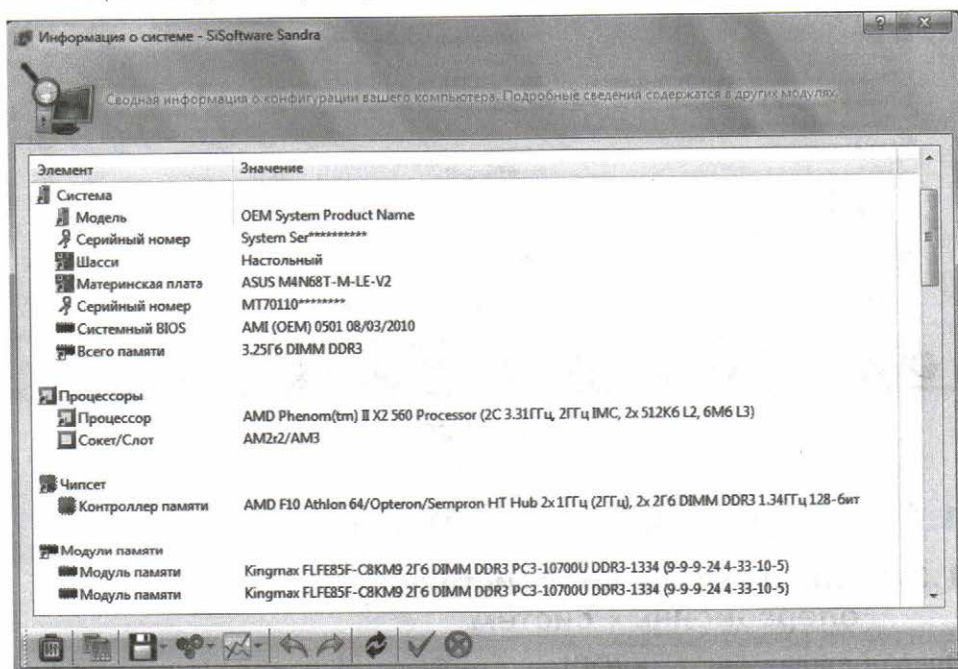


Рис. 1.19

Для данного компьютера получим:

- материнская плата — ASUS;
- всего памяти — 3,25 Гб;
- процессор — AMD Phenom II X2 560.

3. Выбрать пункт *Процессоры*. В появившемся диалоговом окне с помощью полос прокрутки выбрать интересные сведения (рис. 1.20).

Для данного компьютера получим:

- процессор — AMD Phenom II X2 560;
- ядер на процессор — 2 единицы;
- скорость — 3,31 ГГц;
- скорость системной шины — 201 МГц.

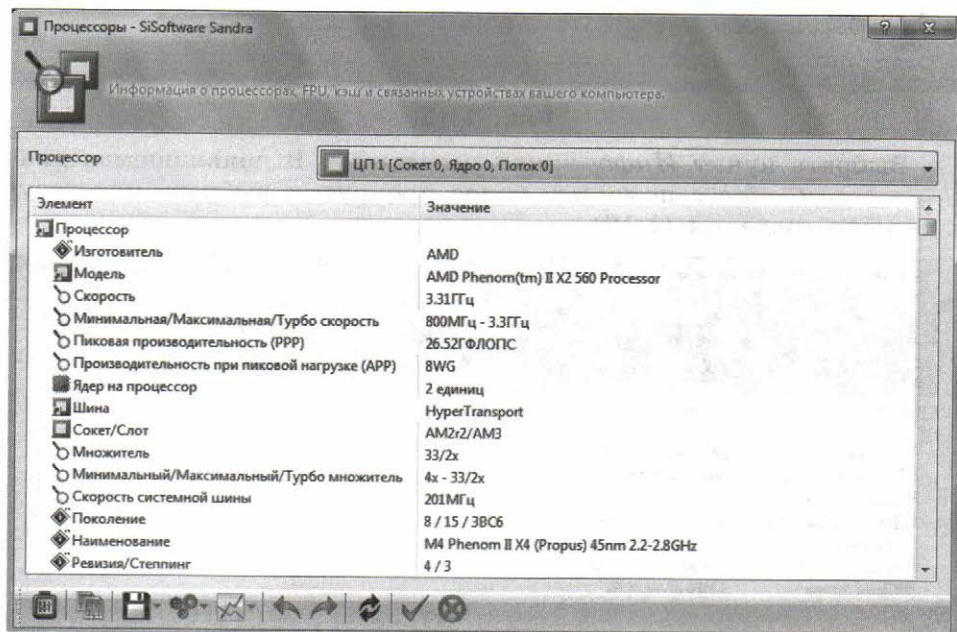


Рис. 1.20

1.3. Операционные системы

1.3.1. Основные характеристики операционных систем



Операционная система (ОС) — это базовый комплекс компьютерных программ, обеспечивающий управление аппаратными средств-

вами компьютера, работу с файловой системой, ввод и вывод данных с помощью периферийных устройств, а также выполнение прикладных программ. Важной частью современных операционных систем являются средства, обеспечивающие работу в локальных сетях и глобальной сети Интернет. Современные операционные системы предоставляют пользователю графический интерфейс.

При включении компьютера операционная система загружается в оперативную память раньше остальных программ и затем обеспечивает их выполнение.

В настоящее время для настольных и портативных персональных компьютеров чаще всего используются операционные системы Microsoft Windows и Linux. На персональные компьютеры фирмы Apple устанавливается операционная система macOS.

Для мобильных устройств (планшетных ПК, смартфонов) чаще всего используются операционные системы Android (разработчик — компания Google), iOS (компания Apple) и Windows (Microsoft).

Файловая система. Разные операционные системы используют различные файловые системы, т. е. различные способы организации, хранения и именования данных на носителях информации. В операционной системе имеются программные модули, управляющие файловой системой.

В процессе работы на компьютере наиболее часто над файлами производятся следующие операции:

- копирование (копия файла помещается в другую папку);
- перемещение (сам файл перемещается в другую папку);
- удаление (запись о файле удаляется из папки);
- переименование (изменяется имя файла).

Минимальным адресуемым элементом носителя информации является кластер, который может включать в себя несколько секторов (объем сектора составляет 512 байт). Размер кластера может составлять от 512 байт до 64 Кбайт. Кластеры нумеруются в линейной последовательности (на магнитных дисках от первого кластера нулевой дорожки до последнего кластера последней дорожки).

Файловая система организует кластеры в файлы и каталоги (папки). Каталог реально является файлом, содержащим список файлов в этом каталоге. Файловая система отслеживает, какие из кластеров в настоящее время используются, какие свободны, какие помечены как неисправные.



Командный процессор. В состав операционной системы входит специальная программа — командный процессор, которая запрашивает у пользователя команды и выполняет их.

Пользователь может дать команду запуска программы, выполнения какой-либо операции над файлами (копирование, удаление, переименование), вывода документа на печать и т. д. Операционная система должна эту команду выполнить.

Драйверы устройств. К системной плате компьютера подключаются различные устройства (жёсткие магнитные диски, оптические дисководы, монитор, клавиатура, мышь, принтер и др.). Каждое устройство выполняет определённую функцию (ввод, хранение, вывод информации), при этом техническая реализация устройств существенно различается.

Драйверы устройств — специальные программы, которые обеспечивают управление работой устройств и согласование информационного обмена с другими устройствами, а также позволяют производить настройку некоторых их параметров. Каждому типу устройств соответствует свой драйвер.

В процессе установки операционной системы определяются тип и конкретные модели установленных устройств и подключаются необходимые для их функционирования драйверы. При включении компьютера производится загрузка драйверов в оперативную память. Пользователь имеет возможность вручную установить или переустановить драйверы.

Графический интерфейс. Для упрощения работы пользователя в состав современных операционных систем входят программные модули, создающие графический пользовательский интерфейс. В операционных системах с графическим интерфейсом пользователь может вводить команды с помощью диалоговых окон и элементов управления.

Диалоговые окна могут включать в себя разнообразные элементы управления:

- вкладки — позволяют использовать «страницы» внутри диалогового окна;
- кнопки — обеспечивают выполнение тех или иных действий;
- текстовые поля, которые позволяют ввести информацию;
- списки — представляют собой наборы предлагаемых на выбор значений;
- переключатели — служат для выбора одного из взаимоисключающих вариантов;
- флажки — обеспечивают присваивание каким-либо параметрам определённых значений;

- **счётчики** — представляют собой пару стрелок и позволяют увеличивать или уменьшать значение в связанном с ними поле;
- **ползунки** — позволяют плавно изменять значения каких-либо параметров;
- **контекстные меню** — позволяют ознакомиться со свойствами объектов, а также выполнить над ними разрешённые операции.

Такие же элементы управления используются и в системах объектно-ориентированного программирования, что позволяет создавать в них приложения с использованием графического интерфейса операционной системы.

В старых операционных системах (типа MS-DOS и первых версий Linux) управление компьютером осуществлялось при помощи текстового интерфейса вводом текстовых команд с наборами их параметров. В отличие от этого графический интерфейс позволяет непосредственно манипулировать изображаемыми на экране объектами — нажимать кнопки, выбирать флажки и переключатели, раскрывать меню и выбирать его пункты, выделять и перетаскивать различные объекты из одного места в другое (в том числе файлы — из одного окна с содержимым папки в другое). Подобные манипуляции обычно выполняются при помощи мыши, а на мобильных устройствах (планшетах, смартфонах) с экраном, чувствительным к нажатию, — и непосредственно пальцами. Благодаря применению графического интерфейса работа с компьютером становится более наглядной, а освоение компьютера — более лёгким для пользователей.

Служебные программы. В состав операционной системы входят также служебные программы, которые позволяют обслуживать диски (проверять, сжимать, дефрагментировать и т. д.), выполнять операции с файлами (архивировать и т. д.), работать в компьютерных сетях и т. д.

Справочная система. Для удобства пользователя в состав операционной системы обычно входит справочная система. Она позволяет оперативно получить необходимую информацию как о функционировании операционной системы в целом, так и о работе её отдельных модулей.

Самотестирование компьютера. После включения компьютера или его перезагрузки процессор начинает считывать и выполнять микропрограммы тестирования компьютера и первого этапа загрузки операционной системы, которые хранятся в микросхеме BIOS.

Включение компьютера осуществляется включением питания (кнопка *Power* на системном блоке). Перезагрузка операционной



системы производится нажатием кнопки *Reset* на системном блоке компьютера или одновременным нажатием комбинации клавиш *Ctrl+Alt+Del* на клавиатуре с последующим выбором команды [*Завершение работы—Перезагрузка*].

После включения или перезагрузки компьютера прежде всего начинает выполняться программа тестирования, которая проверяет работоспособность основных устройств компьютера: процессора, видеоадаптера, оперативной памяти, дисководов, контроллеров жёстких дисков и клавиатуры. В случае обнаружения неисправностей выдаются диагностические сообщения в виде различных последовательностей коротких и длинных звуковых сигналов или в виде текстовых сообщений. После успешной инициализации видеоплаты краткие диагностические сообщения выводятся на экран монитора.

BIOS Setup. Пользователь может установить новые параметры конфигурации компьютера и запомнить их в специальной микросхеме памяти, которая при выключенном компьютере питается от батарейки, установленной на системной плате. Для этого в процессе выполнения самотестирования обычно необходимо нажать клавишу {*Del*}, {*F1*}, {*F2*} или {*F10*} (в зависимости от модели компьютера). Загрузится системная утилита BIOS Setup, имеющая интерфейс в виде системы иерархических меню. В случае выхода из строя батарейки конфигурационные параметры теряются. При этом компьютер перестаёт нормально загружаться либо после выдачи пользователю соответствующего запроса компьютер загружается с параметрами BIOS, задаваемыми по умолчанию.

Загрузка операционной системы. Файлы операционной системы хранятся во внешней, долговременной памяти (на магнитных, оптических дисках или во флеш-памяти). Однако программы могут выполняться, только если они находятся в оперативной памяти, поэтому файлы операционной системы необходимо загрузить в оперативную память. Разрешение этого противоречия состоит в последовательной, поэтапной загрузке операционной системы.



Диск, на котором находятся файлы операционной системы и с которого производится её загрузка, называется **системным**.

После включения компьютера производится его самотестирование. Затем наступает первый этап загрузки операционной системы — специальная программа, содержащаяся в микросхеме BIOS, начинает поиск загрузчика операционной системы. Современные версии BIOS позволяют загружать операционную систему не только с магнитных и оптических дисков, но и с флеш-памяти.

Если диск системный и программа-загрузчик оказывается на месте, то она загружается в оперативную память и ей передаётся управление работой компьютера. Программа ищет файлы операционной системы на системном диске и загружает их в оперативную память в качестве программных модулей.

Если системные диски в компьютере отсутствуют, на экране монитора появляется сообщение «Non system disk», и компьютер «зависает», т. е. загрузка операционной системы прекращается и компьютер остается неработоспособным.

Описанный выше двухступенчатый механизм загрузки операционной системы обеспечивает универсальность. Программа загрузки, содержащаяся в BIOS, всегда одна и та же и запускает загрузчик операционной системы, который всегда располагается на носителе в одном и том же месте. Сам же загрузчик операционной системы уже для каждой операционной системы свой и учитывает её конкретные особенности. Поэтому на компьютер можно установить и загружать практически любую операционную систему (если только эта ОС поддерживает данный тип компьютеров).

Установка на компьютер различных операционных систем. На компьютер могут быть установлены одновременно несколько разных операционных систем. Для этого жёсткий диск должен быть разбит на **логические разделы** (независимые области на диске), в каждом из которых может быть создана своя файловая система.

Таким образом, на компьютер в разные логические разделы могут быть установлены операционные системы семейства Microsoft Windows, Linux и macOS. Для перехода от одной операционной системы к другой необходимо перезагрузить компьютер.

Использовать на одном и том же компьютере несколько различных операционных систем позволяют так называемые **виртуальные машины**. Для создания на компьютере виртуальной машины и для работы с созданной виртуальной машиной необходимо запустить специальную программу. Она выделяет особую область в оперативной памяти компьютера и создает на его диске специальный файл, представляемые для пользователя как ОЗУ и жёсткий диск отдельного, «самостоятельного» компьютера. На таком «виртуальном компьютере» можно установить любую ОС (как на «настоящем» компьютере) и уже в среде этой ОС устанавливать различные программы. При выполнении программ (в том числе ОС, которую в этом случае называют «гостевой») в виртуальной машине этим программам доступны все вычислительные ресурсы данного компьютера (если только

отдельные ресурсы не заблокирует сам пользователь, например, из соображений безопасности). Применение виртуальных машин позволяет легко переходить от одной операционной системы к другой и одновременно работать в нескольких операционных системах, просто переключаясь между окнами виртуальных машин, без перезагрузки компьютера.



Вопросы и задания

1. Каковы состав и функции операционных систем?
2. Как происходит загрузка операционной системы?



Практическая работа 1.3

Сведения о логических разделах дисков

Задание. В операционных системах Windows и Linux получить сведения о логических разделах дисков.

Варианты выполнения работы:

- в операционной системе Windows или в операционной системе Linux.



Получение сведений

о логических разделах дисков



В операционной системе Windows получим сведения о логических разделах дисков.

1. В операционной системе Windows запустить служебную программу управления дисками командой [*Панель управления—Администрирование—Управление компьютером*].
2. В левой части появившегося диалогового окна *Управление компьютером* выбрать пункт *Управление дисками*.

Будут показаны логические разделы дисков с информацией об их объёмах и файловых системах (рис. 1.21).

На данном компьютере Диск 0 поделён на четыре логических раздела.

Из них два последних логических раздела без названия принадлежат другой операционной системе (в данном случае Linux).



В операционной системе Linux получим информацию о доступных файловых устройствах и их файловой системе, объёме дисков и свободном пространстве на них.

3. В операционной системе Linux запустить утилиту определения свободного дискового пространства [*Прочее—Система—KDiskFree (Использования дисков)*].

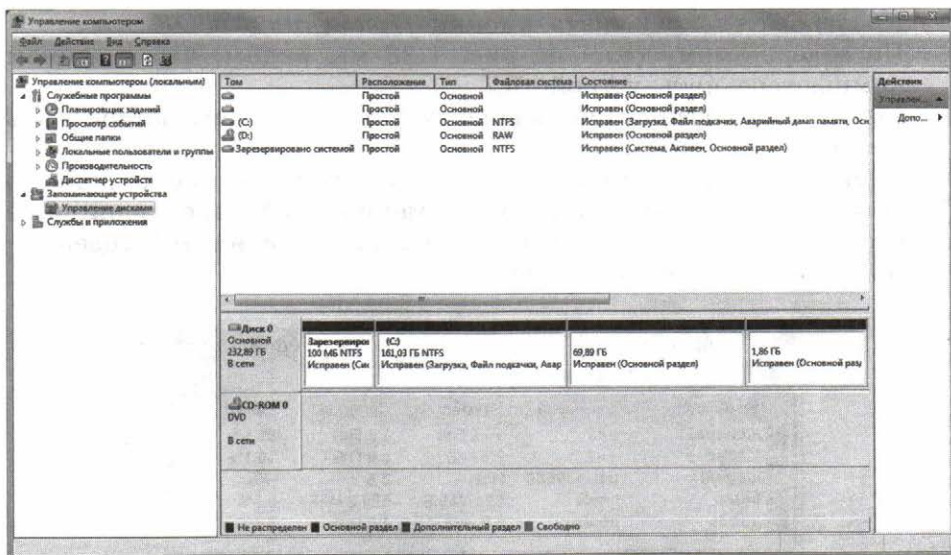


Рис. 1.21

4. Настроить параметры отображения информации о доступных файловых устройствах, выполнив команду [Настройка—Настроить KDiskFree...].

В появившемся диалоговом окне *Настройка—KDiskFree* на вкладке *Параметры* установить галочки для отображения нужных параметров доступных дисков (рис. 1.22).

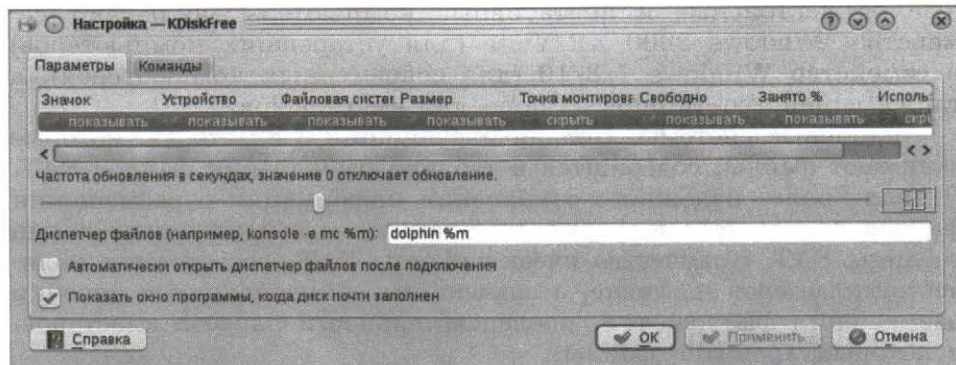


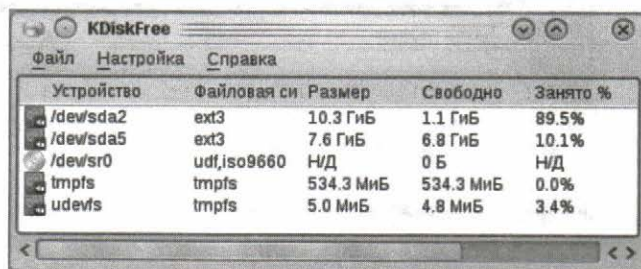
Рис. 1.22

Последовательно нажать кнопки *Применить* и *ОК*.

По содержимому окна на рис. 1.23 мы видим, что в данном компьютере пять устройств.

/dev — это обозначение основных устройств компьютера в ОС Linux.

Оптический диск получил имя sr0, основные диски данного компьютера — sda2 и sda5, временное файловое хранилище имеет имя tmpfs; менеджер устройств, управляющий содержимым /dev, имеет имя udevfs.



Устройство	Файловая си	Размер	Свободно	Занято %
/dev/sda2	ext3	10.3 ГиБ	1.1 ГиБ	89.5%
/dev/sda5	ext3	7.6 ГиБ	6.8 ГиБ	10.1%
/dev/sr0	udf,iso9660	Н/Д	0 Б	Н/Д
tmpfs	tmpfs	534.3 МиБ	534.3 МиБ	0.0%
udevfs	tmpfs	5.0 МиБ	4.8 МиБ	3.4%

Рис. 1.23

1.3.2. Операционная система Windows

Операционная система Windows остаётся наиболее популярной операционной системой для персональных компьютеров. Эта операционная система установлена на большинстве персональных компьютеров в мире.

Операционные системы корпорации Microsoft, ориентированные на настольные и портативные компьютеры, включают семейство Windows 2000/XP/Vista (для устаревших компьютеров) и семейство Windows 7/8/10 (для современных компьютеров), а также семейство Windows Phone для смартфонов.

Файловые системы. Полная информация о кластерах, которые занимают файлы, содержится в таблице размещения файлов FAT. В целях более надёжного сохранения информации о размещении файлов на носителе информации хранятся две идентичные копии таблицы FAT. Количество ячеек таблицы FAT соответствует количеству кластеров на диске, а значениями ячеек являются цепочки размещения файлов, т. е. последовательности адресов кластеров, в которых хранятся файлы.

FAT16. Эта файловая система выделяет 16 бит для хранения адреса кластера, соответственно, она может адресовать $2^{16} = 65\,536$ кластеров.

Большинство файловых систем, в которых используется FAT16, способны работать только с кластерами размером 32 Кбайт, и максимально возможный объём носителей информации в этом случае не превышает 2 Гбайт:

$$32 \text{ Кбайт} \cdot 56\,536 = 1\,809\,152 \text{ Кбайт} \approx 1766 \text{ Мбайт} \approx 2 \text{ Гбайт.}$$



По этой причине FAT16 сегодня почти не используется.

FAT32. Эта файловая система выделяет 32 бита для хранения адреса кластера, соответственно, она может адресовать $2^{32} = 4\,294\,967\,296$ кластеров. Объём кластера может составлять от 512 байт до 32 Кбайт, но по умолчанию равен 8 секторов (4 Кбайт), поэтому FAT32 может использоваться для носителей информации объёмом:

$$\begin{aligned} 4 \text{ Кбайт} \cdot 4\,294\,967\,296 &= 17\,179\,869\,184 \text{ Кбайт} = \\ &= 16\,777\,216 \text{ Мбайт} = 16\,384 \text{ Гбайт} = 16 \text{ Тбайт.} \end{aligned}$$



Таким образом, FAT32 может использоваться для жёстких дисков самого большого объёма. Однако поскольку в ячейках FAT должны быть указаны номера кластеров, максимально возможный номер кластера равен $2^{32} - 1 = 4\,294\,967\,295$. По этой причине при использовании FAT32 максимально возможный размер файла составляет приблизительно 4 Гб. Файлы же большего объёма (например, видеофайлы) не могут быть записаны на носитель с FAT32.

NTFS. Эта файловая система позволяет устанавливать различные объёмы кластера (от 512 байт до 64 Кбайт, по умолчанию 4 Кбайт).

NTFS по сравнению с FAT32 увеличивает надёжность и эффективность использования дискового пространства. NTFS использует систему журналирования для повышения надёжности файловой системы. Журналируемая файловая система создаёт и сохраняет список изменений, которые она будет проводить с файловой системой перед фактическим выполнением этих изменений. Такие записи хранятся в отдельной части файловой системы, называемой журналом, или логом. Как только изменения файловой системы будут внесены в журнал, журналируемая файловая система применит эти изменения к файлам. В случае непредвиденных сбоев при выполнении файловых операций NTFS может восстановить повреждённые данные на диске, используя записи в журнале.

Графический интерфейс. *Рабочий стол* — это основа и вершина графического интерфейса операционной системы. Он появляется после включения компьютера и входа в операционную систему Windows. Подобно поверхности обычного стола, он



служит рабочей «поверхностью», на которой появляются диалоговые окна запущенных приложений и открытых папок.

Панель задач располагается у нижнего края экрана. На ней отображаются значки запущенных приложений, позволяющие переключаться между ними. Она также содержит слева кнопку *Пуск*, которой можно воспользоваться для доступа к программам, папкам и параметрам компьютера, а справа располагаются индикаторы раскладки клавиатуры, сетевых подключений и часы.

В операционной системе Windows 8 после включения компьютера на экране вначале появляется полоса меню с иконками («плитками») основных установленных программ. Такой интерфейс получил название *Metro* и позволяет запустить нужную программу сразу же, не входя в *Рабочий стол* (который в этом случае тоже запускается как одна из имеющихся программ). При этом меню *Пуск* на *Рабочем столе* было убрано. Однако подобный интерфейс оказался неудобным для пользователей, и в последней версии Windows 10 её разработчики снова сделали *Рабочий стол* основой интерфейса и вернули в него меню *Пуск* (часть которого выполнена из «плиток» наподобие меню *Metro*).

На *Рабочем столе* располагаются **значки** (иконки) — маленькие рисунки, обозначающие программы, файлы, папки и другие объекты. При первом запуске Windows на *Рабочем столе* имеется как минимум один значок: *Корзина*. Пользователь компьютера может добавить на *Рабочий стол* и другие значки. Двойной щелчок по значку на *Рабочем столе* запускает или открывает объект, который значок представляет.

Для упрощения доступа к ресурсам компьютера с *Рабочего стола* можно создать ярлыки избранных файлов и приложений. **Ярлык** — это значок, представляющий ссылку на объект, а не сам объект (ярлык можно узнать по стрелке на значке). Двойной щелчок по ярлыку открывает объект. При удалении ярлыка удаляется только ярлык, но не исходный объект. Многие приложения при установке создают ярлыки на *Рабочем столе* автоматически.

Открываемые приложения, файлы или папки появляются на *Рабочем столе* в окнах (именно от них операционная система Windows получила своё название). Хотя содержимое каждого окна индивидуально, у всех окон есть одни и те же основные элементы.

- **Строка заголовка.** Отображает название документа, программы или папки.
- **Кнопки сворачивания, разворачивания и закрытия окна.** С помощью этих кнопок можно, соответственно, скрыть окно,

развернуть его на весь экран или закрыть (более подробно об этом немного далее).

- **Строка меню.** Содержит команды, выбираемые щелчком мышь.
- **Полоса прокрутки.** Позволяет прокручивать содержимое окна для просмотра информации.
- **Границы и углы.** Можно перетаскивать их указателем мыши, изменяя размер окна.
- **Меню, кнопки, полосы прокрутки, флажки и другие элементы управления** позволяют выбирать команды, изменять параметры и работать с окнами.

Безопасность компьютера. Центр обеспечения безопасности Windows показывает текущее состояние защиты компьютера и рекомендует меры по усилению безопасности. Центр обеспечения безопасности проверяет следующие основные элементы защиты компьютера (рис. 1.24).

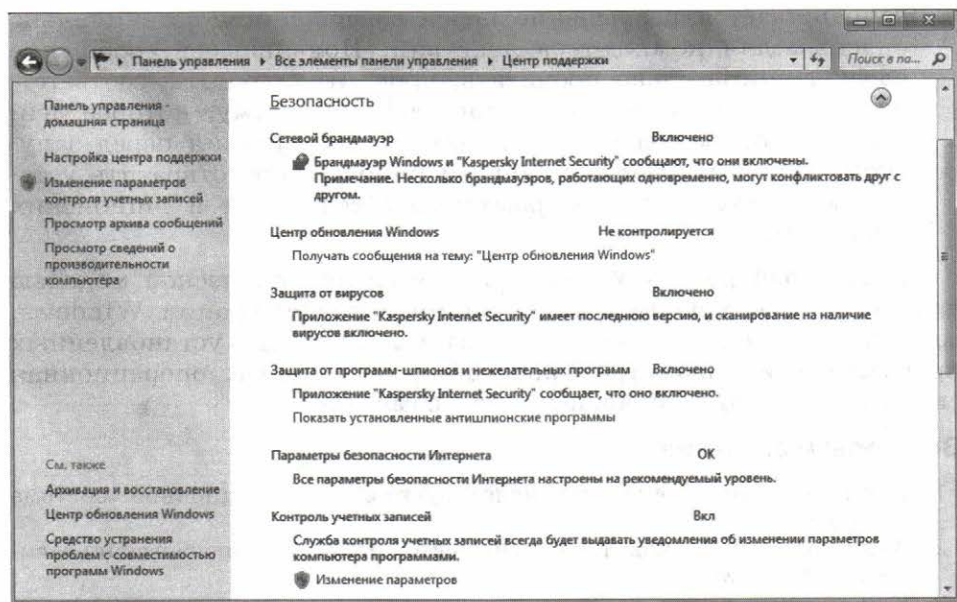


Рис. 1.24

- **Брандмауэр (сетевой брандмауэр, межсетевой экран).** Брандмауэр защищает компьютер, предотвращая доступ к нему из локальной сети или из Интернета хакеров и вредоносных программ.

- *Центр обновления Windows.* Windows может регулярно проверять наличие обновлений и автоматически их устанавливать.
- *Защита от вирусов.* Антивирусное программное обеспечение (не входит в состав операционной системы) должно быть установлено, это поможет защитить компьютер от вирусов, червей и других угроз безопасности.
- *Параметры безопасности Интернета.* Проверяются параметры безопасности браузера Internet Explorer.
- *Контроль учётных записей* позволяет предотвратить несанкционированные изменения в компьютере. Контроль учётных записей позволяет создать для каждого пользователя отдельный вход для работы с компьютером, контролировать его и при помощи пароля ограничивать доступ к компьютеру посторонних людей. При этом каждому пользователю назначается статус с ограниченными правами — с определённым набором действий, которые может выполнять данный пользователь (например, может быть запрещена установка на компьютер новых программ). Для учётной записи обеспечивается защита, запрашивается разрешение перед совершением потенциально опасных для компьютера действий. Появившееся сообщение следует внимательно прочитать, проконтролировать, соответствует ли название выполняемого действия тому, которое действительно производится. Проверяя эти действия перед запуском, контроль учётных записей помогает предотвратить установку вредоносного программного обеспечения и шпионских программ.

Системный реестр Windows является иерархической моделью данных, в нём хранится информация о конфигурации Windows, содержатся сведения об оборудовании системы, установленных программах и параметрах настройки, к которым операционная система постоянно обращается во время работы.



Вопросы и задания

1. Какие файловые системы используются в операционной системе Windows?
2. Какие элементы входят в графический интерфейс операционной системы Windows?
3. Как обеспечивается безопасность в операционной системе Windows?



Практическая работа 1.4

Значки и ярлыки на Рабочем столе

Задание 1. В операционной системе Windows установить нужные значки на *Рабочем столе*.

Задание 2. В операционной системе Windows поместить на *Рабочий стол* ярлык часто используемого приложения, папки или устройства (например, ярлык принтера).

Варианты выполнения работы:

- выбрать разные наборы значков на *Рабочем столе*;
- выбрать создание ярлыков различных приложений, папок и устройств.



Установка значков на Рабочем столе в операционной системе Windows

В операционной системе Windows установим требуемые значки на *Рабочем столе*.

1. В операционной системе Windows в контекстном меню *Рабочего стола* выбрать пункт *Персонализация*.
2. В левой части появившегося диалогового окна *Панель управления\Оформление и персонализация\Персонализация* активизировать ссылку *Изменение значков рабочего стола*.
3. В появившемся диалоговом окне *Параметры значков рабочего стола* установить флажки тех значков, которые нужны пользователю (рис. 1.25).

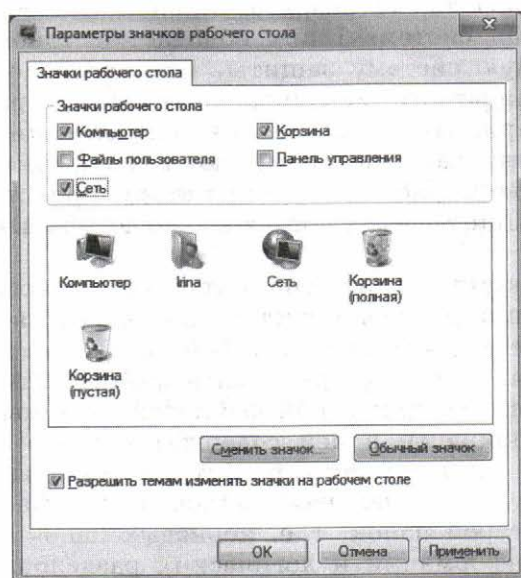


Рис. 1.25



Создание ярлыка на Рабочем столе в операционной системе Windows

В операционной системе Windows поместим на *Рабочий стол* ярлык (например, принтера).

1. В операционной системе Windows ввести команду *Устройства и принтеры*.
2. В появившемся диалоговом окне *Панель управления\Оборудование и звук\Устройства и принтеры* выбрать значок принтера и перетащить его на Рабочий стол.

На *Рабочем столе* появится ярлык принтера (рис. 1.26).



Рис. 1.26

1.3.3. Операционная система Linux

Операционная система Linux — ядро и основные компоненты системы, а также большинство пользовательских приложений для Linux — свободно распространяемые программы. Это означает, что их можно запускать на любом количестве компьютеров, без ограничений распространять, получать исходные тексты этих программ и вносить в них любые исправления. Свобода использования программ обеспечила их широкое применение и интерес к ним со стороны тысяч разработчиков и пользователей.

Операционная система Linux обладает высокой надёжностью и имеет отличную систему защиты. Благодаря доступности сети Интернет и открытости исходных текстов программ, сообщить об ошибке и принять участие в её исправлении независимому программисту или даже пользователю так же просто, как и специалисту фирмы-разработчика или автору проекта. Именно поэтому ошибки защиты выявляются особенно эффективно и быстро исправляются.

Файловая система. Операционные системы хранят данные на диске при помощи файловых систем. Классическая иерархическая файловая система представляет собой вложенные друг в друга папки, в которых могут содержаться и файлы. Одна из папок является вершиной иерархической файловой системы (а выражаясь технически — корнем), в ней содержатся все остальные папки и файлы. Если жёсткий диск разбит на разделы, то в каждом разделе организуется отдельная файловая система с собственным корнем и структурой папок, т. е. корневых папок столько, сколько носителей информации и логических разделов.

В операционной системе Linux корневая папка всегда только одна, а все остальные папки в неё вложены, т. е. для пользова-

теля файловая система представляет собой единое целое. В действительности разные части файловой системы могут находиться на совершенно разных устройствах: разных разделах жёсткого диска, на разнообразных съёмных носителях (оптических дисках, флеш-памяти), даже на других компьютерах (с доступом через сеть). Для того чтобы получить единое файловое дерево с одним корнем, используется процедура монтирования.

Монтирование — это подключение в одну из папок целой файловой системы, находящейся где-то на другом устройстве. Для монтирования необходима пустая папка — она называется **точкой монтирования**. Подключённую (смонтированную) файловую систему можно в любой момент отключить — **размонтировать**, после чего та папка, куда она была смонтирована, снова окажется пустой. На рисунке 1.27 показан смонтированный в папку hda1 логический раздел с установленной операционной системой Windows.

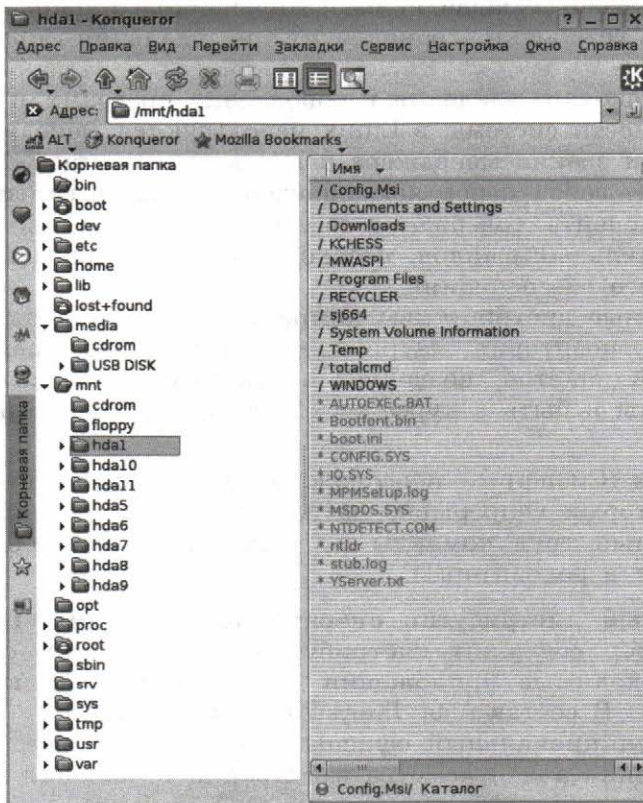


Рис. 1.27

Пользователю обычно не требуется выполнять монтирование и размонтирование вручную: при загрузке системы будут смонтированы все устройства, на которых хранятся части файловой системы, а при останове (перед выключением) системы все они будут размонтированы. Файловые системы на съёмных носителях (оптических дисках и пр.) также монтируются и размонтируются автоматически — при подключении носителя либо при обращении к соответствующему каталогу.

В операционной системе Linux чаще всего используется журналируемая файловая система `ext3`, в которой кластер может иметь размер от 1 до 8 Кбайт. Например, файловая система `ext3` используется в версии ALT Linux, широко применяемой в школах.



Дистрибутивы операционной системы Linux. Большинство пользователей для установки Linux используют дистрибутивы. Дистрибутив — это не просто набор программ, а ряд решений для разных задач пользователей, объединённых едиными системами установки, обновления и удаления программных пакетов.

Для установки, удаления и обновления программ и поддержания целостности системы в Linux используются **менеджеры пакетов**. С точки зрения менеджера пакетов программное обеспечение представляет собой набор компонентов — **программных пакетов**. Такие компоненты содержат в себе набор исполняемых программ и вспомогательных файлов, необходимых для корректной работы программного обеспечения.

Стабильные дистрибутивы компании ALT Linux выпускаются на основе репозитория Sisyphus — это ежедневно обновляемый репозиторий пакетов, позволяет пользователям легко обновлять свои системы и быть в курсе всех новостей мира свободных программ.



Sisyphus (Сизиф) — персонаж греческой мифологии. Миф о Сизифе, который непрерывно катил в гору камень, символизирует постоянный труд команды по усовершенствованию решений, заложенных в репозиторий.

Школьный дистрибутив операционной системы Linux. Альт Линукс Юниор содержит дистрибутив операционной системы, а также необходимые приложения для преподавания курса «Информатика». В состав Альт Линукс Юниор входят:

- полнофункциональный офисный пакет OpenOffice;
- приложения для просмотра веб-страниц и общения в сети Интернет;
- редакторы векторной и растровой графики;

- приложения для обработки цифровых фотографий и нелинейного монтажа цифрового видео;
- звуковые редакторы для прослушивания и редактирования аудиофайлов;
- современные системы алгоритмического и объектно-ориентированного программирования.

Вход в систему. Операционная система Linux — **многопользовательская система**. На практике это означает, что для работы в системе нужно в ней **зарегистрироваться**, т. е. ввести системное имя пользователя — **логин** (login name) и **пароль** (password). Загрузка операционной системы заканчивается интерфейсом входа в систему: выводится приглашение ввести логин и пароль.

В операционной системе Linux у каждого пользователя обязательно есть свой «домашний» каталог, предназначенный для хранения всех собственных данных пользователя. Именно с этого каталога пользователь начинает работу после регистрации в системе. «Домашние» каталоги пользователей обычно собраны в каталоге /home, их имя чаще всего совпадает с учётным именем пользователя в системе: Например, для пользователя test домашним каталогом будет /home/test.

Пользователь является полным хозяином внутри своего каталога, однако остальная часть файловой системы доступна ему только для чтения, но не для записи. Такая система разграничения доступа к файлам позволяет не бояться вирусов и защищает данные от случайного повреждения другими пользователями.

Графический интерфейс. Графический интерфейс операционной системы Linux можно настраивать. Один из вариантов — это использование графической оболочки KDE (от англ. *Kool Desktop Environment*), которая позволяет создать графический интерфейс, очень похожий на графический интерфейс операционной системы Windows. Этот графический интерфейс базируется на *Рабочем столе*, в нём есть *Панель задач* и (слева) кнопка меню запуска приложений (аналог кнопки *Пуск*). Справа на *Панели задач* располагаются индикаторы раскладки клавиатуры, сетевых подключений и часы.

Приложения и папки открываются на *Рабочем столе* в диалоговых окнах, содержащих элементы управления: меню, кнопки, полосы прокрутки, флажки, переключатели и т. д.

Выключение компьютера. Компьютер с загруженной операционной системой Linux нельзя выключать, просто отключив электропитание компьютера. Большое количество информации, которая должна располагаться на диске, система держит в



оперативной памяти для повышения быстродействия. Неожиданное выключение питания приводит к потере этой информации. При работе в графической среде для завершения работы нужно выбрать соответствующий пункт в главном меню. В графическом интерфейсе входа в систему также есть кнопка меню, в котором имеется пункт «выключить компьютер».



Вопросы и задания

1. Какая файловая система используется в операционной системе Linux?
2. В чём состоит смысл монтирования и размонтирования в операционной системе Linux?
3. Какие элементы входят в графический интерфейс операционной системы Linux?



Практическая работа 1.5

Настройка графического интерфейса операционной системы Linux

Задание. В операционной системе Linux осуществить настройку графического интерфейса в среде KDE.

Варианты выполнения работы:

- настроить различные параметры графического интерфейса.

Настройка графического интерфейса в среде KDE



В операционной системе Linux осуществим настройку графического интерфейса в среде KDE.

1. В операционной системе Linux в меню запуска приложений выбрать пункт [*Прочее—Настройка—Параметры системы*].
2. В появившемся диалоговом окне *Параметры системы* щёлкнуть по значку *Рабочий стол* и перейти на вкладку *Хранитель экрана*.
3. В появившемся справа диалоговом окне *Хранитель экрана* установить нужные параметры (рис. 1.28). Выбрать из списка заставку (например, *Часы*) и с помощью счётчика установить, через сколько минут бездействия она появляется (например, *10 мин*).

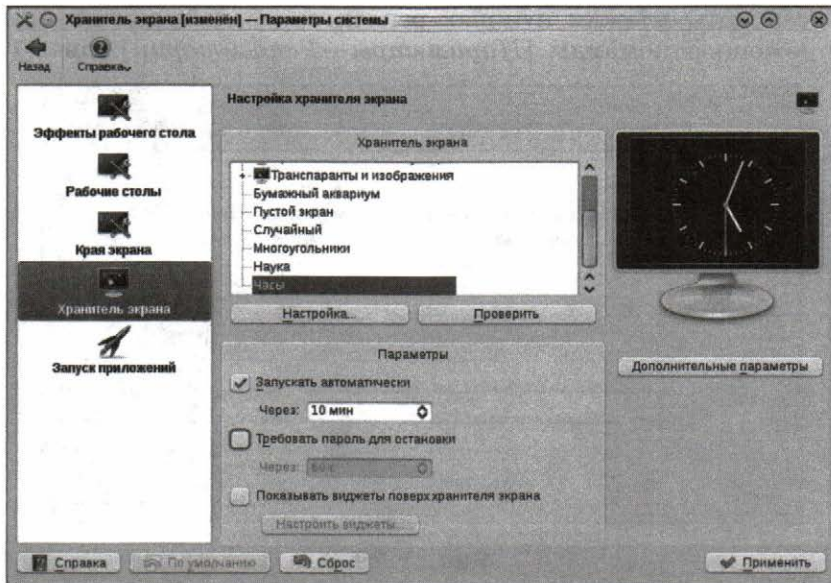


Рис. 1.28

Практическая работа 1.6

Установка пакетов в операционной системе Linux¹⁾

Задание. В операционной системе Linux установить браузер SeaMonkey (с использованием репозитория).

Варианты выполнения работы:

- выбрать различные комбинации пакетов (для различных комбинаций пакетов изменяется внешний вид и функциональные возможности браузера).

Установка программных пакетов приложений в операционной системе Linux



Определим репозитории²⁾, в которых могут быть пакеты требуемого приложения.

1. Ввести команду [*Настройка—Менеджер пакетов (Программа управления пакетами Synaptic)*].

¹⁾ Устанавливаемые пакеты предлагаются учителем по его выбору. Работа производится на домашнем компьютере (так как требуются права системного администратора).

²⁾ Репозиторий — место, где хранятся и поддерживаются какие-либо данные, чаще всего в виде файлов, доступных для распространения по сети.

Установить флажки нужных репозиторийев, которые определить с помощью команды [Параметры—Репозитории] (рис. 1.29).

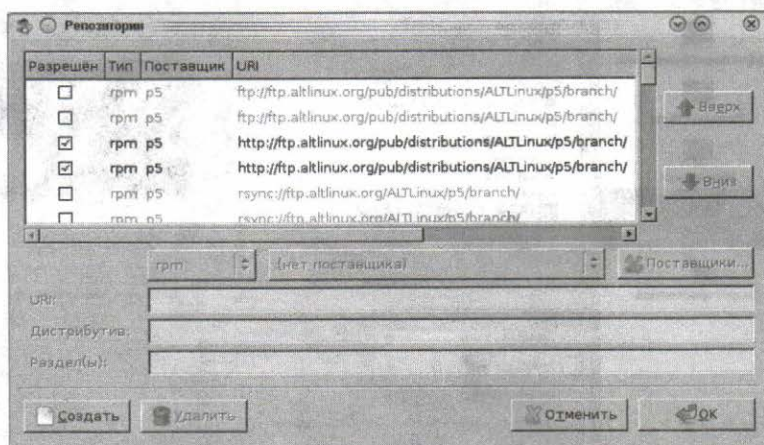


Рис. 1.29

Найдём с помощью меню в левом окне и перечне пакетов в правом окне нужную программу (в данном случае браузер SeaMonkey).

- Найти в левом окне пункт *Сети/WWW*, а в правом окне — *SeaMonkey*.

Будет найдено несколько пакетов, необходимых для установки этого приложения (рис. 1.30).

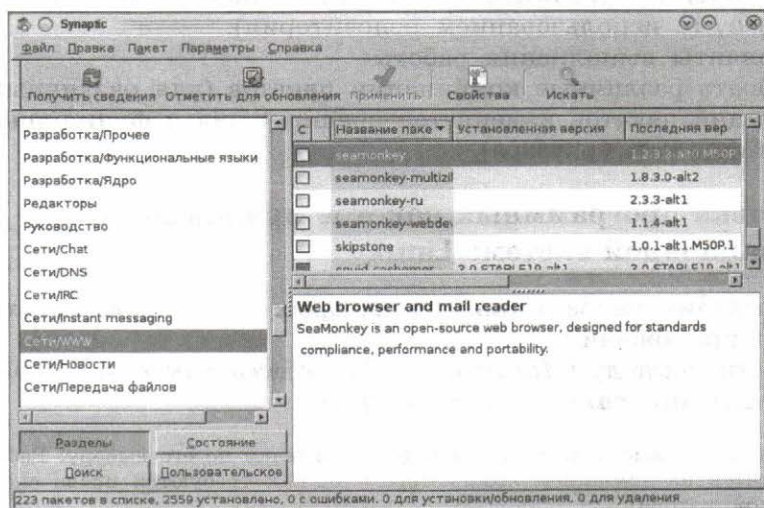


Рис. 1.30

Установим выбранные пакеты (приложение).

3. В контекстном меню выбранных пакетов активизировать пункт *Отметить для установки*.

В меню программы *Установка пакетов* выбрать команду [*Применить*].

В появившемся диалоговом окне выбрать пункт *Для установки* и щёлкнуть по кнопке *Применить* (рис. 1.31).

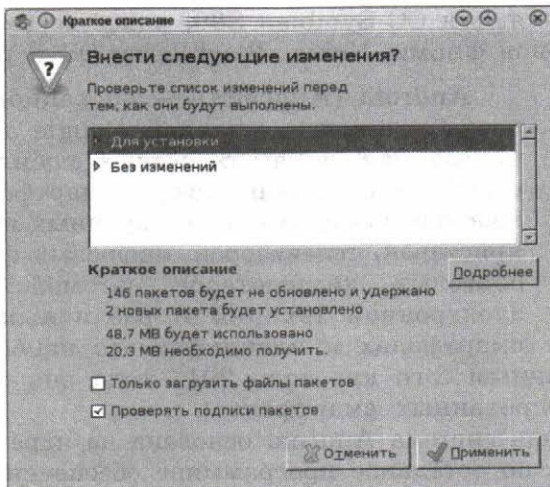


Рис. 1.31

Пакеты (приложение) будут установлены вместе с требуемыми для функционирования приложения пакетами. Программа автоматически получит пункт в меню.

1.3.4. Мобильные операционные системы

Мобильные устройства (планшеты, смартфоны) значительно отличаются от настольных компьютеров как аппаратной базой (используемыми в их схеме электронными компонентами), так и принципами работы с ними. Для мобильных устройств характерны следующие особенности:

- компактность по весу и по размерам;
- пониженное энергопотребление (для обеспечения длительной автономной работы от аккумулятора);
- обеспечение беспроводных коммуникаций (сотовая связь, беспроводные каналы Bluetooth, Wi-Fi, 3G и т. д.);
- чувствительный к нажатию (сенсорный) экран, одновременно выполняющий функции монитора и мыши, и соответствующие отличия в принципах работы пользователя с информацией;

- повышенные требования к защищённости информации и к простоте работы с устройством, рассчитанным на широкий круг пользователей.

Эти и другие особенности мобильных устройств потребовали создания для них особых разновидностей операционных систем, среди которых основными являются Android, iOS и Windows специальных версий (Windows Mobile, Windows Phone). Менее известными являются ОС Symbian S60, ранее весьма популярная среди смартфонов фирмы Nokia, BlackBerry OS и ряд других.



Android (Андроид) — операционная система, разработанная компанией Google и широко используемая на самых разных мобильных устройствах — от планшетов и смартфонов до электронных книг, «умных» наручных часов, игровых приставок, телевизоров, цифровых фотоаппаратов, бортовых компьютеров автомобилей, бытовой электронной техники и различных роботов. На смартфонах и планшетах это наиболее популяр-

ная ОС (по данным 2-го квартала 2014 года она была установлена на 86% проданных смартфонов).

Операционная система Android основана на ядре (основной части) ОС Linux, но остальное программное обеспечение у Android-систем своё. Программные приложения для Android обычно разрабатываются на языке программирования Java и чаще всего распространяются через специальный сайт — «магазин приложений» Google Play, где имеются как коммерческие (платные), так и бесплатные программы. При обращении к «магазину приложений» непосредственно с мобильного устройства автоматически определяется его модель и используемая версия операционной системы, а при наличии совместимости выбираемой из онлайн-каталога программы с данным устройством загрузка и установка программы выполняется непосредственно через Интернет. Возможно также распространение программ для ОС Android в виде дистрибутивов — специальных файлов в формате APK (*Android Package*) — архивных файлов, содержащих все предназначенные для установки файлы программы.

Файловая система ОС Android в целом аналогична файловой системе ОС Linux.

Наиболее современными версиями ОС Android, используемыми в мобильных устройствах, поступающих в продажу, являются версии Android 4.4 KitKat (октябрь 2013 года) и Android 5.0/5.1/5.1.1 Lollipop (октябрь 2014 года). А в мае 2015 года была анонсирована новейшая версия Android M.

Популярность ОС Android подтолкнула компанию Google и к выпуску версий (сборок) этой операционной системы для «обычных» настольных компьютеров и ноутбуков, а в 2015 году для браузера Google Chrome было предложено дополнение, позволяющее запускать программы для ОС Android на настольных компьютерах.

iOS (до июня 2010 года — iPhone OS) — операционная система для планшетов iPad, смартфонов iPhone и медиапроигрывателей iPod, разработанная фирмой Apple. Ядро этой ОС аналогично ядру ОС Apple macOS для настольных компьютеров фирмы Apple, но операционная система iOS способна работать только с процессорами, используемыми в мобильных компьютерах.

Распространение программ для устройств, работающих под управлением iOS, производится через фирменный сайт — «магазин приложений» Apple App Store, загрузка и установка программ производится через Интернет при непосредственном обращении к сайту «магазина приложений» с мобильного устройства под управлением iOS.

Мобильные версии Windows — Windows Mobile (использовалась с 2000 по 2011 год) и Windows Phone (с октября 2010 года) — разработаны компанией Microsoft параллельно с разработкой «настольных» версий Windows с учётом особенностей мобильных устройств. Так, в операционной системе **Windows Phone** использован «плиточный» интерфейс меню установленных программ, аналогичный меню Metro в Windows 8.

Вместе с тем компания Microsoft в последние годы при разработке ОС Windows всё больше ориентировалась на универсальность операционной системы для настольных и мобильных устройств. Так, уже в версии Windows 8 появилось «плиточное» меню Metro, явно ориентированное на устройства с сенсорным экраном, и предусмотрена установка программ через Интернет из фирменного «магазина приложений» Windows Store. А в новейшей версии Windows 10 уже речь идёт о единой ОС для всех устройств — настольных компьютеров, ноутбуков, планшетов и смартфонов.

1.3.5. Особенности программного обеспечения мобильных систем

Так же как мобильные операционные системы, прикладное программное обеспечение для мобильных систем имеет свои особенности, связанные с особенностями самих мобильных устройств.

Разработка и тестирование. Важно учитывать, что разработка программ для мобильных устройств, как правило, производится на обычных настольных компьютерах, на которых функционируют специальные среды разработки (либо обычные среды программирования, позволяющие создавать ПО для планшетов и смартфонов). Соответственно, возникает задача первичного тестирования созданных программ (проверки их работоспособности) ещё до их переноса на «целевое» мобильное устройство. Для тестирования мобильного ПО на «немобильном» настольном ПК обычно используются специальные среды-эмуляторы, имитирующие работу соответствующего мобильного устройства (от его аппаратных особенностей до малого размера и разрешения экрана).

Высокая экономичность по ресурсам. Мобильные устройства (особенно планшетные ПК и смартфоны) обычно слабее по аппаратным характеристикам чем настольные ПК и ноутбуки. Это касается как быстродействия процессора, так и объёмов памяти и характеристик графической подсистемы. Чтобы обеспечить работоспособность и достаточно высокую скорость отклика (реакции программы на действия пользователя), алгоритмы разрабатываемого для мобильных устройств программного обеспечения должны быть оптимизированы как по скорости работы, так и по использованию ресурсов памяти. (Например, при необходимости создания для мобильного устройства программы для определения НОД натуральных чисел следовало бы использовать модифицированный алгоритм Евклида с использованием операции вычисления остатка от деления (MOD), а не классический алгоритм Евклида.)

Максимальное использование флеш-памяти. В связи с ограниченным объёмом оперативной памяти, имеющейся в большинстве мобильных устройств, в них, как правило, предусмотрена возможность расширения памяти за счёт установки карты флеш-памяти. Однако используемое программное обеспечение тоже должно поддерживать работу с картой памяти. Так, если программа должна работать с большими объёмами данных (например, графические данные, фото или видео), то хранение этих данных программа должна производить не в основной оперативной памяти, в которой функционирует сама эта программа, а на карте флеш-памяти.

«Разношёрстность» мобильных устройств и жёсткая привязка к мобильной платформе. Для мобильных устройств используются различные аппаратно-программные платформы (основные из которых — Android, iOS, Windows/Windows Mobile, Symbian,

BlackBerry), программное обеспечение для которых несовместимо между собой. Поэтому для более полного охвата аудитории пользователей мобильных устройств разработчикам ПО приходится создавать версии своих программ для каждой мобильной платформы в отдельности. Следует также учитывать, что версии ОС для мобильных устройств эволюционируют достаточно быстро (например, новые версии Android появляются почти каждый год), и в них появляются принципиально новые функции и возможности. Поэтому разработчики ПО нередко вынуждены создавать отдельные версии программ и под разные версии одной и той же мобильной платформы, а также достаточно часто предлагать своим пользователям установку обновлений ПО.

Существуют также и попытки создания «универсального» (кроссплатформного) ПО, как правило, основанные на использовании созданных индивидуально для каждой мобильной платформы «сред исполнения», обеспечивающих уже в свою очередь работающим в них программам единую для всех платформ среду.

Однако, несмотря на все попытки разработчиков, пользователи мобильных устройств достаточно часто сталкиваются с несовместимостью имеющихся у них устройств с желаемым программным обеспечением: нужная программа есть только для устройства с другим типом ОС или с другой, более новой версией ОС.

Ориентация ПО на непрофессионального пользователя. Мобильными устройствами (планшетами, смартфонами) пользуется очень широкий круг людей, многие из которых очень далеки от «традиционной» компьютерной техники и не умеют работать с «обычными» компьютерами. Поэтому интерфейс программного обеспечения для мобильных устройств должен быть максимально дружелюбным, интуитивно понятным для любого пользователя.

Уже упоминавшееся выше ограничение вычислительных ресурсов мобильных устройств и широта круга их пользователей приводят к тому, что мобильные устройства редко используются для решения сложных узкопрофессиональных задач (для этого специалисты предпочитают использовать настольные компьютеры или мощные ноутбуки). По этим причинам программное обеспечение для мобильных устройств в основном рассчитано на решение повседневных бытовых задач (стратегия «карманного помощника»), на сферу досуга и развлечения.

Особенность тактильного интерфейса. Если пользователи «обычного» компьютера работают преимущественно с клавиатурой и мышью, то для пользователей большинства мобильных

устройств основой интерфейса является тачскрин (чувствительный к нажатию экран, одновременно являющийся и устройством вывода графической информации, и устройством ввода и управления). Соответственно, разрабатываемое для мобильных устройств программное обеспечение должно обладать всей полнотой возможностей для реализации работы пользователя с тачскрином. Кроме того, многие современные мобильные устройства имеют встроенные компас, гироскоп, акселерометр (служащие для определения положения и перемещений устройства в пространстве), а также встроенные системы геопозиционирования. Программное обеспечение должно корректно работать с этими устройствами (что особенно важно для современных программных приложений, поддерживающих работу с 3D-шлемами и реализацию технологий виртуальной и дополненной реальности).

Как правило, для разработчиков прикладных программ разработчики соответствующих новых технологий предоставляют специальные среды программирования (пакеты SDK), в которых уже имеются готовые программные модули для работы с теми или иными аппаратными модулями мобильных устройств и для поддержки тех или иных новых функций.

Объектность и модульность. Для облегчения создания программ для мобильных устройств среды программирования часто предоставляют возможность работы с объектами (к которым могут относиться как элементы пользовательского интерфейса — окна, кнопки и прочее, так и, например, игровые персонажи), для которых достаточно «прописать» желаемый сценарий поведения на условном «псевдоязыке», принятом в данной среде программирования. Далее уже сама среда разработки берёт на себя основную часть работы по формированию требуемых алгоритмов и их реализации для конкретного типа мобильных устройств.

Во многих случаях среда разработки мобильного ПО фактически предоставляет разработчику набор готовых объектов, из которых требуемую программу можно собрать, как из кубиков или из деталей детского конструктора.

Коммуникационная направленность. Вспомним, что для мобильных устройств одной из основных сфер применения является сфера коммуникаций. Причём речь идет не только о простом общении (голосовом или письменном), но и вообще о возможности обмена информацией. Например, программы для фотографирования и видеосъёмки обычно предлагают пользователю возможность сразу же опубликовать полученное фото или видео в социальной сети, коллекции (типа Instagram) и т. д.

Другое направление применения коммуникационных технологий нацелено на компенсацию более слабых аппаратных возможностей мобильных устройств: наиболее сложные, ресурсоёмкие вычисления, обработка и хранение больших объёмов данных всё чаще реализуются удалённо — на специально выделенных интернет-серверах, к которым мобильное устройство обращается как клиент, передавая требуемые данные для обработки и получая готовый результат. Такая стратегия хранения и обработки данных получила название облачных технологий.

Коммерческие особенности ПО для мобильных устройств. Ещё одна особенность программного обеспечения для мобильных устройств связана с политикой их коммерческого распространения. Как правило, такие программы распространяются через специальные сайты — коллекции ПО (магазины приложений), создаваемые и поддерживаемые разработчиками соответствующих устройств, причём программная продукция перед её размещением на таком сайте обязательно должна проходить контроль качества и отсутствия вредоносного кода. Для каждой мобильной платформы обычно предусмотрен отдельный «магазин приложений» (например, Google Play — для Android, AppleStore — для iOS и т. д.). Пользователям предлагаются как платные, так и бесплатные приложения. Обычно требуется заранее выполнить привязку своего устройства к созданному на сайте-«магазине» аккаунту (выбрать для себя логин и пароль). На сайт при этом передаются аппаратные и программные характеристики устройства, что позволяет выполнять автоматический контроль совместимости программ с этим устройством (при просмотре сайта с самого мобильного устройства демонстрируется только совместимое ПО, в отличие от просмотра сайта с обычного настольного компьютера).

Установка выбранного на сайте ПО производится автоматически: необходимо зайти на сайт-«магазин» с требуемого мобильного устройства, выбрать программу и нажать кнопку установки — выполняется скачивание через Интернет дистрибутива программы и сразу же — его установка. Таким образом, пользователь мобильного устройства сам не работает с дистрибутивом, от него не требуется ввода серийного номера, активации ПО или иных действий, подтверждающих его лицензионность: доказательством служит сам факт установки приложения с сайта-«магазина». Впрочем, на мобильных устройствах существует и резервная возможность установки ПО со сторонних сайтов или из отдельно скопированного дистрибутива (архива в специальном формате и с особым расширением имени файла, например для ОС Android

используется снабжённый электронной подписью zip-архив с расширением ark).

При установке платного приложения его стоимость обычно списывается с особого отдельного счёта, привязанного к основному счёту пользователя мобильной телефонной связи. Стоимость программ для мобильных устройств, как правило, невелика.

Во многих случаях разработчики ПО предлагают пользователям мобильных устройств как платные, так и бесплатные версии. При этом бесплатная версия обычно имеет ограниченный набор функций, может демонстрировать во время работы коммерческую рекламу (на что расходуется интернет-трафик) или содержать ссылки на покупку платных приложений (как других, в порядке рекламы, так и частей этого же бесплатного приложения).

Таким образом, основными особенностями мобильных программных приложений являются:

- упрощённость и ориентированность на повседневные задачи, досуг и развлечения;
- простота установки и использования для непрофессионалов;
- ориентация на коммуникационные возможности мобильных устройств;
- особая политика распространения через сайты — «магазины приложений» и значительный объём бесплатного ПО, разработчики которого зарабатывают на рекламе и предоставлении ссылок на платный контент.

1.4. Защита от несанкционированного доступа к информации

1.4.1. Защита с использованием паролей



Для защиты от несанкционированного доступа к программам и данным, хранящимся на компьютере, используются **пароли**. Компьютер разрешает доступ к своим ресурсам только тем пользователям, которые зарегистрированы и ввели правильный пароль. Каждому конкретному пользователю может быть разрешён доступ только к определённым информационным ресурсам. При этом может производиться регистрация всех попыток несанкционированного доступа.

Вход по паролю может быть установлен в программе BIOS Setup: компьютер не начнёт загрузку операционной системы, если не введён правильный пароль (рис. 1.32). Преодолеть такую защиту нелегко, более того, возникнут серьёзные проблемы доступа к данным, если пользователь забудет этот пароль.

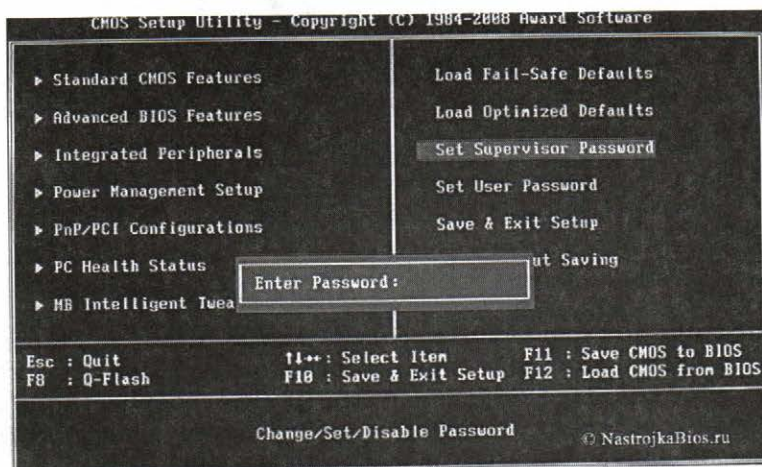


Рис. 1.32

Защита с использованием пароля применяется при загрузке операционной системы (при загрузке системы каждый пользователь должен ввести свой пароль) (рис. 1.33, 1.34).

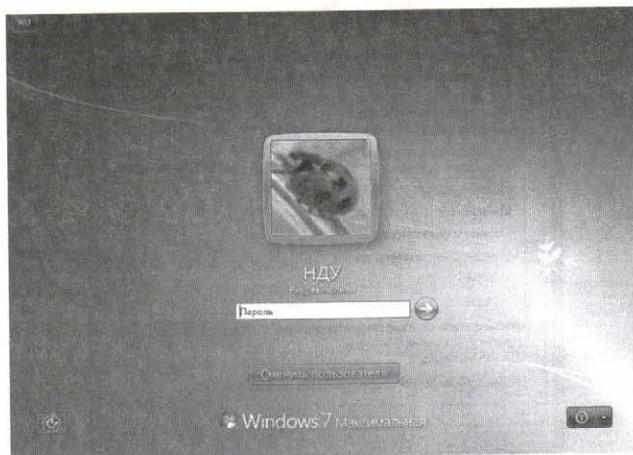


Рис. 1.33



Рис. 1.34

От несанкционированного доступа может быть защищён каждый диск, папка или файл локального компьютера. Для них могут быть установлены определённые права доступа (полный доступ, изменение, чтение, запись и др.), причём права могут быть различными для различных пользователей (рис. 1.35).

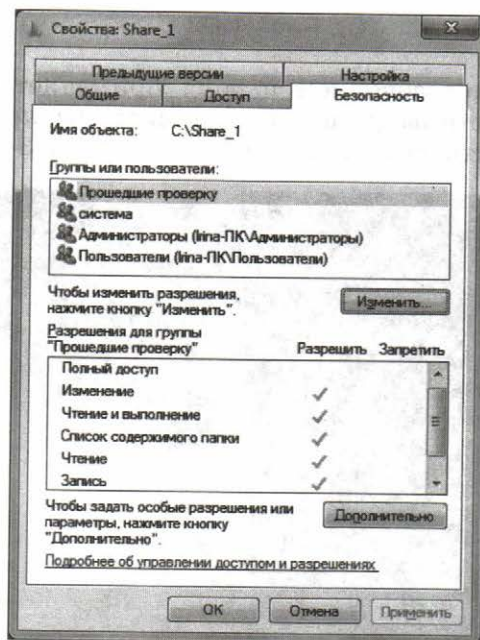


Рис. 1.35



Вопросы и задания

Как защищается информация в компьютере с использованием паролей?

1.4.2. Биометрические системы защиты

В настоящее время для защиты от несанкционированного доступа к информации всё более часто используются **биометрические системы идентификации**. Используемые в этих системах характеристики являются неотъемлемыми качествами личности человека и поэтому не могут быть утерянными и подделанными. К биометрическим системам защиты информации относятся системы идентификации:

- по отпечаткам пальцев;
- по характеристикам речи;
- по радужной оболочке глаза;
- по изображению лица;
- по геометрии ладони руки.

Идентификация по отпечаткам пальцев. Оптические сканеры считывания отпечатков пальцев устанавливаются на ноутбуки, мыши, клавиатуры, флеш-память, а также применяются в виде отдельных внешних устройств и терминалов (например, в аэропортах и банках). На рис. 1.36 показан оптический сканер отпечатка пальца, вмонтированный в ноутбук.

Если узор отпечатка пальца не совпадает с узором допущенного к информации пользователя, то доступ к информации невозможен.



Рис. 1.36

Идентификация по характеристикам речи. Идентификация человека по голосу — один из традиционных способов распознавания, интерес к этому методу связан и с прогнозами внедрения голосовых интерфейсов в операционные системы. Можно легко узнать собеседника по телефону, не видя его. Также можно

определить психологическое состояние по эмоциональной окраске голоса. Голосовая идентификация бесконтактна, и существуют системы ограничения доступа к информации на основании частотного анализа речи (рис. 1.37).

Каждому человеку присуща индивидуальная частотная характеристика каждого звука (фонемы).

В романе А. И. Солженицына «В круге первом» описана голосовая идентификация человека ещё в 40-е годы прошлого века.

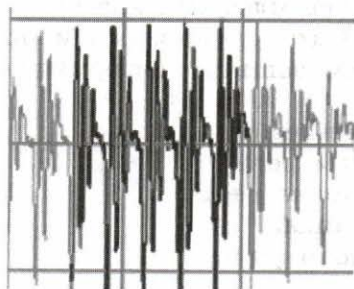


Рис. 1.37

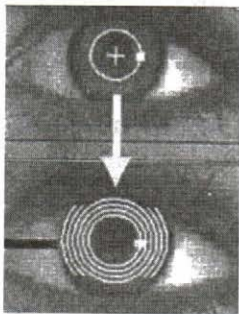


Рис. 1.38

Идентификация по радужной оболочке глаза. Радужная оболочка глаза является уникальной для каждого человека биометрической характеристикой. Она формируется в первые полтора года жизни и остаётся практически неизменной в течение всей жизни.

Изображение глаза выделяется из изображения лица, и на него накладывается специальная маска штрих-кодов (рис. 1.38). Результатом является матрица, индивидуальная для каждого человека.

Для идентификации по радужной оболочке глаза применяются специальные сканеры, подключённые к компьютеру.

Идентификация по изображению лица. Для идентификации личности часто используется технологии распознавания по лицу. Они ненавязчивы, так как распознавание человека происходит на расстоянии, без задержек и отвлечения внимания и не ограничивают пользователя в свободе перемещений.

По лицу человека можно узнать его историю, симпатии и антипатии, болезни, эмоциональное состояние, чувства и намерения по отношению к окружающим. Всё это представляет особый интерес для автоматического распознавания лиц (например, для выявления потенциальных преступников).

Идентификационные признаки учитывают форму лица, его цвет, а также цвет волос. К важным признакам можно отнести также координаты точек лица в местах, соответствующих смене контраста (брови, глаза, нос, уши, рот и овал).

С 2009 года началась выдача новых загранпаспортов, в микросхеме которых хранится цифровая информация о владельце.

Идентификация по ладони руки. В биометрике в целях идентификации используется простая геометрия руки — размеры и форма, а также некоторые информационные знаки на тыльной стороне руки (образы на сгибах между фалангами пальцев, узоры расположения кровеносных сосудов).

Сканеры идентификации по ладони руки установлены в некоторых аэропортах, банках и на атомных электростанциях (рис. 1.39).



Рис. 1.39

Вопросы и задания

Какие существуют биометрические методы защиты информации?

Практическая работа 1.7

Биометрическая защита:

идентификация по характеристикам речи

Задание. В операционной системе Windows или Linux с помощью звукового редактора Audacity записать одно и то же слово (например, «информатика»), произнесённое несколькими людьми. По частотной характеристике речи попробовать идентифицировать этих людей.

Варианты выполнения работы:

- в операционной системе Windows или в операционной системе Linux;
- записать и проанализировать частотные характеристики речи разных людей.





Запись с помощью звукового редактора Audacity одного и того же слова, произнесённого несколькими людьми. Идентификация этих людей по частотной характеристике речи



1. В операционной системе Windows или Linux запустить звуковой редактор Audacity.
2. Записать одно и то же слово (например, «информатика»), произнесённое несколькими людьми.
3. Проанализировать частотные характеристики записей и попробовать идентифицировать этих людей по темпу речи и характерным особенностям произношения звуков (фоном) (рис. 1.40).

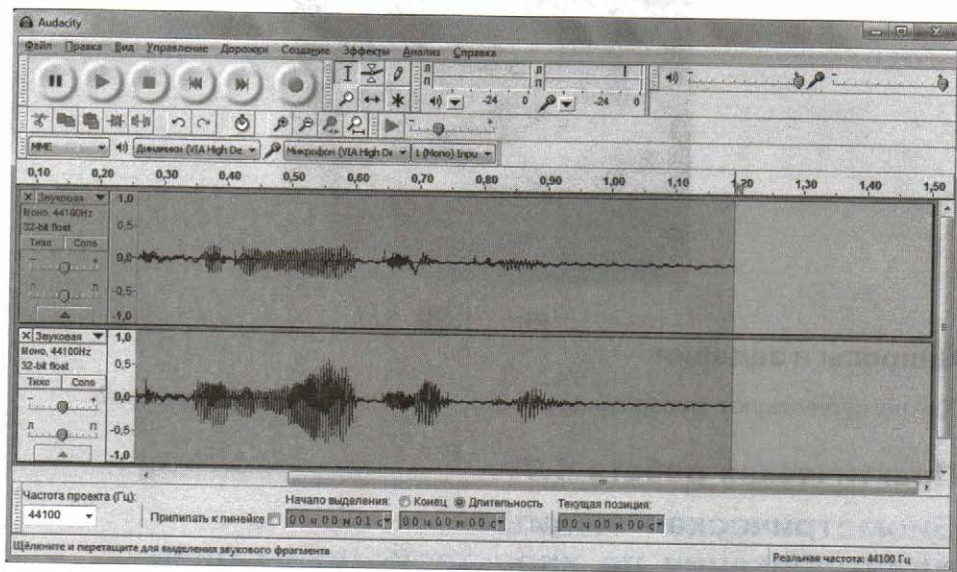


Рис. 1.40

1.5. Физическая защита данных на дисках

Для обеспечения большей скорости чтения/записи и надежности хранения данных на жёстких дисках используются RAID-массивы (англ. *Redundant Arrays of Independent Disks* — избыточный массив независимых дисков). Несколько жёстких дисков подключаются к **RAID-контроллеру**, который рассматривает их как единый логический носитель информации.

Существует два способа реализации RAID-массива: аппаратный и программный. Аппаратный дисковый массив состоит из нескольких жёстких дисков, управляемых при помощи специальной платы контроллера RAID-массива. Программный RAID-массив реализуется при помощи специального драйвера. В программный массив организуются дисковые разделы, которые могут занимать как весь диск, так и его часть. Программные RAID-массивы, как правило, менее надёжны, чем аппаратные, но обеспечивают более высокую скорость работы с данными.

Существует несколько разновидностей RAID-массивов, так называемых уровней. Операционные системы поддерживают несколько уровней RAID-массивов.

RAID 0. Для создания массива этого уровня понадобится как минимум два диска одинакового размера. Запись осуществляется по принципу чередования: данные делятся на порции одинакового размера (A1, A2, A3 и т. д.) и поочередно распределяются по всем дискам, входящим в массив (рис. 1.41). Поскольку запись ведётся на все диски, при отказе одного из них будут утрачены все хранившиеся на массиве данные, однако запись и чтение на разных дисках происходят параллельно и, соответственно, быстрее.

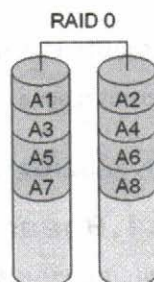


Рис. 1.41

RAID 1. Массивы этого уровня построены по принципу зеркалирования, при котором все порции данных (A1, A2, A3 и т. д.), записанные на одном диске, дублируются на другом (рис. 1.42). Для создания такого массива потребуется два или более дисков одинакового размера. Избыточность обеспечивает отказоустойчивость массива: в случае выхода из строя одного из дисков данные на другом остаются неповреждёнными. Расплата за надёжность — фактическое сокращение дискового пространства вдвое. Скорость чтения и записи остается на уровне обычного жёсткого диска.

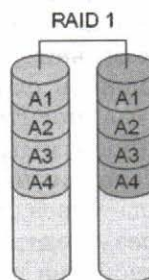


Рис. 1.42

Существуют и другие уровни RAID-массивов — с номерами 2, 3, 4, 5 и 6. Кроме того, возможно создание двухступенчатых RAID-массивов: например, для ускорения доступа к данным при сохранении высокой отказоустойчивости системы можно собрать RAID-массив уровня 0 из нескольких RAID-массивов уровня 1. В подобных случаях номера уровней RAID-массивов будут дву-

значными, например вышеупомянутая система (массив RAID 0, построенный из массивов RAID 1) будет именоваться RAID 10.



Самостоятельно разберите, что означают такие обозначения, как RAID 01, RAID 50, RAID 05, RAID 06 и RAID 60.

Все перечисленные обозначения RAID-массивов были предложены Калифорнийским университетом и считаются негласным стандартом. Кроме них, также существуют нестандартные уровни: RAID-Z (модификация уровня RAID 5), RAID-7, RAID 7.3 и RAID-DP (три последних варианта — модификации, предложенные различными фирмами-изготовителями компьютерных систем).



Вопросы и задания

Какие существуют массивы дисков RAID?

1.6. Защита от вредоносных программ

1.6.1. Вредоносные и антивирусные программы

Типы вредоносных программ. Вредоносные программы — это программы, наносящие вред данным и программам, хранящимся на компьютере. Существующие вредоносные программы подразделяются на следующие разновидности (рис. 1.43). Среди них основными являются:

- **компьютерный вирус** — вредоносная программа, способная заражать файлы в пределах одного компьютера, а также съёмные носители информации, записывая свои копии в другие программы;
- **сетевые черви** — программы, способные заражать компьютеры через локальные и/или глобальные сети, в том числе через электронную почту. Некоторые сетевые черви могут иметь свойства троянской программы и/или компьютерного вируса;
- **троянские программы** — программы, которые незаметно для пользователя осуществляют сбор конфиденциальной информации (например, паролей доступа) и её передачу злоумышленнику, разрушение или злонамеренную модификацию информации на компьютере, нарушение работоспособности компьютера или использование его ресурсов в вредоносных целях (например, для рассылки спама или атак на серверы Интернета);
- **вредоносные утилиты** — программы, создающие угрозу информационной безопасности: различные «конструкторы вирусов», программы для организации удалённых атак или

«взлома» других компьютеров, утилиты для рассылки спама по электронной почте и в виде SMS-сообщений, «хакерские» утилиты (например, для раскрытия паролей архивов), программы-блокираторы и т. д.



Рис. 1.43

Первыми вредоносными программами, созданными для компьютеров, были компьютерные вирусы. Поэтому «компьютерными вирусами» часто называют все вредоносные программы.

За создание, использование и распространение вредоносных программ в России и большинстве стран предусмотрена уголовная ответственность.

Антивирусные программы. Современные антивирусные программы обеспечивают комплексную защиту программ и данных на компьютере от всех типов вредоносных программ и методов их проникновения на компьютер (Интернет, локальная сеть, электронная почта и съёмные носители информации). Популярные антивирусные программы: avast!, Avira, Dr. Web, ESET NOD32, Kaspersky. Для защиты от вредоносных программ каждого типа в антивирусе предусмотрены отдельные компоненты.

Принцип работы антивирусных программ основан на проверке файлов, загрузочных секторов дисков и оперативной памяти и поиске в них известных и новых вредоносных программ.

Следует отметить недопустимость использования на одном и том же компьютере двух разных антивирусных программ.

В настоящее время в антивирусных программах используются две основные технологии обнаружения вирусов: анализ сигнатур и вероятностный анализ.





Для поиска известных вредоносных программ используются сигнатуры. **Сигнатура** — это некоторая постоянная последовательность программного кода, специфичная для конкретной вредоносной программы. Если антивирусная программа обнаружит такую последовательность в каком-либо файле, то этот файл считается заражённым вирусом и подлежит лечению или удалению.



Сигнатуры, известные на данный момент времени разработчикам антивирусных программ, содержатся в особом файле (или группе файлов) — **антивирусной базе данных**. Для успешного обнаружения вредоносных программ необходимо, чтобы антивирусная база данных была самой свежей. Поэтому необходимо регулярно выполнять обновление установленной на компьютере антивирусной программы для получения со специально созданного разработчиками этой программы сайта последней версии антивирусной базы данных. Иначе вредоносные программы, появившиеся позже, чем была подготовлена имеющаяся на компьютере антивирусная база, не будут обнаружены антивирусной программой (так как их сигнатуры в антивирусной базе отсутствуют).

Преимуществом анализа сигнатур является однозначность обнаружения и определения вирусов. Но для этого из нового вируса сначала должна быть выделена его сигнатура и помещена в антивирусную базу. Поэтому такой метод позволяет обнаруживать только уже известные вирусы, а новый вирус, ещё не занесённый в антивирусную базу (либо если на данном компьютере антивирусная база не была своевременно обновлена), обнаружен не будет.

Для обнаружения принципиально новых вирусов может использоваться **вероятностный анализ**. При этом возможны три основные его разновидности: эвристический анализ, поведенческий анализ и анализ контрольных сумм.

Эвристический анализ основан на специальных алгоритмах, позволяющих с определённой вероятностью выявлять подозрительные файлы. В процессе такого анализа проверяется структура файла и его соответствие ранее выявленным особенностям программного кода вирусов («вирусным шаблонам»). Недостатками эвристического анализа являются возможность ложных срабатываний (когда незаражённый файл опознается как заражённый) и возможность пропуска вирусов. Поэтому при использовании вероятностного метода антивирусная программа только выдаёт сообщение о подозрении на наличие вируса, но его удаление при этом невозможно.

Поведенческий анализ позволяет выявлять подозрительные программные объекты на основе анализа выполняемых ими действий. Поведенческий анализатор может отслеживать попытки

прямого доступа к файлам, форматирования жёстких дисков без запроса разрешения у пользователя, удаления всех данных с диска, изменения загрузочной записи без соответствующих уведомлений и пр. Поведенческие анализаторы могут выявлять неизвестные на данный момент вирусы, но (по тем же причинам, что и для эвристического анализа) дают возможность лишь с какой-то вероятностью заподозрить наличие вирусов.

Анализ контрольных сумм обеспечивает возможность отслеживать изменения в файлах. На основании анализа характера таких изменений (их одновременности, массовости, идентичности изменения длин файлов) можно сделать вывод о возможном заражении компьютера. Анализаторы контрольных сумм («ревизоры») тоже позволяют только с некоторой вероятностью заподозрить наличие вируса. Сегодня эта технология обычно применяется в антивирусных сканерах: при первой проверке с файлов (считающихся заведомо незаражёнными) снимаются контрольные суммы и составляется таблица их значений, а при всех последующих проверках реальные контрольные суммы файлов сравниваются с табличными, и при обнаружении их несоответствия пользователю выдаётся предупреждение о возможном заражении.

В настоящее время практически любой антивирус использует сразу несколько технологий из списка перечисленных выше, причём использование сигнатурного и эвристического анализа является обязательным.

Большинство антивирусных программ сочетает в себе функции постоянной защиты (антивирусный монитор) и функции защиты по требованию пользователя (антивирусный сканер).

Антивирусный монитор запускается автоматически при старте операционной системы и работает в качестве фонового системного процесса, проверяя на вредоносность совершаемые другими программами действия. Основная задача антивирусного монитора состоит в обеспечении максимальной защиты от вредоносных программ при минимальном замедлении работы компьютера.

Антивирусный сканер запускается пользователем по заранее выбранному расписанию или в произвольный момент. Антивирусный сканер производит поиск вредоносных программ в оперативной памяти, а также на жёстких и сетевых дисках компьютера.

К недостаткам антивирусных программ можно отнести большие размеры используемых ими антивирусных баз данных, которые должны содержать информацию о максимально возможном количестве вирусов, что, в свою очередь, приводит к относительно небольшой скорости поиска вирусов.

Признаки заражения компьютера. Есть ряд признаков, свидетельствующих о проникновении на компьютер вредоносных программ:

- вывод на экран непредусмотренных сообщений или изображений;
- подача непредусмотренных звуковых сигналов;
- произвольный запуск на компьютере каких-либо программ;
- частые «зависания» и сбои в работе компьютера;
- медленная работа компьютера при запуске программ;
- исчезновение или изменение файлов и папок;
- частое обращение к жёсткому диску (часто мигает лампочка на системном блоке);
- зависание или неожиданное поведение браузера (например, окно программы невозможно закрыть).

Кроме того, есть некоторые характерные **признаки поражения сетевым вирусом через электронную почту:**

- друзья или знакомые говорят о полученных от вас сообщениях, которые вы не отправляли;
- в вашем почтовом ящике находится большое количество сообщений без обратного адреса и заголовка.

Действия при наличии признаков заражения компьютера. Прежде чем предпринимать какие-либо действия, необходимо сохранить результаты работы на внешнем носителе (внешнем жёстком диске, CD- или DVD-диске, флеш-памяти и пр.). Далее необходимо:

- отключить компьютер от локальной сети и Интернета, если он к ним был подключён;
- если симптом заражения состоит в том, что невозможно загрузиться с жёсткого диска компьютера (компьютер выдаёт ошибку, когда вы его включаете), то попробовать загрузиться в режиме защиты от сбоев или с диска аварийной загрузки Windows;
- запустить антивирусную программу.

Вопросы и задания

1. В чём различие между антивирусными сканерами и мониторами?
2. Какие существуют признаки заражения компьютера вирусом?
3. Что необходимо сделать в первую очередь в случае заражения компьютера вирусом?

1.6.2. Компьютерные вирусы и защита от них

Обязательным свойством компьютерного вируса является способность к «размножению» (самокопированию). Вирусы также могут

незаметно для пользователя внедряться в файлы, загрузочные секторы дисков и документы. Название «вирус» подобные компьютерные программы получили из биологии именно по признаку способности к саморазмножению.

После заражения компьютера вирус может активизироваться и заставить компьютер выполнять какие-либо действия. Активизация вируса может быть связана с различными событиями (наступление определенной даты или дня недели, запуск программы, открытие документа и т. д.).

Компьютерные вирусы являются вредоносными программами, которые могут «размножаться» и скрытно внедрять свои копии в файлы, загрузочные секторы дисков и документы. Активизация компьютерного вируса может вызывать уничтожение программ и данных.



По величине вредных воздействий вирусы можно разделить на:

- неопасные, влияние которых ограничивается уменьшением свободной памяти на диске, графическими, звуковыми и другими внешними эффектами;
- опасные, которые могут привести к сбоям и «зависаниям» при работе компьютера;
- очень опасные, активизация которых может привести к потере программ и данных (изменению или удалению файлов и каталогов), форматированию жёсткого диска и т. д.

По «среде обитания» вирусы можно разделить на загрузочные, файловые, макровирусы, скриптовые и комбинированные.

Загрузочные вирусы. Загрузочные вирусы заражают загрузочный сектор жёсткого диска или носителя на основе флеш-памяти. Принцип действия загрузочных вирусов основан на алгоритмах запуска операционной системы при включении или перезагрузке компьютера. После необходимых тестов установленного оборудования программа системной загрузки считывает первый физический сектор загрузочного диска (жёсткого, оптического или флеш-памяти в зависимости от параметров, установленных в BIOS Setup) и передаёт на него управление.

При заражении дисков загрузочные вирусы «подставляют» свой код вместо программы, получающей управление при загрузке системы, и отдают управление не оригинальному коду загрузчика, а коду вируса. При инфицировании диска вирус в большинстве случаев переносит оригинальный загрузочный сектор в какой-либо другой сектор диска (например, в первый свободный).



Первая эпидемия загрузочного компьютерного вируса произошла в 1986 году, когда вирус Brain заражал загрузочный сектор дискет для персональных компьютеров. Вирус Brain являлся также и первым вирусом-невидимкой, так как при попытке обнаружения заражённого загрузочного сектора вирус незаметно «подставлял» его незаражённый оригинал.

Профилактическая защита от таких вирусов состоит в отказе от загрузки операционной системы с внешних носителей (жёстких дисков, CD- или DVD-дисков, флеш-памяти и пр.) и установке в BIOS вашего компьютера защиты загрузочного сектора от изменений. С помощью программы BIOS Setup можно провести настройку BIOS таким образом, что будет запрещена (заблокирована) любая запись в загрузочный сектор диска, и компьютер будет защищён от заражения загрузочными вирусами.

Файловые вирусы. Файловые вирусы различными способами внедряются в исполняемые файлы и обычно активизируются при их запуске. После запуска заражённого файла вирус находится в оперативной памяти компьютера и является активным (т. е. может заражать другие файлы) вплоть до момента выключения компьютера или перезагрузки операционной системы.

По способу заражения файловые вирусы разделяют на:

- перезаписывающие вирусы, которые записывают свой код вместо кода программы, не изменяя названия исполняемого файла. При запуске программы выполняется код вируса, а не сама программа;
- вирусы-компаньоны, которые, как и перезаписывающие вирусы, создают свою копию на месте заражаемой программы, но в отличие от перезаписываемых не уничтожают оригинальный файл, а переименовывают или перемещают его. При запуске программы вначале выполняется код вируса, а затем управление передается оригинальной программе;
- паразитические вирусы — это файловые вирусы, изменяющие содержимое файла, добавляя в него свой код. Код может внедряться в начало, середину или конец программы и выполняется перед, вместе или после программы. При этом заражённая программа сохраняет полную или частичную работоспособность.



В 1999 году произошла эпидемия очень опасного файлового вируса Win95.CIH, названного «Чернобыль» из-за даты активации 26 апреля. Вирус уничтожал данные на жёстком диске, а на системных платах стирал содержимое BIOS, что приводило к необходимости их замены.

Практически все загрузочные и файловые вирусы **резидентны**, т. е. они находятся в оперативной памяти компьютера и в про-

цессе работы пользователя могут осуществлять опасные действия (стирать данные на дисках, изменять названия и другие атрибуты файлов и т. д.). Лечение от резидентных вирусов затруднено, так как даже после удаления заражённых файлов с дисков вирус остается в оперативной памяти и возможно повторное заражение файлов.

Профилактическая защита от файловых вирусов состоит в том, что не рекомендуется запускать на исполнение файлы, полученные из сомнительных источников и предварительно не проверенные антивирусными программами.

Макровирусы. Существуют макровирусы для интегрированного офисного приложения Microsoft Office (Word, Excel, PowerPoint и Access). Макровирусы фактически являются макрокомандами (макросами) на встроенном языке программирования Visual Basic for Applications (VBA), которые помещаются в документ.

При работе с документом пользователь выполняет различные действия: открывает документ, сохраняет, печатает, закрывает и т. д. При этом приложение ищет и выполняет соответствующие стандартные макросы. Макровирусы содержат стандартные макросы, вызываются вместо них и заражают каждый открываемый или сохраняемый документ.

Макровирусы являются **ограниченно резидентными**, т. е. они находятся в оперативной памяти и заражают документы, пока открыто приложение. Кроме того, макровирусы заражают шаблоны документов и поэтому активизируются уже при запуске заражённого приложения.

Профилактическая защита от макровирусов состоит в предотвращении запуска вируса. При открытии документа в приложениях Microsoft Office сообщается о присутствии в них макросов (потенциальных вирусов) и предлагается запретить их загрузку. Выбор запрета на загрузку макросов надёжно защитит ваш компьютер от заражения макровирусами, однако отключит и полезные макросы, содержащиеся в документе.

Скриптовые вирусы представляют собой вредоносные программы на скриптовых языках программирования (например, JavaScript), встроенные в html-код веб-страниц.

Комбинированные вирусы обладают свойствами нескольких вышеперечисленных разновидностей (пример — файлово-загрузочные вирусы).

Современные компьютерные вирусы снабжены специальными средствами для защиты от обнаружения и уничтожения антивирусными программами. Так, **полиморфизм** вирусов заключается в том, что при каждом заражении создаваемая копия вируса имеет иной порядок команд своей программы, хотя алгоритм работы

вируса остаётся неизменным. В результате два образца одного и того же вируса могут не иметь практически ни одного совпадающего участка, что затрудняет их обнаружение и опознавание. Для этой же цели создатели вирусов используют шифрование программного кода вируса.



Вопросы и задания

1. Каковы характерные особенности компьютерных вирусов как типа вредоносных программ?
2. Какие существуют типы компьютерных вирусов?



Практическая работа 1.8

Защита от компьютерных вирусов

Задание 1. В операционной системе Windows лечить и удалять файловые вирусы в режиме реального времени.

Задание 2. В операционной системе Linux лечить и удалять файловые вирусы в режиме реального времени.

Задание 3. В операционной системе Windows лечить и удалять файловые вирусы на выбранных носителях информации при признаках заражения.

Задание 4. В операционной системе Linux лечить и удалять файловые вирусы в выбранных папках при признаках заражения.

Варианты выполнения работы:

- в операционной системе Windows или Linux.



Защита от файловых вирусов в режиме реального времени с использованием файлового монитора Файловый Антивирус, входящего в Kaspersky Antivirus

С помощью антивирусной программы Kaspersky Antivirus проверим компьютер на наличие вирусов и при их обнаружении вылечим или удалим заражённые файлы.

Файловый Антивирус контролирует файловую систему компьютера в реальном времени (файловый монитор). Он проверяет все открываемые, запускаемые и сохраняемые файлы на вашем компьютере и всех присоединённых дисках. Каждое обращение к файлу перехватывается приложением, и файл проверяется на присутствие известных вирусов. Дальнейшая работа с файлом возможна только в том случае, если файл не заражён или был успешно вылечен программой Kaspersky Antivirus. Если

же файл по каким-либо причинам невозможно вылечить, он будет удалён.

Прежде всего необходимо через Интернет обновить саму антивирусную программу и вирусную базу данных.

1. В операционной системе Windows соединиться с Интернетом. На *Панели задач* в контекстном меню значка антивирусной программы Kaspersky Antivirus выбрать пункт *Обновление*. В появившемся диалоговом окне будет отображаться процесс обновления антивирусной программы и вирусной базы данных¹⁾ (рис. 1.44).

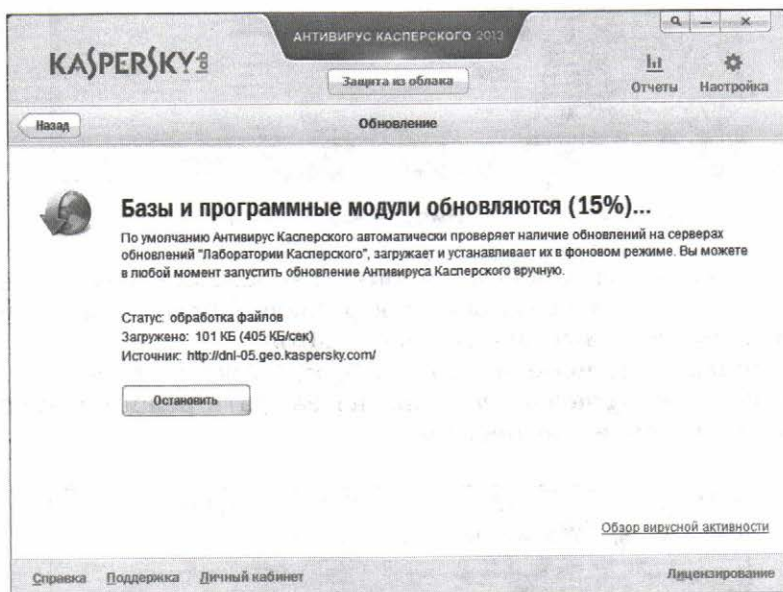


Рис. 1.44

Настроим параметры антивирусного монитора (*Файловый Антивирус*).

2. В контекстном меню значка антивирусной программы выбрать пункт *Настройка*. В меню левой части появившегося диалогового окна выбрать *Файловый Антивирус*. В правой части окна щёлкнуть по кнопке *Настройка...* (рис. 1.45).

¹⁾ В зависимости от используемой версии антивирусной программы вид диалоговых окон в этой и других практических работах может различаться, но смысл имеющихся на них надписей и диалоговых элементов (кнопок, меню и пр.) остаётся одним и тем же.

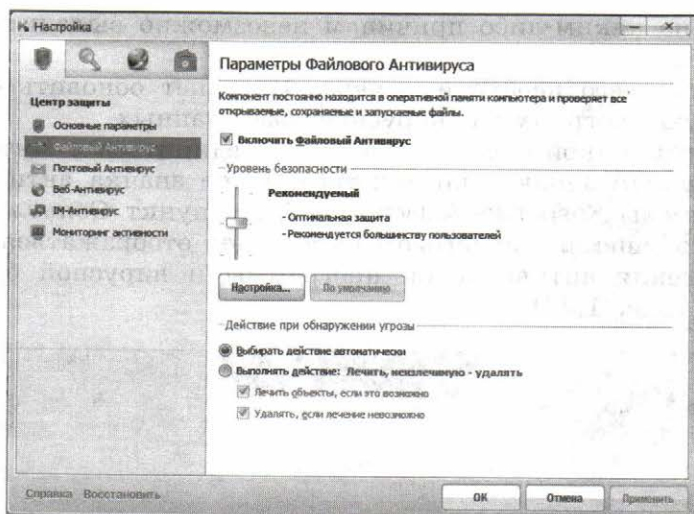


Рис. 1.45

3. В появившемся диалоговом окне на вкладке *Общие* выбрать типы проверяемых файлов и в разделе *Область защиты* выбрать проверяемые диски (рис. 1.46).
 На вкладке *Дополнительно* выбрать режим проверки.
 На вкладке *Производительность* выбрать режим использования и проверяемые носители.

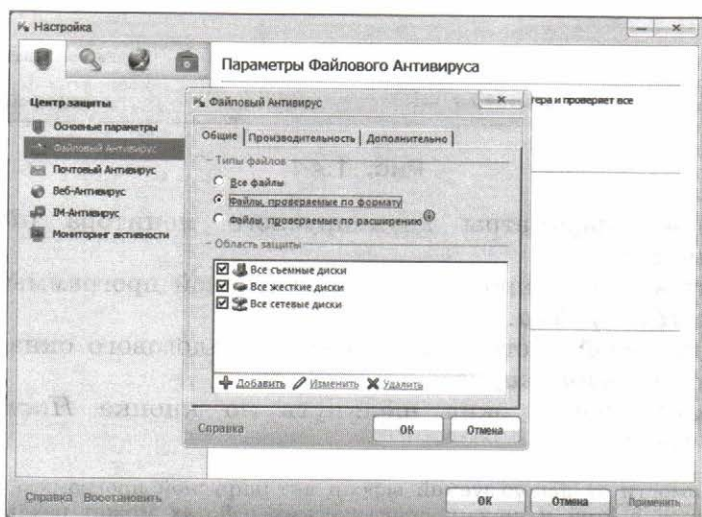


Рис. 1.46

4. Антивирусный монитор (*Файловый Антивирус*) запускается автоматически при старте операционной системы и работает в качестве фонового системного процесса, проверяя на вредоносность совершаемые другими программами действия. Активизировать ссылку *Открыть отчёт*. Появится информационное окно с результатами проверки (рис. 1.47).

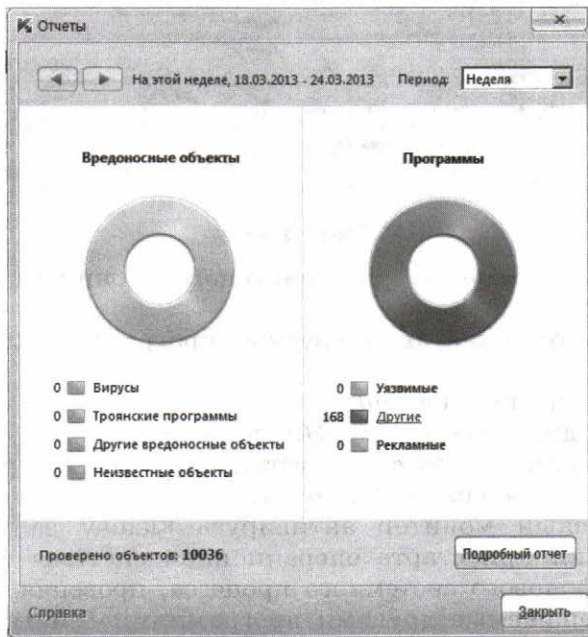


Рис. 1.47

Защита от файловых вирусов в режиме реального времени с использованием файлового монитора, входящего в антивирус KlamAV



Прежде всего, необходимо через Интернет обновить саму антивирусную программу и вирусную базу данных.

1. В операционной системе Linux соединиться с Интернетом.

Запустить антивирус KlamAV и в диалоговом окне приложения выбрать пункт *Обновить*.

В появившемся диалоговом окне будет отображаться процесс обновления антивирусной программы и вирусной базы данных (рис. 1.48).

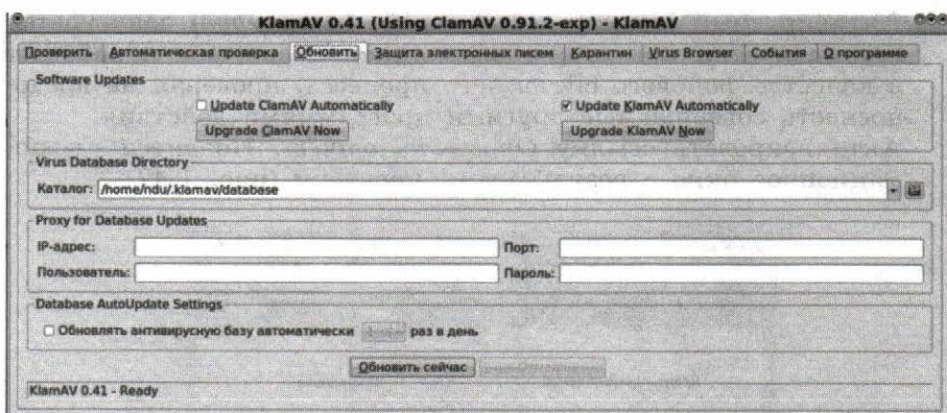


Рис. 1.48

Настроим параметры антивирусного монитора антивируса KlamAV.

2. В диалоговом окне антивирусной программы выбрать пункт *События*.

Щёлкнуть по кнопке *Options*.

Появится диалоговое окно *Настройка*.

Выбрать пункт *Auto-Scan* и установить режим проверки файлов в реальном времени (рис. 1.49).

3. Антивирусный монитор антивируса KlamAV запускается автоматически при старте операционной системы и работает в качестве фонового системного процесса, проверяя на вредоносность совершаемые другими программами действия.

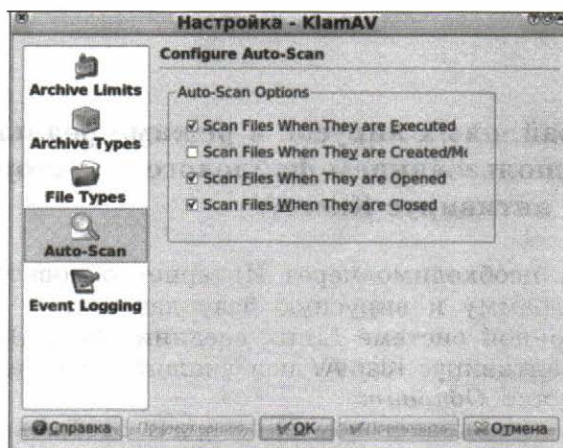


Рис. 1.49



Поиск, лечение или удаление файловых вирусов в Windows

Крайне важно периодически проводить проверку вашего компьютера на присутствие вирусов. Это необходимо делать для того, чтобы исключить возможность распространения вирусов, которые не были обнаружены компонентами постоянной защиты в реальном времени.

Специалистами *Лаборатории Касперского* выделены задачи проверки на вирусы и поиска уязвимостей.

- **Полная проверка** — тщательная проверка всей системы. По умолчанию проверяются следующие объекты: системная память; объекты, исполняемые при старте системы; резервное хранилище системы; почтовые базы; жёсткие, съёмные и сетевые диски.
- **Проверка важных областей** — проверка тех объектов, загрузка которых осуществляется при старте операционной системы, а также проверка системной памяти, загрузочных секторов диска.
- **Поиск уязвимостей** — устанавливает наличие в программном обеспечении уязвимостей, которые могут быть использованы злоумышленниками для распространения вредоносных объектов и доступа к персональным данным.
- **Выборочная проверка** — проверка любого выбранного пользователем объекта файловой системы компьютера.

Проведём проверку на вирусы выбранных областей хранения программ и данных.

1. В контекстном меню значка антивирусной программы выбрать пункт *Настройка*.

В меню левой части появившегося диалогового окна выбрать *Основные параметры* (рис. 1.50). Просмотреть установленные параметры.

2. Выбрать пункт *Полная проверка*.

В правой части окна нажать кнопку *Объекты для проверки...*, и в появившемся диалоговом окне просмотреть список проверяемых объектов компьютера (рис. 1.51).

Закрыть диалоговое окно *Объекты для проверки*.

3. Выбрать пункт *Проверка важных областей*.

В правой части окна нажать кнопку *Объекты для проверки...*, и в появившемся диалоговом окне просмотреть список проверяемых объектов компьютера (рис. 1.52).

Закрыть диалоговое окно *Объекты для проверки*.

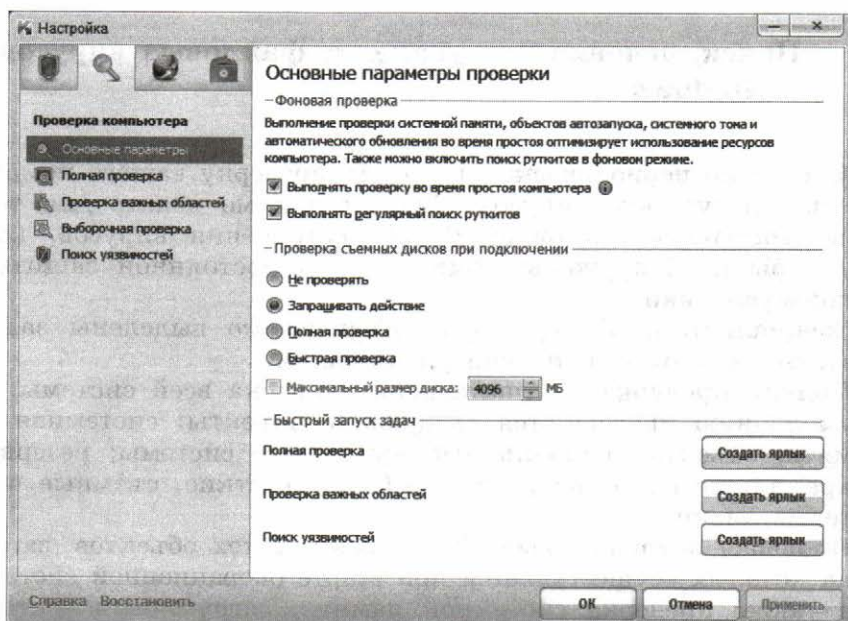


Рис. 1.50

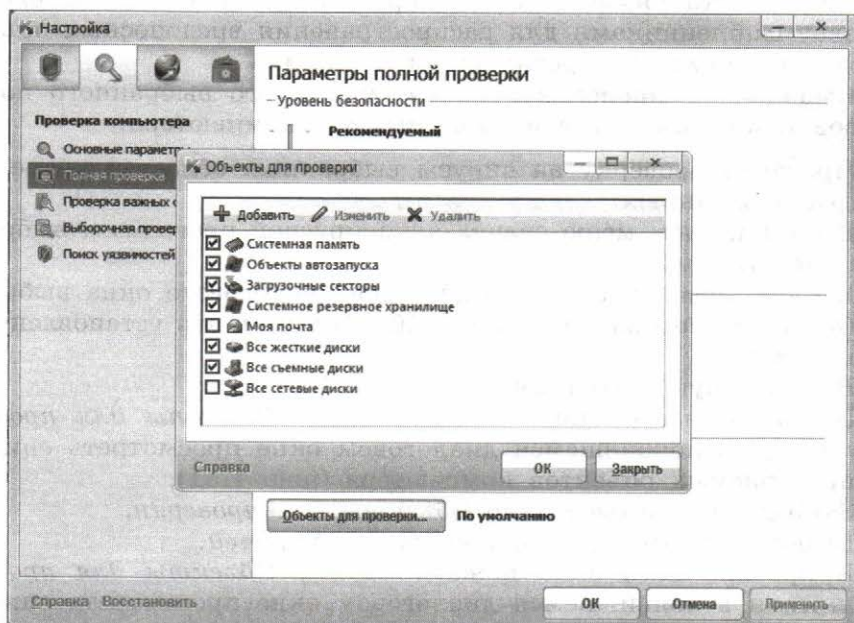


Рис. 1.51

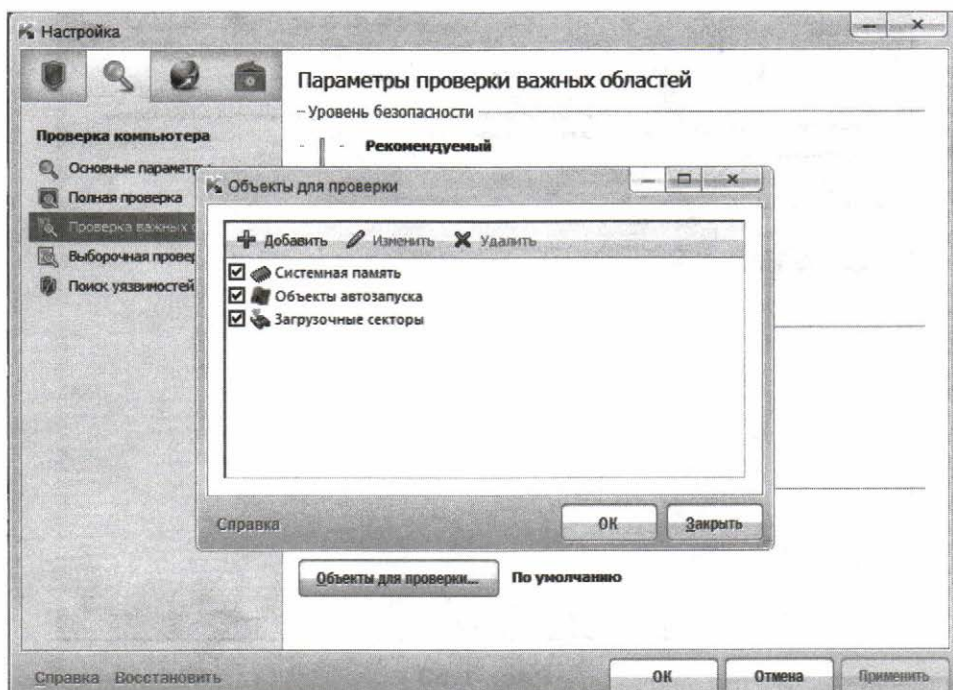


Рис. 1.52

4. Выбрать пункт *Поиск уязвимостей*.

В правой части окна нажать кнопку *Объекты для проверки...*, и в появившемся диалоговом окне просмотреть список проверяемых объектов компьютера (рис. 1.53).

Заккрыть диалоговое окно *Объекты для проверки*.

5. Выбрать пункт *Выборочная проверка*.

В правой части окна нажать кнопку *Настройка...*, и в появившемся диалоговом окне посмотреть список проверяемых объектов компьютера (рис. 1.54).

Заккрыть диалоговое окно *Выборочная проверка*.

6. Запустить проверку выбранной области хранения программ и данных (например, *Проверка важных областей*).

В этом же окне можно увидеть, сколько файлов проверено, сколько вирусов обнаружено (если они есть) и другую информацию (рис. 1.55).

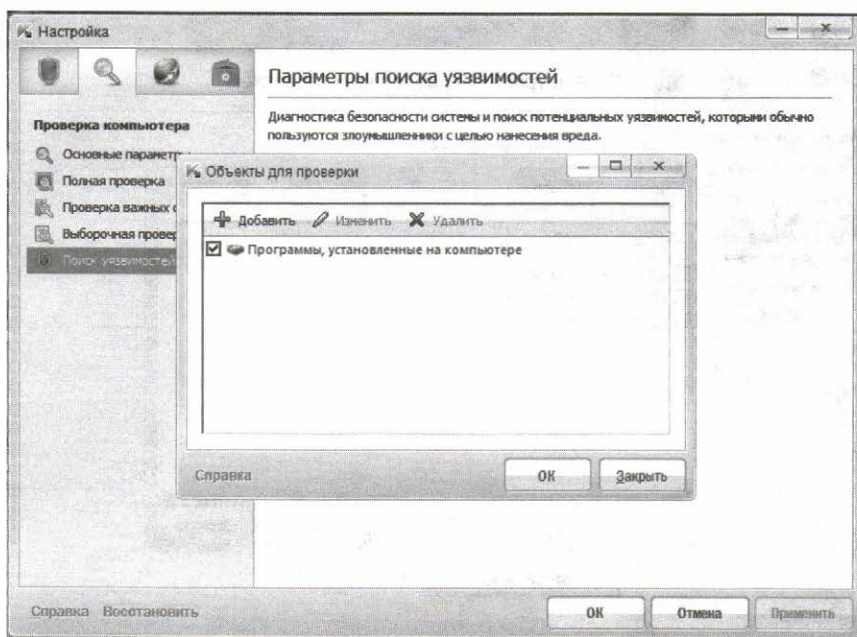


Рис. 1.53

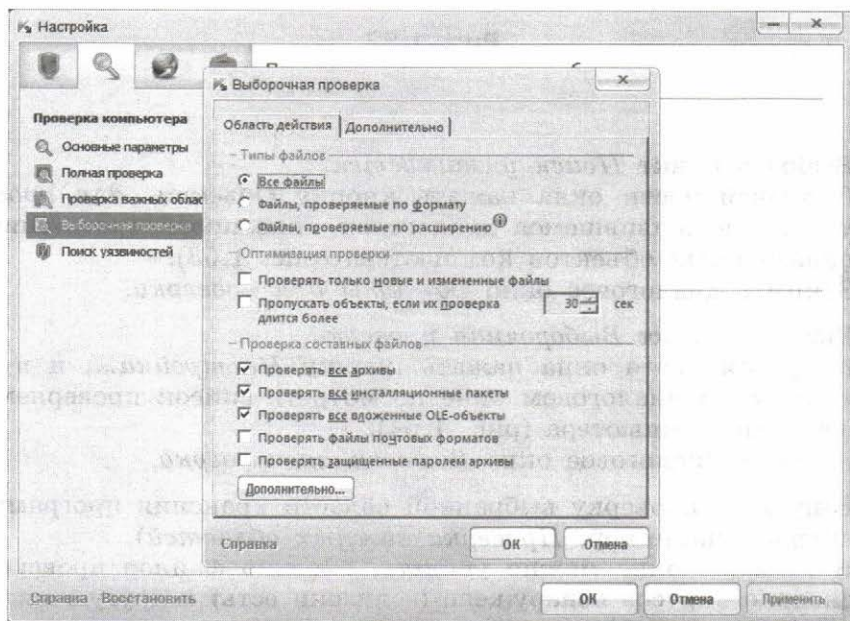


Рис. 1.54

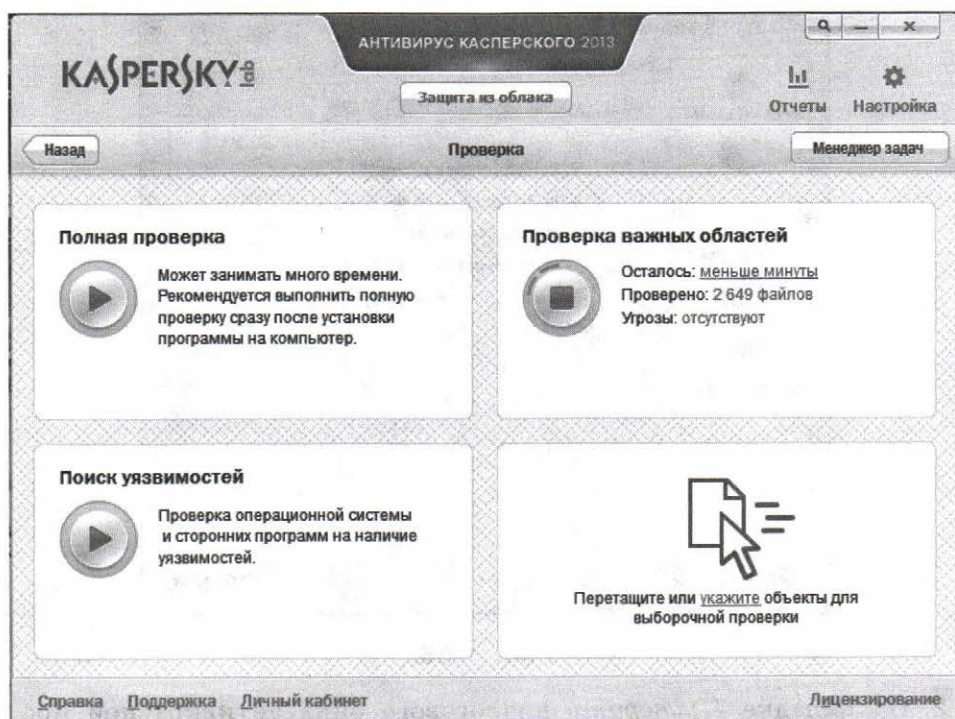


Рис. 1.55

Поиск, лечение или удаление файловых вирусов в Linux



В операционной системе Linux для поиска вирусов в состав антивируса KlamAV включён сканер, который проверяет на наличие вирусов выбранные пользователем папки.

1. В диалоговом окне антивирусной программы выбрать пункт *События*.

Щёлкнуть по кнопке *Options*.

Появится диалоговое окно *Настройка* (рис. 1.56).

Выбрать для проверки исполняемые файлы (пункт *Scan 'Portable Executable' Files*) и файлы документов, которые могут содержать макровирусы (пункт *Scan the Macros in Microsoft Office Files*).

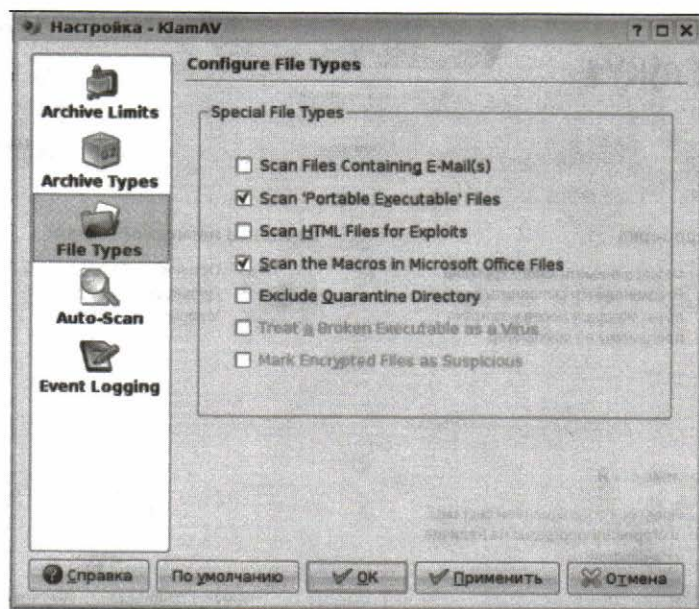


Рис. 1.56

2. На вкладке *Проверить* диалогового окна антивирусной программы выбрать папки для проверки на наличие файловых вирусов (рис. 1.57).

Начать проверку, щёлкнув по кнопке *Scan*.

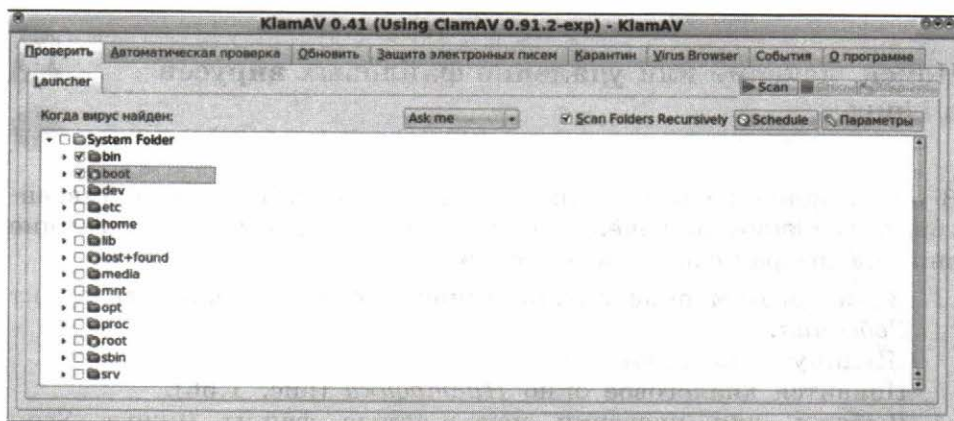


Рис. 1.57

3. После сканирования на наличие вирусов будет показан отчёт о результатах проверки.

Вопросы и задания

Опираясь на описание в тексте учебника, настройте антивирусную программу на вашем домашнем компьютере. Опишите последовательность команд и сделайте соответствующие скриншоты. Результаты работы оформите в текстовом документе и сохраните на внешнем носителе для проверки учителем.

1.6.3. Сетевые черви и защита от них

К сетевым червям (англ. *worm*) относятся вредоносные программы, распространяющие свои копии по локальным и/или глобальным сетям. Для своего распространения сетевые черви используют разнообразные сервисы глобальных и локальных компьютерных сетей: Всемирную паутину, электронную почту и т. д.

Сетевые черви являются вредоносными программами, которые проникают на компьютер, используя сервисы компьютерных сетей. Активизация сетевого червя может вызывать уничтожение программ и данных, а также похищение персональных данных пользователя.

Основным признаком, по которому типы червей различаются между собой, является способ распространения червя — как он передаёт свою копию на удалённые компьютеры. Однако многие сетевые черви используют более одного способа распространения своих копий по компьютерам локальных и глобальных сетей.

Черви, использующие уязвимости программного обеспечения. Червь ищет в сети компьютеры, на которых используются операционная система и приложения, содержащие критические уязвимости.

Под уязвимостями при этом подразумеваются, например, открытые входные порты в сетевых компонентах операционной системы, различные ошибки в работе утилит и пр. Поскольку уязвимости обычно порождены ошибками в программах, пользователям рекомендуется не забывать отслеживать и вовремя устанавливать специальные обновления и исправления установленных у них программ, выпускаемые их изготовителями. Для заражения уязвимых компьютеров червь посылает специально оформленный сетевой пакет или запрос и анализирует полученный ответ. В результате код (или часть кода) червя проникает на компьютер-жертву. Если сетевой пакет содержит только часть кода червя, он затем скачивает основной файл и запускает его на исполнение на заражённом компьютере.

Профилактическая защита от таких червей состоит также в том, что рекомендуется своевременно скачивать из Интернета и устанавливать обновления системы безопасности операционной системы и приложений.

Черви, использующие файлообменные сети. Эти черви распространяются через «торренты». Механизм работы подобных червей достаточно прост — для внедрения в файлообменную сеть червь достаточно скопировать себя в папку обмена файлами на одном из компьютеров. Сетевой червь просто помещает свою копию в папку «для скачивания», давая этой копии какое-либо «соблазнительное» название (например, под видом дистрибутива какой-то пиратской программы), так что другие пользователи скачивают и запускают его, заражая червём свой компьютер. Всю остальную работу по распространению червя файлообменная сеть «берёт на себя» — при поиске файлов в сети она сообщит удалённым пользователям о данном файле-черве и предоставит его для скачивания.



В сентябре 2001 года началась стремительное «расползание» сетевого червя Nimda, который мог атаковать компьютеры сразу несколькими способами: через сообщения электронной почты, через открытые ресурсы локальных сетей, а также используя уязвимости в системе безопасности операционной системы серверов Интернета.

Веб-черви. Отдельную категорию составляют черви, использующие для своего распространения веб-серверы. Заражение происходит в два этапа. Сначала червь проникает на компьютер-сервер и модифицирует веб-страницы сервера. Затем червь «ждёт» посетителей, которые запрашивают информацию с заражённого сервера (например, открывают в браузере заражённую веб-страницу), и таким образом проникает на другие компьютеры сети.

Разновидностью веб-червей являются **скрипты** — активные элементы (программы) на языках JavaScript или VBScript, которые могут содержаться в файлах веб-страниц. Заражение локального компьютера происходит при их передаче по Всемирной паутине с серверов Интернета в браузер локального компьютера.



В ноябре 1998 года появился первый скрипт-вирус VBScript. Rabbit, заражающий скрипты веб-страниц, а через полтора года, в мае 2000 года, грянула глобальная эпидемия скрипт-вируса LoveLetter.

Профилактическая защита от таких червей состоит в том, что в браузере можно запретить получение активных элементов и выполнение скриптов на локальном компьютере.

Ещё более эффективны веб-антивирусные программы, которые включают межсетевой экран и модуль проверки скриптов на языках JavaScript и VBScript.

Межсетевой экран. Межсетевой экран (брандмауэр, файрвол) — это программное или аппаратное обеспечение, которое проверяет информацию, входящую в компьютер из локальной сети или Интернета, а затем либо отклоняет её, либо пропускает в компьютер, в зависимости от параметров брандмауэра.

Межсетевой экран обеспечивает проверку всех веб-страниц, поступающих на компьютер пользователя. Каждая веб-страница перехватывается и анализируется межсетевым экраном на присутствие вредоносного кода. Как правило, брандмауэр предоставляет возможности управления сетевыми портами компьютера (возможность открывать или закрывать те или иные порты либо назначать их использование для тех или иных программ), а также возможность блокировать информационный обмен данного компьютера с конкретными серверами или компьютерами в компьютерной сети. Такая возможность достигается благодаря использованию «чёрных списков» IP-адресов: в такой список заносятся IP-адреса, информационный обмен с которыми пользователь хочет прекратить потому, что либо с них поступает нежелательная информация (спам), либо информационный обмен с ними связан с риском информационных угроз (например, если данный сервер заражён сетевым червём). Обычно в брандмауэре заранее предусмотрен «чёрный список» с IP-адресами вредоносных ресурсов сети Интернет, известными разработчикам брандмауэра, а также предусмотрена возможность обучения: каждый раз при начале информационного обмена с новым IP-адресом выдаётся запрос пользователю — разрешить или запретить этот информационный обмен. Встроенный брандмауэр обычно имеется в составе операционной системы либо используется отдельный программный пакет.

Если веб-страница, к которой обращается пользователь, содержит вредоносный код, то доступ к нему блокируется. При этом на экран выводится уведомление о том, что запрашиваемая страница заражена. Если веб-страница не содержит вредоносного кода, то она сразу же становится доступной для пользователя.

Если на компьютере используются программы передачи мгновенных сообщений или сетевые игры, которым требуется принимать информацию из Интернета или локальной сети, то межсетевой экран запрашивает пользователя о блокировании или разрешении подключения. Если пользователь разрешает подключение, то брандмауэр Windows создает исключение, чтобы в будущем не тревожить пользователя запросами по поводу поступления информации для этой программы.



Проверка скриптов в браузере. Проверка всех скриптов, обрабатываемых в браузере, производится следующим образом:

- каждый запускаемый на веб-странице скрипт перехватывается модулем проверки скриптов и анализируется на присутствие вредоносного кода;
- если скрипт содержит вредоносный код, то модуль проверки скриптов блокирует его, уведомляя пользователя специальным всплывающим сообщением;
- если в скрипте не обнаружено вредоносного кода, то он выполняется.

Почтовые черви. Почтовые черви распространяются через письма электронной почты. После заражения компьютера червь при любой попытке отправить кому-либо электронное письмо заражает его при отправке либо даже сам рассылает заражённые письма по всем адресам, имеющимся в адресной книге. Заражённые письма чаще всего содержат какой-либо английский текст в строке темы и «соблазнительные» по названию вложенные файлы или ссылки на сайты. Если заинтересовавшийся пользователь, получивший письмо якобы от кого-либо из знакомых, пытается запустить вложенный файл на просмотр или посетить ссылку, то он заражает свой компьютер червём, код которого содержится во вложенном файле или переписывается с сайта при посещении «вирусной» ссылки.

Профилактическая защита от почтовых червей состоит в том, что не рекомендуется открывать вложенные в почтовые сообщения файлы, полученные из сомнительных источников.

IM- и IRC-черви распространяются через чат-системы и аналогичны по принципам действия почтовым червям.

Вопросы и задания

1. Как сетевые черви проникают на компьютер?
2. Какие типы сетевых червей существуют?

Практическая работа 1.9

Защита от сетевых червей

Задание 1. В операционной системе Windows предотвратить проникновение веб-червей из локальной или глобальной сети Интернет на локальный компьютер.

Задание 2. В операционной системе Linux предотвратить проникновение веб-червей из локальной или глобальной сети Интернет на локальный компьютер.

Задание 3. В операционной системе Windows предотвратить проникновение почтовых червей из локальной или глобальной сети Интернет на локальный компьютер.

Задание 4. В операционной системе Linux предотвратить проникновение почтовых червей из локальной или глобальной сети Интернет на локальный компьютер.

Варианты выполнения работы:

- в операционной системе Windows или Linux.



Защита от веб-червей с использованием межсетевого экрана *Веб-Антивирус*, входящего в программу **Kaspersky Antivirus**

Настроим параметры межсетевого экрана *Веб-Антивирус*.

1. В контекстном меню значка антивирусной программы выбрать пункт *Настройка*.

В меню левой части появившегося диалогового окна выбрать пункт *Веб-Антивирус* (рис. 1.58).

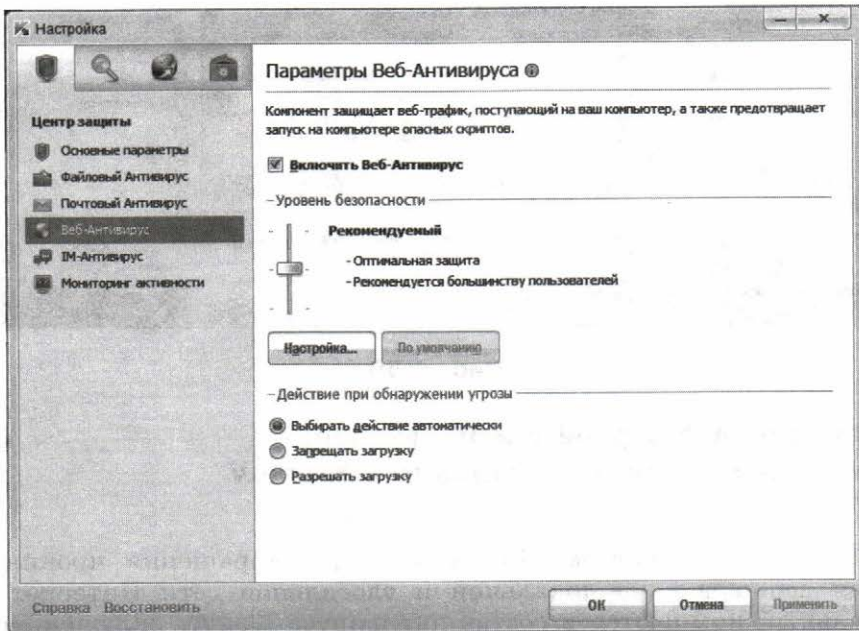


Рис. 1.58

Локальный компьютер получает веб-страницы через логические порты, обозначающиеся числовыми кодами.

2. В диалоговом окне *Настройка* перейти на вкладку *Дополнительные параметры* и выбрать пункт *Сеть*. В правой части окна щёлкнуть на кнопке *Выбрать...*.

В диалоговом окне *Сетевые порты* перечислен список портов, которые чаще всего используются для передачи веб-страниц (рис. 1.59).

Межсетевой экран предусматривает контроль прохождения веб-страниц через эти порты.

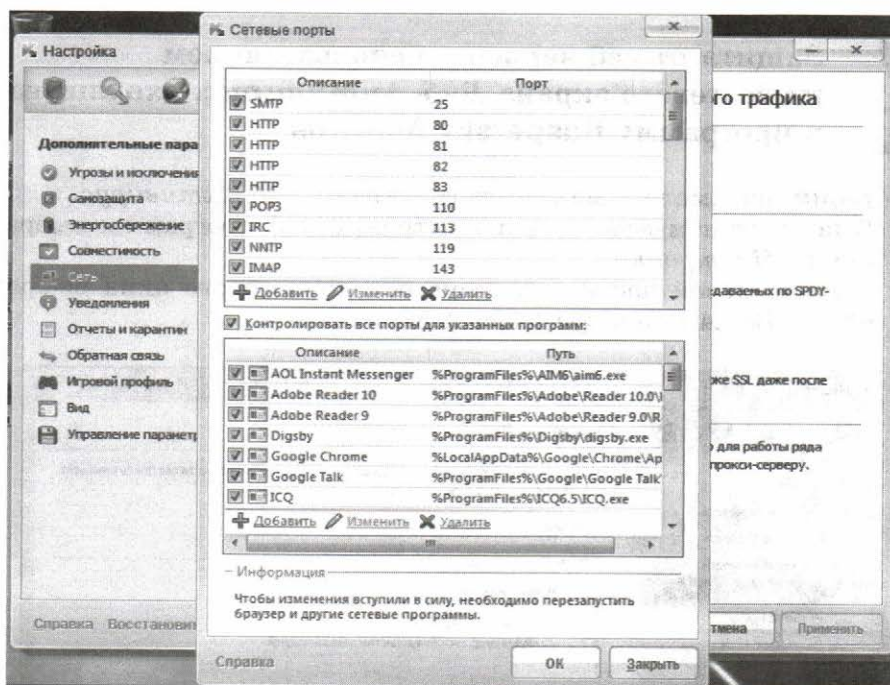


Рис. 1.59

Защита от веб-червей с использованием межсетевого экрана, входящего в KlamAV



В операционной системе Linux для предотвращения проникновения веб-червей из локальной и глобальной сети Интернет на локальный компьютер в состав антивируса KlamAV включён межсетевой экран.

1. В диалоговом окне антивирусной программы выбрать пункт *События*.

Щёлкнуть по кнопке *Options*.

Появится диалоговое окно *Настройка*.

Выбрать для проверки файлы веб-страницы (пункт *Scan HTML Files for Exploits*) (рис. 1.60).

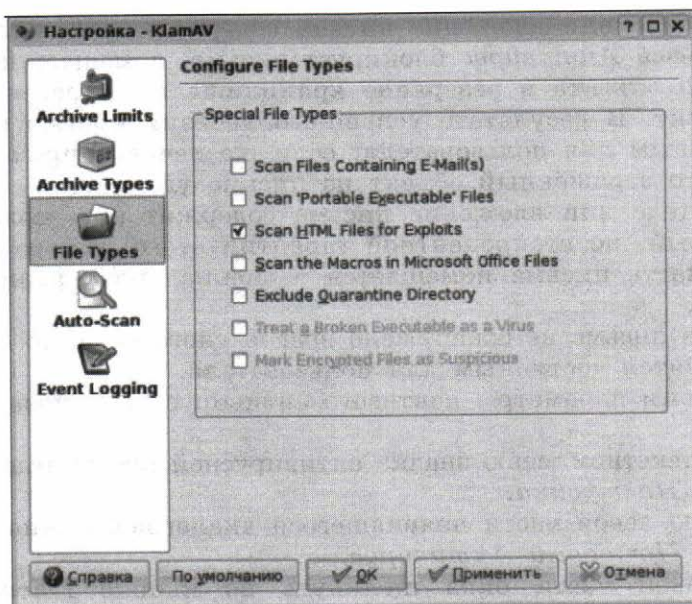


Рис. 1.60



Защита от почтовых червей с использованием *Почтового Антивируса*, входящего в Kaspersky Antivirus

В состав программы Kaspersky Antivirus включён специальный компонент, обеспечивающий защиту входящей и исходящей почты на наличие почтовых червей, — *Почтовый Антивирус*. Он запускается при старте операционной системы, постоянно находится в оперативной памяти компьютера и проверяет все почтовые сообщения.

Каждое письмо, принимаемое или отправляемое пользователем, перехватывается *Почтовым Антивирусом*, затем почтовое сообщение разбирается на составляющие его части: заголовок письма, тело, вложения.

Тело и вложения почтового сообщения проверяются на присутствие в них вредоносных программ. Распознавание вредоносных объектов происходит на основании баз, используемых в работе приложения, и с помощью эвристического алгоритма. Базы содержат описание всех известных на настоящий момент вредоносных программ и способов их обезвреживания. Эвристический алгоритм позволяет обнаруживать новые вирусы, ещё не описанные в базах.

Если тело или вложение письма содержит вредоносный код, то *Почтовый Антивирус* блокирует письмо, помещает копию заражённого объекта в **резервное хранилище** и пытается обезвредить объект. В результате успешного лечения письмо становится доступным для пользователя; если же лечение произвести не удастся, то заражённый объект из письма удаляется.

Если тело или вложение письма содержит код, похожий на вредоносный, но стопроцентной гарантии этого нет, то подозрительная часть письма помещается в специальное хранилище — **карантин**.

Если в письме не обнаружено вредоносного кода, то оно сразу же становится доступным для пользователя.

Настроим параметры почтового антивируса *Почтовый Антивирус*.

1. В контекстном меню значка антивирусной программы выбрать пункт *Настройка*.

В меню левой части появившегося диалогового окна выбрать пункт *Почтовый Антивирус*.

В правой части окна щёлкнуть по кнопке *Настройка...* (рис. 1.61).

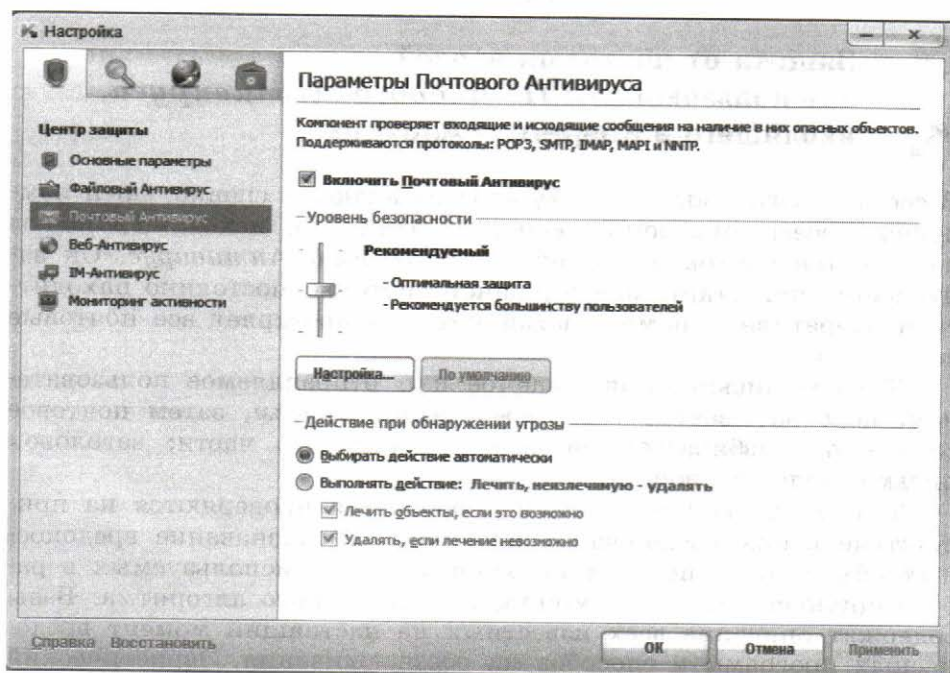


Рис. 1.61

- В появившемся диалоговом окне *Почтовый Антивирус* на вкладке *Общие* выбрать параметры защиты почтовых сообщений от почтовых вирусов (рис. 1.62).

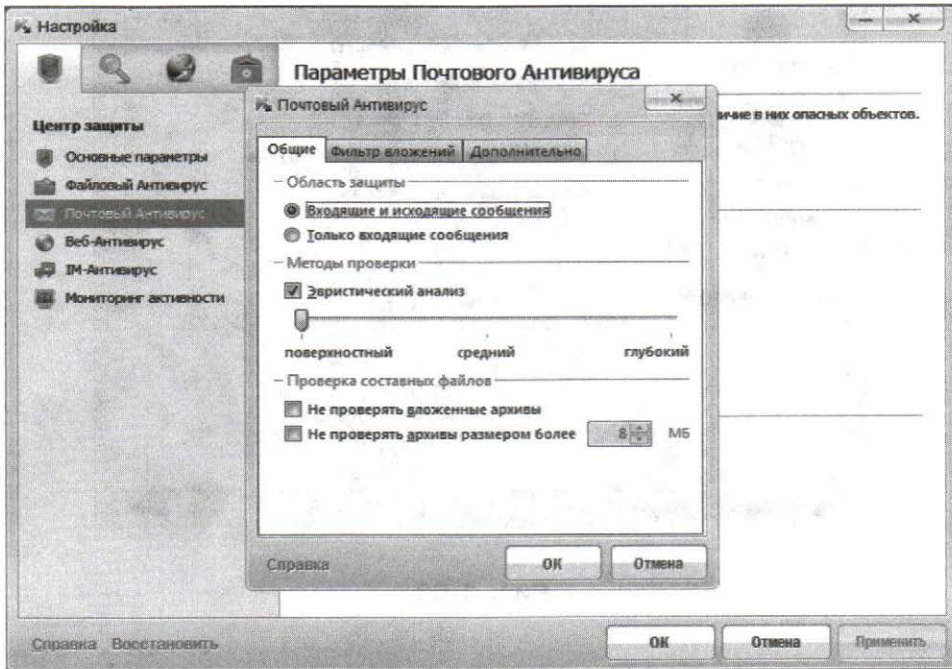


Рис. 1.62

Защита от почтовых червей с использованием антивируса KlamAV



В операционной системе Linux для предотвращения проникновения почтовых червей из локальной и глобальной сети Интернет на локальный компьютер в состав антивируса KlamAV включен почтовый антивирус.

- В диалоговом окне антивирусной программы выбрать пункт *События*.

Щёлкнуть по кнопке *Options*.

Появится диалоговое окно *Настройка*.

Выбрать для проверки файлы электронной почты (пункт *Scan Files Containing E-Mail(s)*) (рис. 1.63).

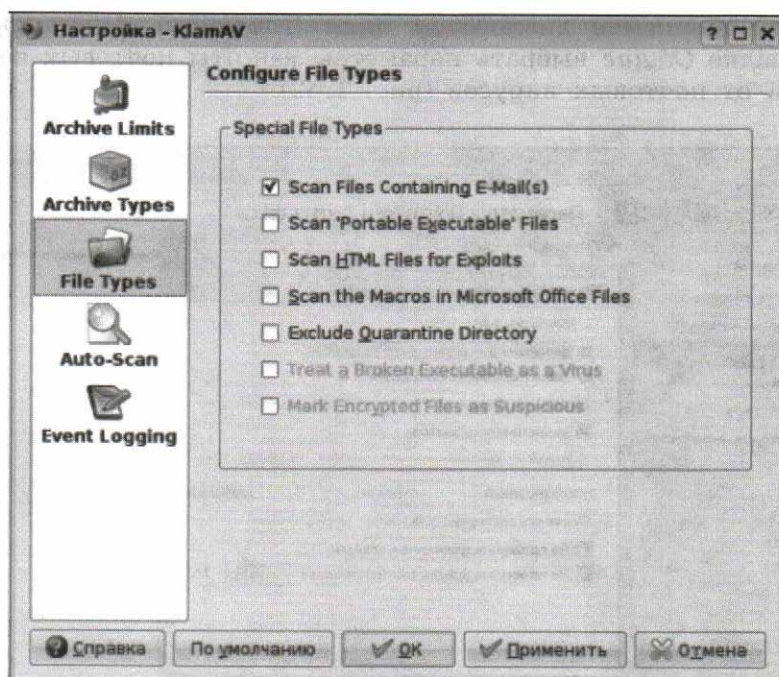


Рис. 1.63

1.6.4. Троянские программы и защита от них

Название «тройанские» дано этим программам «в честь» троянского коня, которого, по легенде, хитроумный Одиссей применил для захвата города *Троя*.

Троянская программа, в отличие от компьютерного вируса или червя, не способна к саморазмножению (хотя многие современные троянские программы совмещены с вирусами или червями). Троянские программы чаще всего нацелены не столько на разрушение или злонамеренную модификацию данных на заражённом компьютере, сколько на несанкционированный сбор и передачу информации злоумышленнику — автору такой программы (например, на кражу паролей или номеров банковских карт) или на скрытное от пользователя использование ресурсов компьютера в зловредных целях (например, для реализации удалённых атак на отдельные серверы Интернета). Кроме того, особой разновидностью троянских программ являются «логические бомбы» — программы, внедряемые злоумышленниками на компьютеры пользователей и нацеленные на разрушение инфор-

мации и нарушение работоспособности компьютера по заранее установленной дате или рассылаемой через компьютерную сеть особой командой.

В последнее время троянские программы часто являются одной из составных частей почтовых или сетевых червей либо могут заражать компьютер во время просмотра веб-страницы, содержащей скрипт — загрузчик вируса. Такие вредоносные программы часто служат для того, чтобы активизировать на заражённом компьютере обычно не используемые каналы соединения с сетью Интернет (создать на компьютере «чёрный ход» — *backdoor*). Затем, пользуясь такой «лазейкой», создатель троянской программы может уже вручную проникнуть на заражённый компьютер (взломать его) и получить доступ к хранящимся на диске данным, принадлежащим пользователю взломанного компьютера, или использовать такой компьютер для своих целей (рассылки спама, организации сетевых атак и т. д.).

Троянская программа обычно маскируется под какую-либо полезную программу или новую версию популярной утилиты в расчете, что её скачает кто-либо из пользователей и запустит на своём компьютере. Нередко троянские программы рассылаются по электронным адресам или предлагаются для скачивания на различных сайтах под видом полезных бесплатных утилит (либо такие бесплатные программы содержат «троянского коня» в качестве «программной закладки»).

Троянская программа, троянец (от англ. *trojan*) — вредоносная программа, которая выполняет несанкционированную пользователем передачу управления компьютером удалённому пользователю, а также действия по удалению, модификации, сбору и пересылке информации третьим лицам.

Троянские программы различаются между собой по тем действиям, которые они производят на заражённом компьютере.

Троянские утилиты удалённого администрирования. Троянские программы этого класса являются утилитами удалённого администрирования компьютеров в сети. Они позволяют скрытно от пользователя принимать или отсылать файлы, запускать и уничтожать их, выводить сообщения, стирать информацию, перезагружать компьютер и т. д.

При запуске троянец устанавливает себя в системе и затем следит за ней. При этом пользователю не выдаётся никаких сообщений о действиях троянской программы в системе. В результате



«пользователь» этой троянской программы не знает о её присутствии в системе, в то время как его компьютер открыт для удалённого управления.

Троянские программы данного типа являются одним из самых опасных видов вредоносного программного обеспечения, поскольку в них заложена возможность самых разнообразных злоумышленных действий, в том числе они могут быть использованы для обнаружения и передачи конфиденциальной информации.



В 2003 году широкое распространение получила троянская программа Backdoor.Win32.BO, которая осуществляет следующие действия:

- высылает имена компьютера, пользователя и информацию о системе: тип процессора, размер памяти, версию системы, информацию об установленных устройствах;
- посылает/принимает, уничтожает, копирует, переименовывает, исполняет любой файл;
- отключает пользователя от сети;
- «завешивает» компьютер;
- читает или модифицирует системный реестр.

Троянские программы — шпионы. Такие троянцы осуществляют электронный шпионаж за пользователем заражённого компьютера: вводимая с клавиатуры информация, снимки экрана, список активных приложений и действия пользователя с ними сохраняются в каком-либо файле на диске и периодически отправляются злоумышленнику.

Троянские программы этого типа часто используются для кражи информации пользователей различных систем онлайн-платежей и банковских систем.

Рекламные программы. Рекламные программы (англ. *Adware: Advertisement* — реклама и *Software* — программное обеспечение) встраивают рекламу в основную полезную программу и могут выполнять функцию троянских программ. Рекламные программы могут скрытно собирать различную информацию о пользователе компьютера и затем отправлять её злоумышленнику.

Защита от троянских программ. Троянские программы часто изменяют записи системного реестра операционной системы, который содержит все сведения о компьютере и установленном программном обеспечении. Для их удаления необходимо восстановление системного реестра, поэтому компонент, восстанавливающий системный реестр, входит в современные операционные системы.

Вопросы и задания



Какие вредоносные действия выполняют троянские программы?

Практическая работа 1.10



Самозащита компьютера с помощью программы Kaspersky Antivirus

Задание 1. В операционной системе Windows предотвратить проникновение и действия троянских программ.

Задание 2. В операционной системе Windows восстановить систему (системный реестр) после проникновения троянских программ.

Варианты выполнения работы:

- включить разные параметры проактивной защиты.



Защита компьютера от изменения и удаления файлов от удалённого взаимодействия с использованием *Самозащиты*, входящей в Kaspersky Antivirus

В состав Kaspersky Antivirus включен специальный компонент *Самозащита*, который анализирует и сообщает пользователю:

- действия каждого запускаемого на компьютере приложения, которые являются характерными при опасной (шпионской) активности;
- попытки изменения системного реестра (удаление, добавление ключей системного реестра, ввод значений для ключей в недопустимом формате), характерные для шпионской деятельности.

Настроим параметры компонента *Самозащита*.

1. В контекстном меню значка антивирусной программы выбрать пункт *Настройка*.

В меню левой части появившегося диалогового окна перейти на вкладку *Дополнительные параметры* и выбрать пункт *Самозащита*.

2. В правой части окна установить флажки (рис. 1.64):
 - *Включить самозащиту*;
 - *Отключить возможность внешнего управления системной службой*.

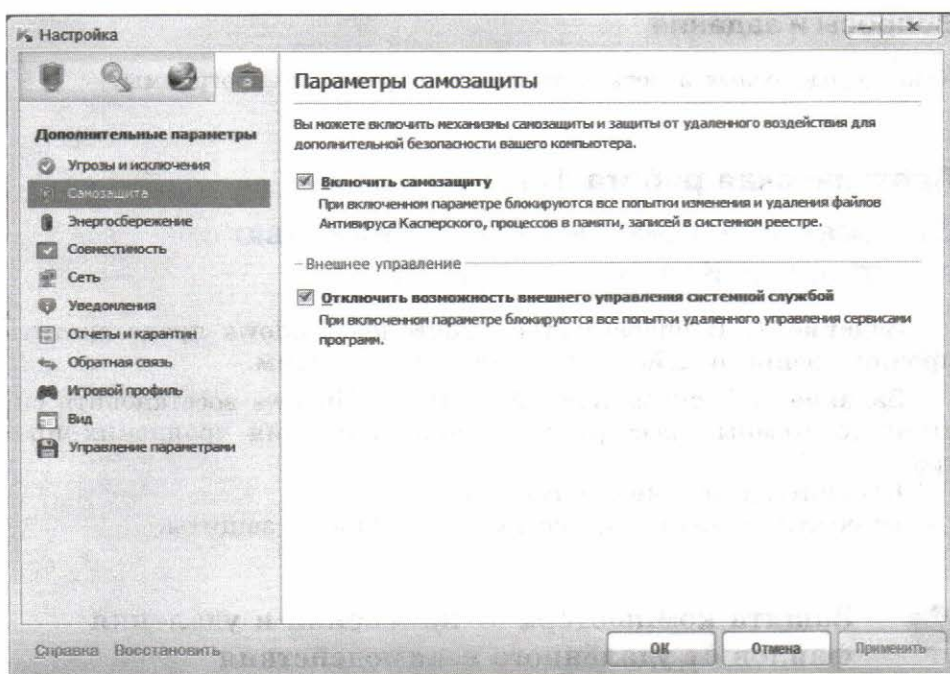


Рис. 1.64



Восстановление операционной системы (системного реестра) после проникновения троянских или других вредоносных программ

В операционной системе Windows 7 восстановление системы можно провести следующим образом.

1. Ввести команду *Панель управления—Архивирование данных компьютера*.

В появившемся диалоговом окне *Архивация или восстановление файлов* активировать ссылку *Восстановить системные параметры или компьютер*.

Появится диалоговое окно *Восстановление ранее сохраненного состояния этого компьютера*, в котором можно выбрать точку восстановления системы (системного реестра).

2. Щёлкнуть по кнопке *Запуск восстановления системы*.

Появится диалоговое окно *Свойства системы*, в котором на вкладке *Защита системы* можно выбрать или создать точку восстановления системы (системного реестра) (рис. 1.65).

Кроме того, можно создать точки восстановления для любых логических дисков.

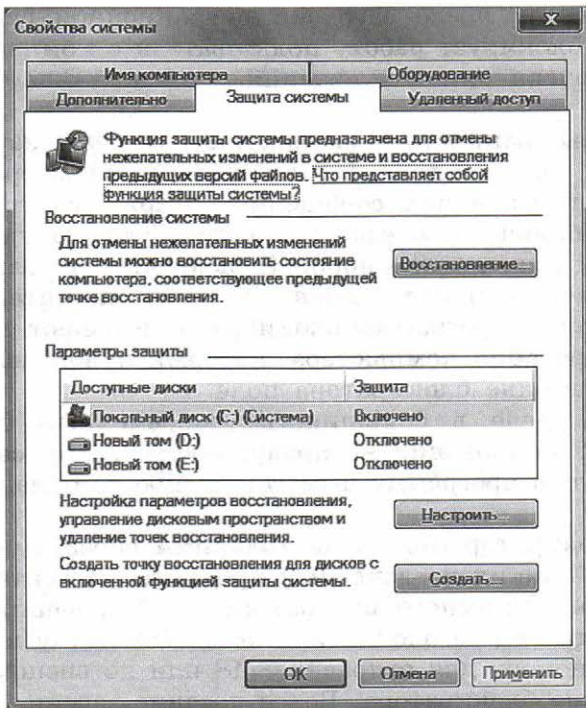


Рис. 1.65

1.6.5. Блокираторы и другие программы-вымогатели

В последние годы всё более широкое распространение получили вредоносные программы — **блокираторы**. После заражения компьютера такая программа (сразу или через некоторое время) выводит на экран монитора окно с сообщением, что пользователь якобы просматривал в браузере запрещённую информацию или что на компьютере якобы установлено нелицензионное программное обеспечение. Такое сообщение (для «устрашения» пользователя) обычно делается якобы от имени прокуратуры или другой организации подобного рода. Блокиратор, работающий на уровне браузера, представляет собой веб-страницу, которую невозможно закрыть (а в некоторых случаях невозможно закрыть и окно браузера). Более опасный блокиратор, работающий на уровне операционной системы, представляет собой окно программы, которое не только не закрывается стандартными способами, но и постоянно располагается поверх любых других окон (в том числе окна *Диспетчера задач*, раскрываемого комбинацией клавиш Ctrl + Alt + Del). Такая программа к тому же автоматически

запускается сразу после загрузки операционной системы и таким образом блокирует работу пользователя с компьютером даже после перезапуска или выключения и последующего включения компьютера.

Программы-блокираторы полностью нацелены на вымогательство денег у пользователя заражённого компьютера: обычно в тексте выводимого в них сообщения содержится требование отправить с мобильного телефона платную SMS на указанный номер (обычный или четырёхзначный, подобный используемым для оплаты различных информационных услуг на интернет-сайтах). Взамен создатели программы-блокиратора обещают прислать код для разблокирования компьютера, который нужно ввести в предусмотренном в окне блокиратора поле.

На самом деле в большинстве современных блокираторов функция разблокирования не предусмотрена, т. е. создавшие такую вредоносную программу вымогатели просто обманывают пользователей.

Снять блокиратор можно несколькими способами. Для более ранних версий таких вредоносных программ на сайтах фирм-разработчиков антивирусного программного обеспечения можно попытаться найти код разблокировки по сообщаемому вымогателями телефонному номеру для отправки SMS или по внешнему виду сообщения в окне блокиратора. Более поздние версии блокираторов можно удалить с компьютера (и тем самым разблокировать его), воспользовавшись специальными системными CD-дисками, скачиваемыми с сайтов антивирусных программ на другом, незаражённом компьютере. Например, на сайте Лаборатории Касперского можно скачать информационный образ диска *Kaspersky Rescue Disk* (<http://support.kaspersky.ru/viruses/rescuedisk>), записать его на диск CD-R и затем загрузить с полученного оптического диска заражённый компьютер. После этого при помощи размещённой на таком диске антивирусной программы выполняется удаление блокиратора, а затем рекомендуется выполнить полную проверку компьютера на наличие вредоносных программ для их удаления.

Существуют и другие программы-вымогатели. К ним относится, например, программа, рассылаемая по электронной почте (подобно почтовым червям) и маскирующаяся под резюме претендента на работу, подготовленное в формате PDF. При заражении компьютера эта программа шифрует файлы с данными (документы Word, Excel, PowerPoint, файлы PDF, графические файлы), а затем выводит сообщение о требовании выплатить деньги за «код расшифровки» для их восстановления. Эта вредоносная программа особенно опасна тем, что расшифровать содержимое файлов на заражённом компьютере иными способами (не зная код шифрова-

ния) практически невозможно, а создатели программы-вымогателя даже не предполагают присылать никаких «кодов расшифровки».

При заражении компьютера блокиратором или любой другой программой — вымогателем ни в коем случае не следует отправлять их создателям деньги или платные SMS: тем самым вы не получите от них никаких средств для разблокирования, но подержите их незаконные действия!

Отметим, что у большинства сотовых операторов предусмотрена специальная бесплатная возможность по номеру платной SMS узнать, какая сумма будет списана со счёта сотового телефона при её отправке.

1.6.6. Хакерские утилиты и защита от них

Хакерские атаки обычно основаны на особенностях или ошибках в имеющейся операционной системе или прикладном программном обеспечении. Такие ошибки (их называют «дырами в безопасности») позволяют переслать на атакуемый компьютер и запустить на нём программный код, позволяющий затем удалённо (через компьютерную сеть) управлять этим компьютером и его ресурсами, а также запускать на нём те или иные программы без ведома его пользователя. Такие захваченные хакером чужие компьютеры называют «зомби», обычно они используются для рассылки вредоносных программ, для осуществления хакерских атак на другие компьютеры и серверы сети Интернет, а также для «добычи» электронной валюты (биткойнов), операции с которой в России запрещены.

Сетевые атаки. Сетевые атаки на удалённые серверы реализуются с помощью специальных программ, которые посылают на них многочисленные запросы. Это приводит к отказу в обслуживании (зависанию сервера), если ресурсы атакуемого сервера недостаточны для обработки всех поступающих запросов.

DoS-программы (от англ. *Denial of Service* — отказ в обслуживании) реализуют атаку с одного компьютера с ведома пользователя. DoS-программы обычно наносят ущерб удалённым компьютерам и сетям, не нарушая работоспособности заражённого компьютера.

DDoS-программы (от англ. *Distributed DoS* — распределённый DoS) реализуют распределённые атаки с разных компьютеров, причем без ведома пользователей заражённых компьютеров. Для этого DDoS-программа засылается на компьютеры «жертв-посредников» и после запуска в зависимости от текущей даты или по команде от хакера начинает сетевую атаку на указанный сервер в сети.



Некоторые сетевые черви содержат в себе DoS-процедуры, атакующие сайты, которые по каким-либо причинам «невзлюбил» автор червя. Так, червь Codored 20 августа 2001 года организовал успешную атаку на официальный сайт президента США, а червь Mydoom 1 февраля 2004 года «выключил» сайт компании — производителя дистрибутивов UNIX.

Некоторые хакерские утилиты реализуют фатальные сетевые атаки. Такие утилиты используют уязвимости в операционных системах и приложениях и отправляют специально оформленные запросы на атакуемые компьютеры в сети. В результате сетевой запрос специального вида вызывает критическую ошибку в атакуемом приложении, и система прекращает работу.

Утилиты взлома удалённых компьютеров. Утилиты взлома удалённых компьютеров предназначены для проникновения на удалённые компьютеры с целью дальнейшего управления ими (используя методы троянских программ типа утилит удалённого администрирования) или для внедрения во взломанную систему других вредоносных программ.

Профилактическая защита от хакерских утилит состоит в своевременной загрузке из Интернета обновлений системы безопасности операционной системы и приложений.



Руткиты. Руткит (от англ. *root kit* — набор для получения прав root) — программа или набор программ для скрытого взятия под контроль взломанной системы. Это утилиты, используемые для сокрытия вредоносной активности. Они маскируют вредоносные программы, чтобы избежать их обнаружения антивирусными программами. Руткиты модифицируют операционную систему на компьютере и заменяют её основные функции, чтобы скрыть своё присутствие и действия, которые предпринимает злоумышленник на зараженном компьютере.

Защита от хакерских атак, сетевых червей и троянских программ. Защита компьютерных сетей или отдельных компьютеров от несанкционированного доступа может осуществляться с помощью межсетевого экрана.

Межсетевой экран позволяет:

- 1) блокировать хакерские DoS-атаки, не пропуская на защищаемый компьютер сетевые пакеты с определённых серверов (определённых IP-адресов или доменных имен);
- 2) не допускать проникновение на защищаемый компьютер сетевых червей (почтовых, веб-червей и др.);
- 3) препятствовать троянским программам отправлять конфиденциальную информацию о пользователе и компьютере.

Вопросы и задания



1. Какие типы хакерских атак и методы защиты от них существуют?
2. К какому типу вредоносных программ относятся руткиты?

Практическая работа 1.11



Защита от хакерских атак

Задание 1. В операционной системе Windows обнаружить и обезвредить угрозы.

Задание 2. В операционной системе Windows включить брандмауэр Windows (межсетевой экран), который предотвращает попытки доступа хакеров или вредоносных программ к локальному компьютеру через локальную сеть или Интернет.

Варианты выполнения работы:

- по-разному настроить сетевой экран (брандмауэр).



Защита от угроз с помощью компонента *Угрозы и исключения*, входящих в Kaspersky Antivirus

В состав Kaspersky Antivirus включён специальный компонент *Угрозы и исключения*, который позволяет обнаружить объекты следующих типов:

- вирусы, черви, троянские программы и вредоносные утилиты;
- рекламные программы и программы автодозвона;
- неизвестные упаковщики и многократно упакованные файлы.

Настроим параметры компонента *Угрозы и исключения*.

1. В контекстном меню значка антивирусной программы выбрать пункт *Настройка*.

В меню левой части появившегося диалогового окна перейти на вкладку *Дополнительные параметры* и выбрать пункт *Угрозы и исключения*.

В правой части окна щёлкнуть по верхней из двух кнопок *Настройка* (рис. 1.66).

2. В появившемся диалоговом окне *Объекты для обнаружения* установить флажки *Вредоносные утилиты*, *Рекламные программы*, *Программы автодозвона*, *Неизвестные упаковщики* и *Многократно упакованные файлы* (рис. 1.67).

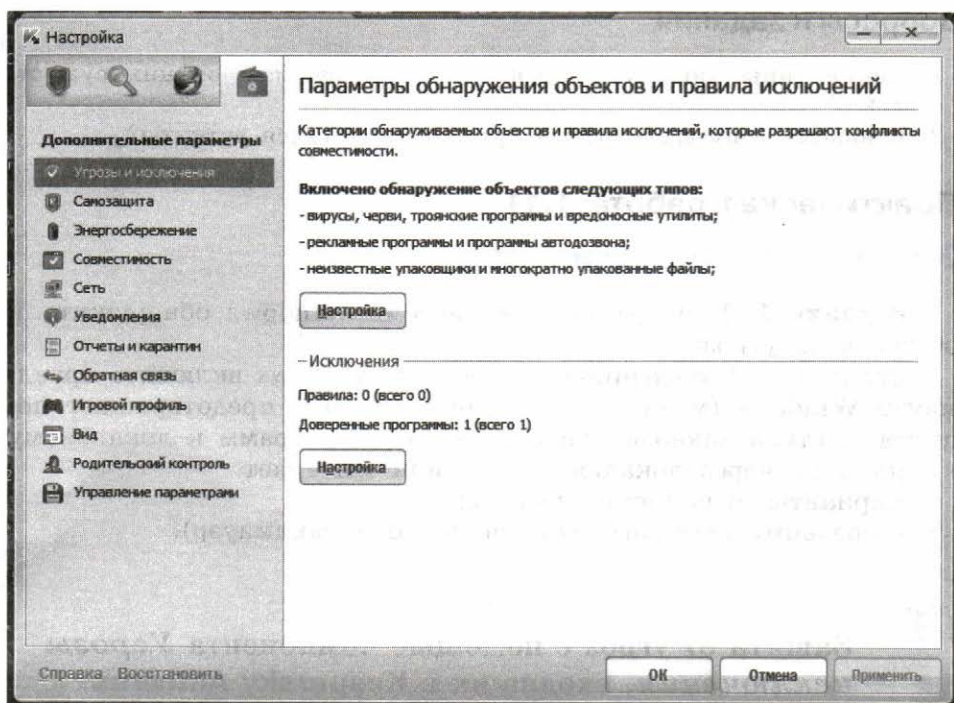


Рис. 1.66

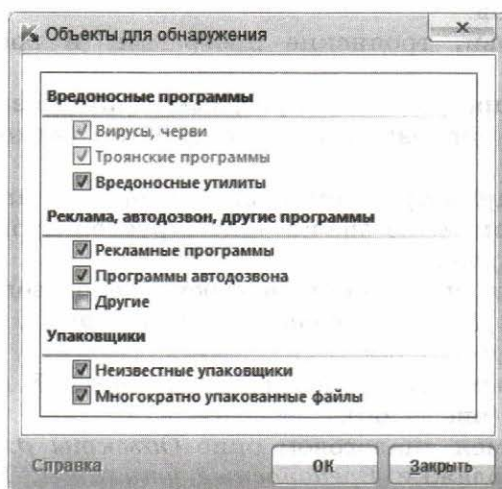


Рис. 1.67



Настройка межсетевого экрана (брандмауэра) в операционной системе Windows

В операционной системе Windows 7 настроим межсетевой экран (брандмауэр).

1. Ввести команду *Панель управления—Брандмауэр Windows*. В левой части появившегося диалогового окна *Панель управления\Все элементы управления\Брандмауэр Windows* активировать ссылку *Включение и отключение брандмауэра Windows*.

2. Появится диалоговое окно *Все элементы управления\Брандмауэр Windows\Настроить параметры*.

Установить все переключатели в положение *Включить* (рис. 1.68).

Брандмауэр Windows предотвращает попытки доступа хакеров или вредоносных программ к локальному компьютеру через сеть.

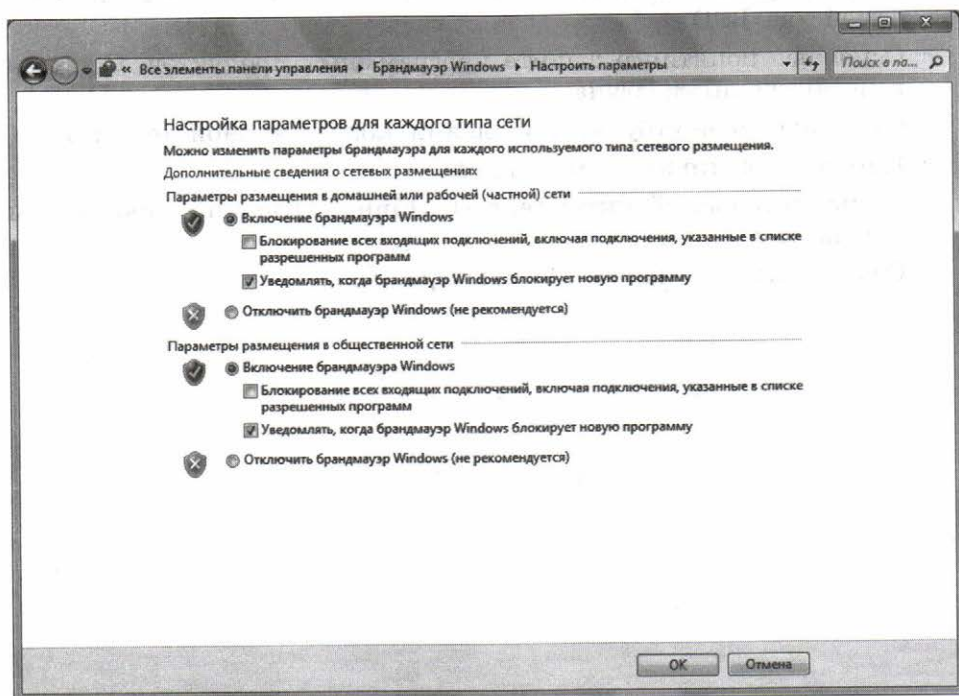


Рис. 1.68

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

- Архитектура ЭВМ
- Базовые утилиты
- Безопасность при использовании информационных технологий
- Безопасность при использовании информационных технологий: несанкционированный доступ к данным, вредоносные программы, криптография
- Введение в операционные системы
- Информационная безопасность
- История развития вычислительных устройств
- Компьютерные вирусы и антивирусные программы
- Конфигурация компьютера. Выбор конфигурации в зависимости от решаемых задач
- Криптографические методы защиты программных продуктов
- Настройки операционной системы на примере Windows 7
- Операционная система GNU/Linux и операционные системы семейства BSD
- Основные понятия о защите программных продуктов. Системы разграничения доступа
- От абака до ноутбука. Поколения компьютерной техники
- Пароль и антивирусная защита
- Пользовательский интерфейс ОС Linux. Основные программы и утилиты
- Сборка персонального компьютера

Глава 2

МОДЕЛИРОВАНИЕ И ФОРМАЛИЗАЦИЯ

При изучении данной главы рекомендуется установить следующее программное обеспечение:

www

для операционной системы Windows:

- систему объектно-ориентированного программирования Microsoft Visual Studio;



- электронные таблицы Microsoft Excel;



для операционных систем Windows и Linux:

- систему объектно-ориентированного программирования Lazarus;



- электронные таблицы OpenOffice Calc.



2.1. Моделирование как метод познания

Моделирование. Человечество в своей деятельности (научной, образовательной, технологической, художественной) постоянно создаёт и использует модели окружающего мира. Строгие правила построения моделей сформулировать невозможно, однако человечество накопило богатый опыт моделирования различных объектов и процессов.

Модели позволяют представить в наглядной форме объекты и процессы, недоступные для непосредственного восприятия (очень большие или очень маленькие объекты, очень быстрые или очень медленные процессы и др.). Наглядные модели часто используются при обучении. В курсе географии первые представления о нашей планете Земля мы получаем, изучая её модель — глобус, в курсе физики изучаем работу двигателя внутреннего сгорания по его модели, в химии при изучении строения вещества используем модели молекул и кристаллических решёток, в биологии изучаем строение человека по анатомическим муляжам и др.

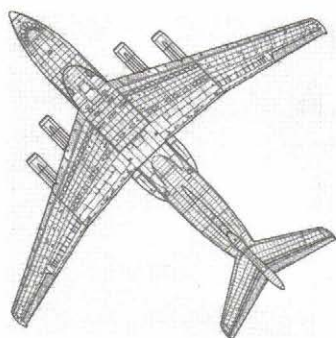


Рис. 2.1

Модели имеют чрезвычайно важную роль в проектировании и создании различных технических устройств, машин и механизмов, зданий, электрических цепей и т. д. Без предварительного создания чертежа (рис. 2.1) невозможно изготовить даже простую деталь, не говоря уже о самолёте.

В процессе проектирования зданий и сооружений кроме чертежей часто изготавливают их макеты. В процессе разработки летательных аппаратов поведение их моделей в воздушных потоках исследуют в аэродинамической трубе.

Разработка электрической схемы обязательно предшествует созданию электрических цепей и т. д.

Развитие науки невозможно без создания **теоретических моделей** (теорий, законов, гипотез и т. д.), отражающих строение, свойства и поведение реальных объектов. Создание новых теоретических моделей иногда коренным образом меняет представление человечества об окружающем мире (гелиоцентрическая система мира Коперника, модель атома Резерфорда–Бора, модель расширяющейся Вселенной, модель генома человека и др.). Истинность теоретических моделей, т. е. их соответствие законам реального мира, проверяется с помощью опытов и экспериментов.

Всё художественное творчество фактически является процессом создания моделей. Например, такой литературный жанр, как басня, переносит реальные отношения между людьми на отношения между животными и фактически создаёт модели человеческих отношений. Более того, практически любое литературное произведение может рассматриваться как модель реальной человеческой жизни. Моделями, в художественной форме отражающими реальную действительность, являются также живописные полотна, скульптуры, театральные постановки и т. д.

Моделирование — это метод познания, состоящий в создании и исследовании моделей.

Модель. Каждый объект имеет большое количество различных свойств. В процессе построения модели выделяются главные, наиболее существенные для проводимого исследования (с точки зрения цели моделирования) свойства. В процессе исследования аэродинамических качеств модели самолёта в аэродинамической трубе

важно, чтобы модель имела геометрическое подобие оригиналу, но не важен, например, её цвет. При построении электрических схем — моделей электрических цепей — необходимо учитывать порядок подключения элементов цепи друг к другу, но неважно их геометрическое расположение друг относительно друга и т. д.

Разные науки исследуют объекты и процессы под разными углами зрения и строят различные типы моделей. В физике изучаются процессы взаимодействия и изменения объектов, в химии — их химический состав, в биологии — строение и поведение живых организмов и т. д.

Возьмём в качестве примера человека, в разных науках он исследуется в рамках различных моделей. В рамках механики его можно рассматривать как материальную точку, в химии — как объект, состоящий из различных химических веществ, в биологии — как систему, стремящуюся к самосохранению и т. д.

Модель — это такой новый объект, который отражает существенные с точки зрения цели проводимого исследования (цели моделирования) свойства изучаемого объекта.



Никакая модель не может заменить сам объект. Но при решении конкретной задачи, когда нас интересуют определённые свойства изучаемого объекта, модель оказывается полезным, а подчас и единственным инструментом исследования.

Вопросы и задания



1. Может ли объект иметь несколько моделей? Приведите пример.
2. Могут ли разные объекты описываться одной и той же моделью? Если да, приведите пример.

2.2. Системный подход в моделировании

Система. В курсе 10 класса вы познакомились с понятием системы. Напомним основные положения. Окружающий нас мир состоит из множества различных объектов, каждый из которых имеет разнообразные свойства, и при этом объекты взаимодействуют между собой. Например, такие объекты, как планеты нашей Солнечной системы, имеют различные свойства (массу, геометрические размеры и т. д.) и по закону всемирного тяготения взаимодействуют с Солнцем и друг с другом.

Планеты входят в состав более крупного объекта — Солнечной системы, а Солнечная система — в состав нашей галактики Млечный Путь. Вместе с тем планеты состоят из атомов различ-

ных химических элементов, а атомы — из элементарных частиц. Можно сделать вывод, что практически каждый объект состоит из других объектов, т. е. представляет собой **систему**.

Система состоит из объектов, которые называются **элементами** системы.

Важным признаком системы является ее **целостное функционирование**. Система является не набором отдельных элементов, а совокупностью взаимосвязанных элементов. Например, компьютер является системой, состоящей из различных устройств, при этом устройства связаны между собой и аппаратно (физически подключены друг к другу), и функционально (между устройствами происходит обмен информацией).

В системе возникает **системный эффект**, заключающийся в появлении у системы новых свойств, не присущих её составным частям. При этом польза от системы в целом превышает суммарную пользу её компонентов. Например, если бы телефон купил себе только один человек, то он не смог бы никому позвонить. Если телефон купят два человека, то они могут позвонить друг другу. И так далее: чем больше телефонов уже куплено, тем больше интерес купить себе телефон у каждого нового участника.

Состояние системы характеризуется ее **структурой**, т. е. составом и свойствами элементов, их отношениями между собой. Система сохраняет свою целостность под воздействием различных внешних воздействий и внутренних изменений до тех пор, пока она сохраняет неизменной свою структуру. Если структура системы меняется (например, удаляется один из элементов), то система может перестать функционировать как целое. Так, если удалить одно из устройств компьютера (например, процессор), компьютер выйдет из строя, т. е. прекратит свое существование как система.

Статические информационные модели. Любая система существует в пространстве и времени. В каждый момент времени система находится в определённом состоянии, которое характеризуется составом элементов, значениями их свойств, величиной и характером взаимодействия между элементами и т. д.

Так, состояние Солнечной системы в любой момент времени характеризуется составом входящих в неё объектов (Солнце, планеты и др.), их свойствами (размерами, положением в пространстве и т. д.), величиной и характером взаимодействия между собой (силами тяготения, с помощью электромагнитных волн и т. д.).

Модели, описывающие состояние системы в определённый момент времени, называются **статическими информационными моделями**.

В физике примерами статических информационных моделей являются модели, описывающие простые механизмы, в биоло-

гии — модели строения растений и животных, в химии — модели строения молекул и кристаллических решёток и т. д.

Динамические информационные модели. Состояние систем изменяется во времени, т. е. происходят процессы **изменения и развития систем**. Так, планеты движутся, изменяется их положение относительно Солнца и друг друга, Солнце, как и любая другая звезда, развивается, меняются его химический состав, излучение и т. д.

Модели, описывающие процессы изменения и развития систем, называются **динамическими информационными моделями**.

В физике динамические информационные модели описывают движение тел, в биологии — развитие организмов или популяций животных, в химии — процессы прохождения химических реакций и т. д.

Вопросы и задания

1. Образуют ли систему комплектующие компьютера: до сборки? После сборки? После включения компьютера?
2. В чём разница между статическими и динамическими информационными моделями? Приведите примеры статических и динамических информационных моделей.

2.3. Формы представления моделей

Модели материальные и модели информационные. Модели можно разделить на два больших класса: модели **предметные (материальные)** и модели **информационные**. Предметные модели воспроизводят геометрические, физические и другие свойства объектов в материальной форме (глобус, анатомические муляжи, модели кристаллических решёток, макеты зданий и сооружений и др.).

Информационные модели представляют объекты и процессы в **образной или знаковой форме**.

Образные модели (рисунки, фотографии и др.) представляют собой зрительные образы объектов, зафиксированные на каком-либо носителе информации (бумаге, фото- и киноплёнке и др.). Широко используются образные информационные модели в образовании (вспомните учебные плакаты по различным предметам) и науках, где требуется классификация объектов по их внешним признакам (в ботанике, биологии, палеонтологии и др.).

Знаковые информационные модели строятся с использованием различных языков (знаковых систем). Знаковая информационная модель может быть представлена в форме текста (например, программы на языке программирования), формулы (например, второго закона Ньютона $\vec{F} = m \cdot \vec{a}$), таблицы (например, таблицы Периодической системы элементов Д. И. Менделеева) и т. д.



Иногда при построении знаковых информационных моделей используются одновременно несколько различных языков. Примерами таких моделей могут служить географические карты, графики, диаграммы и др. Во всех этих моделях используются одновременно как язык графических элементов, так и язык символов — букв, цифр и пр.

На протяжении своей истории человечество использовало различные способы и инструменты для создания информационных моделей. Эти способы постоянно совершенствовались. Так, первые информационные модели создавались в форме наскальных рисунков. В настоящее время информационные модели обычно строятся и исследуются с использованием современных компьютерных технологий.

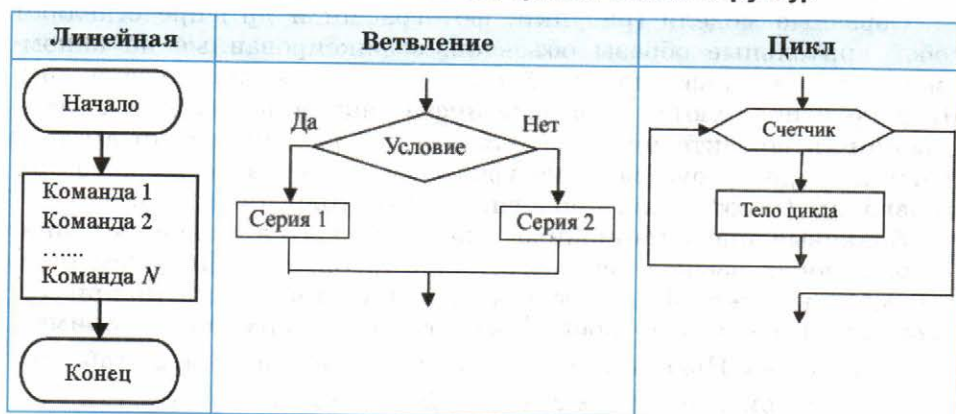
Алгоритм как информационная модель. Алгоритмы лежат в основе современных информационных технологий. Алгоритм является информационной моделью процесса решения задачи. Исполнитель алгоритма выполняет алгоритм формально, не вникая в содержание поставленной задачи.

Человек при разработке и исполнении алгоритмов использует язык блок-схем. Блок-схема позволяет сделать алгоритм более наглядным и выделить в нём **основные алгоритмические структуры** (линейная, ветвление, цикл). Человек может по блок-схеме легко проследить выполнение алгоритма, так как элементы блок-схем соединены стрелками, указывающими последовательность действий.

Элементы алгоритма изображаются на блок-схеме с помощью различных геометрических фигур (табл. 2.1), внутри которых записывается программный код.

Таблица 2.1

Блок-схемы основных алгоритмических структур



Вопросы и задания



1. Какие существуют типы информационных моделей?
2. Каковы основные типы алгоритмических структур?

2.4. Формализация

Естественные (разговорные) языки используются для создания **описательных информационных моделей**. В истории науки известны многочисленные описательные информационные модели; например гелиоцентрическая модель мира, которую предложил Коперник, формулировалась следующим образом:

- Земля вращается вокруг своей оси и вокруг Солнца;
- орбиты всех планет проходят вокруг Солнца.

С помощью формальных языков строятся **формальные информационные модели** (математические, логические и др.). Одним из наиболее широко используемых формальных языков является язык математики. Модели, построенные с использованием математических обозначений и формул, называются **математическими моделями**. Язык математики является совокупностью формальных языков, с некоторыми из них (алгебра, геометрия, тригонометрия) вы знакомитесь в школе, с другими (теория множеств, теория вероятностей и др.), сможете ознакомиться в процессе дальнейшего обучения.

Язык алгебры позволяет формализовать функциональные зависимости между величинами. Так, Ньютон формализовал гелиоцентрическую систему мира, открыв законы механики и закон всемирного тяготения и записав их в виде алгебраических функциональных зависимостей. В школьном курсе физики рассматривается много разнообразных функциональных зависимостей, выраженных на языке алгебры, которые представляют собой математические модели изучаемых явлений или процессов.

Язык алгебры логики (алгебры высказываний) позволяет строить **формальные логические модели**. С помощью алгебры логики можно формализовать (записать в виде логических выражений) простые и сложные высказывания, выраженные на естественном языке. Построение логических моделей позволяет решать логические задачи, строить логические модели устройств компьютера (сумматора, триггера) и т. д.

Процесс построения информационных моделей с помощью формальных языков называется **формализацией**.



В процессе познания окружающего мира человечество постоянно использует моделирование и формализацию. При изучении нового объекта сначала обычно строится его описательная информационная модель на естественном языке, затем она формализуется, т. е. выражается с использованием формальных языков (математики, логики и др.).

Визуализация формальных моделей. В процессе исследования формальных моделей часто производится их визуализация. Для визуализации алгоритмов используются блок-схемы, пространственных соотношений между объектами — чертежи, моделей электрических цепей — электрические схемы, логических моделей устройств — логические схемы и т. д.

Так при визуализации формальных физических моделей с помощью анимации может отображаться динамика процесса, производится построение графиков изменения физических величин и т. д. В настоящее время широкое распространение получили **компьютерные интерактивные визуальные модели**. В таких моделях исследователь может менять начальные условия и параметры протекания процессов и наблюдать изменения в поведении модели.



Вопросы и задания

1. Какие бывают модели? Приведите примеры материальных и информационных моделей.
2. Что такое формализация? Приведите примеры формальных моделей.

2.5. Основные этапы разработки и исследования моделей на компьютере

Использование компьютера для исследования информационных моделей различных объектов и систем позволяет изучить их изменения в зависимости от значений тех или иных параметров. Процесс разработки моделей и их исследование на компьютере можно разделить на несколько основных этапов.

Описательная информационная модель. На первом этапе исследования объекта или процесса обычно строится описательная информационная модель на естественном языке. Такая модель выделяет существенные, с точки зрения целей проводимого исследования, свойства (параметры) объекта, а несущественными свойствами пренебрегает.

Формальная модель. На втором этапе создаётся формальная модель, т. е. описательная информационная модель записывает-

ся с помощью какого-либо формального языка. В такой модели с помощью формул, уравнений, неравенств и т. д. фиксируются формальные соотношения между исходными и искомыми величинами, а также накладываются ограничения на допустимые значения этих величин.

Однако далеко не всегда удаётся найти формулы, явно выражающие искомые величины через исходные. В таких случаях используются приближённые математические методы, позволяющие получать результаты с заданной точностью.

Компьютерная модель. На третьем этапе необходимо формализованную информационную модель преобразовать в компьютерную модель, т. е. выразить её на понятном для компьютера языке. Существуют два пути решения этой задачи:

- создание проекта на одном из языков программирования;
- построение компьютерной модели с использованием некоторого приложения, например электронных таблиц.

В процессе создания компьютерной модели полезно разработать удобный графический интерфейс, который позволит визуализировать формальную модель, а также реализовать интерактивный диалог человека с компьютером на этапе исследования модели.

Компьютерный эксперимент. Четвёртый этап исследования информационной модели состоит в проведении компьютерного эксперимента. Если компьютерная модель существует в виде программы на одном из языков программирования, её нужно запустить на выполнение и получить результаты.

Если компьютерная модель исследуется в приложении, например в электронных таблицах, можно провести сортировку или поиск данных, построить диаграмму или график и т. д.

Анализ полученных результатов и корректировка исследуемой модели. Пятый этап представляет собой анализ полученных результатов и корректировку исследуемой модели. В случае отличия результатов, полученных при исследовании информационной модели, от измеренных параметров реальных объектов можно сделать вывод, что на предыдущих этапах построения модели были допущены ошибки или неточности.

Например, при построении описательной качественной модели могут быть неправильно отобраны существенные свойства объектов, в процессе формализации могут быть допущены ошибки в формулах и т. д. В таких случаях необходимо провести корректировку модели, причём уточнение модели может проводиться многократно, пока анализ результатов не покажет их соответствие изучаемому объекту.



Вопросы и задания

Как вы считаете, в каких случаях могут быть опущены отдельные этапы построения и исследования модели? Приведите известные вам примеры создания моделей в процессе изучения физики, химии, биологии, математики, географии и других предметов.

2.6. Инструменты программирования для разработки и исследования моделей

2.6.1. Понятие массивов



Типы массивов и объявление массива. Массив является набором переменных одного типа, объединённых одним именем. Массивы, как и переменные, могут быть различных типов: **числовые**, **строковые** и т. д.

Массив состоит из пронумерованной последовательности элементов. Номера в этой последовательности называются **индексом**; индекс может принимать целочисленные значения. Каждый из этих элементов является переменной, т. е. обладает именем и значением, и поэтому массив можно назвать **переменной с индексом**.

Массив может быть одномерным или многомерным. **Размерность** массива соответствует числу индексов, необходимых для идентификации отдельного элемента. Можно задать до 32 индексов, хотя случаи использования более трёх индексов очень редки.

Например, одномерный строковый массив, содержащий 33 буквы русского алфавита, можно представить себе в виде табл. 2.2.

Таблица 2.2

Одномерный строковый массив

Индекс	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Элементы массива	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п
Индекс	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	
Элементы массива	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я	

Индекс может принимать целочисленные значения от минимального до максимального индекса (в данном массиве от 0 до 32). Каждый элемент массива обладает собственным значением (в данном массиве значением элемента с индексом 4 является

строка "д"). Размер массива определяется количеством элементов в массиве (в данном массиве 33 элемента).

Объявление массива производится аналогично объявлению переменных, необходимо только дополнительно указать максимальный индекс или диапазон изменения индекса. Объявим одномерный целочисленный массив, содержащий 10 элементов, и одномерный строковый массив, содержащий 33 элемента.

Язык Visual Basic .NET¹⁾

```
Dim N(9) As Byte, B(32) As String
```

Язык Visual C#

```
int[] N = new int[10];  
String[] B = new String[33];
```

Язык Lazarus

```
N: array[0..9] of integer;  
B: array[0..32] of string;
```

Обращение к элементу массива производится по его имени, состоящему из имени массива и значения индекса, например N(5) в языке Visual Basic .NET.

Заполнение массива. Для начала работы с массивом необходимо его предварительно заполнить, т. е. присвоить элементам массива определённые значения. Это можно сделать различными способами:

- заполнить массив случайными числами;
- заполнить массив символами с клавиатуры;
- заполнить массив из файла.

Практическая работа 2.1

Исследование процесса изменения температуры средствами программирования с использованием заполнения массива

Задание. Заполнить числовой массив показаниями температуры воздуха за десять дней (показания температуры целые числа, вводятся с клавиатуры).

Электронное приложение к главе 2.

1) Набранные тексты этой и следующих программ размещены в электронном приложении к главе 2.





Создание проекта на языке Visual Basic .NET

1. Поместить на форму (рис. 2.2):

- список `ListBox1`, который будет использоваться для вывода целочисленного массива, заполненного показаниями температуры;
- кнопку `Button1` для создания обработчика события.

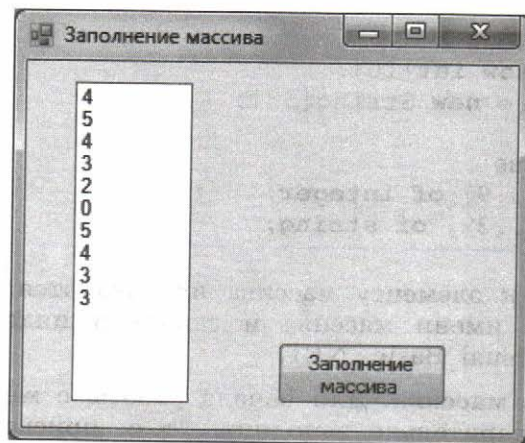


Рис. 2.2

Для ввода элементов массива с клавиатуры можно использовать функцию `InputBox()`. Аргументами этой функции являются две строки, которые отображаются в диалоговом окне ввода, а значением функции — символ, введенный пользователем.

2. Создать обработчик события для заполнения массива с клавиатуры на языке Visual Basic .NET:

```
Dim A(9), I As Byte
Private Sub Button1_Click(sender As Object,
    e As EventArgs) Handles Button1.Click
    For I = 0 To 9
        A(I) = InputBox("Введите элемент массива",
            "Ввод элементов массива с клавиатуры")
        ListBox1.Items.Add(A(I))
    Next I
End Sub
```


3. Запустить проект (см. рис. 2.2).

Щёлкнуть по кнопке *Заполнение массива*, в список будут помещены элементы массива, заполненного числами, введенными с клавиатуры.

Если не все элементы массива помещаются в список, то появляется вертикальная полоса прокрутки.



Создание проекта на языке Visual C#

1. Поместить на форму (рис. 2.2):

- список `ListBox1`, который будет использоваться для вывода целочисленного массива, заполненного показаниями температуры;
- кнопку `Button1` для создания обработчика события.

Для ввода элементов массива с клавиатуры можно использовать функцию `InputBox()`. Аргументами этой функции являются две строки, которые отображаются в диалоговом окне ввода, а значением функции — символ, введенный пользователем.

На языке Visual C# для ввода элементов массива с клавиатуры с использованием функции `InputBox()` необходимо подключить и использовать пространство имён `Microsoft.VisualBasic`¹⁾.

2. Ввести команду [*Проект—Добавить ссылку...*].

В появившемся диалоговом окне *Менеджер ссылок—WindowsFormsApplication5* на вкладке *Сборки* из списка *Платформа* выбрать имя компонента *Microsoft Visual Basic*, поставить около компонента галочку и щёлкнуть по кнопке *ОК*.

3. В *Обозревателе решений* в разделе ссылок *References* добавится ссылка *Microsoft.VisualBasic* на соответствующее пространство имен.

Теперь можно добавить ссылку на пространство имён и обработчик события в программный код. Необходимо учесть, что параметрами функции `InputBox()` также являются значение функции по умолчанию и координаты верхнего левого угла диалогового окна функции.

¹⁾ Пространство имён подключается один раз при написании проекта в установленной программе.



4. Создать обработчик события для заполнения массива с клавиатуры на языке Visual C#:

```
using Microsoft.VisualBasic;
...
int[] A = new int[10];
int i;

private void button1_Click(object sender, EventArgs
e)
{
    for (i = 0; i < 10; i++)
    {
        A[i] = Convert.ToInt16 (Interaction.InputBox
            ("Введите элемент массива", "Ввод элементов
            массива с клавиатуры", "", 500, 500));
        listBox1.Items.Add(Convert.ToString(A[i]));
    }
}
```

5. Запустить проект (см. рис. 2.2).

Щёлкнуть по кнопке *Заполнение массива*, в список будут выведены элементы массива, заполненного числами, введёнными с клавиатуры.

Если не все элементы массива помещаются в список, то появляется вертикальная полоса прокрутки.



Создание проекта на языке Lazarus



1. Поместить на форму (рис. 2.2):

- список `ListBox1`, который будет использоваться для вывода целочисленного массива, заполненного показаниями температуры;
- кнопку `Button1` для создания обработчика события.

Для ввода элементов массива с клавиатуры можно использовать функцию `InputBox()`. Аргументами этой функции являются две строки, которые отображаются в диалоговом окне ввода, а значением функции — символ, введённый пользователем.

2. Создание обработчика события заполнения массива с клавиатуры на языке Lazarus:



```
var A: array[0..9] of integer;
    i: integer;
procedure TForm1.Button1Click(Sender: TObject);
begin
    ListBox1.Items.Clear();
    for i := 0 to 9 do
        begin
            A[i] := StrToInt(InputBox('Ввод элементов массива
                с клавиатуры', 'Введите элемент массива',
                ''));
            ListBox1.Items.Add(IntToStr(A[i]));
        end;
    end;
```

3. Запустить проект (см. рис. 2.2).

Щёлкнуть по кнопке *Заполнение массива*, в список будут выведены элементы массива, заполненного числами, введенными с клавиатуры.

Если не все элементы массива помещаются в список, то появляется вертикальная полоса прокрутки.

2.6.2. Другие составные типы данных

Массив, представляющий собой набор переменных, принято считать **составным типом данных**. Кроме уже рассмотренных **одномерных массивов**, элементы которых имеют только один индекс (порядковый номер) и расположены в памяти компьютера последовательно друг за другом, существуют и другие составные типы данных, имеющие иную структуру.

Матрица (двумерный массив) также представляет собой набор переменных одного типа. Но в этом случае каждая переменная — элемент массива имеет два индекса, а структура получаемого набора данных аналогична таблице (рис. 2.3).

		1	2	...	$n-1$	n	— Номер столбца (второй индекс)
Номер строки (первый индекс)	1	a_{11}	a_{12}	...	$a_{1\ n-1}$	$a_{1\ n}$	
	2	a_{21}	a_{22}	...	$a_{2\ n-1}$	$a_{2\ n}$	
	...	...	...	...	...	...	
	$n-1$	$a_{n-1\ 1}$	$a_{n-1\ 2}$	...	$a_{n-1\ n-1}$	$a_{n-1\ n}$	
	n	$a_{n\ 1}$	$a_{n\ 2}$	...	$a_{n\ n-1}$	$a_{n\ n}$	

Рис. 2.3

Матрица, как и таблица, состоит из некоторого количества строк и столбцов, образующих ячейки — элементы матрицы. При этом каждый элемент обозначается двумя индексами, первый из которых — это номер строки, а второй — номер столбца. Например, если матрица имеет имя M , то её элемент $M[3, 4]$ располагается в четвёртом столбце третьей строки:



Подобно точкам на плоскости, элементы матрицы имеют две «координаты», обозначаемые индексами, поэтому матрицу называют двумерным массивом. Возможны и массивы больших размерностей: например, в трёхмерном массиве каждый элемент обозначен тремя индексами.

Статические и динамические структуры данных. Обычно массив задаётся (описывается) в компьютерной программе в самом её начале, где указывается количество и тип элементов этого массива. В соответствии с этим в памяти компьютера выделяется область для хранения массива. При исполнении программы количество элементов массива (равно как и размеры выделенной под его хранение области в памяти) не изменяется. Такую неизменяющуюся в процессе работы программы структуру данных называют **статической**. Однако в некоторых ситуациях количество обрабатываемых данных может быть заранее неизвестно. В подобном случае можно задать размеры массива заведомо большими, чем возможное количество ожидаемых данных, реально используя (заполняя) только часть этого массива. Но это приводит к нерациональному расходованию ресурса памяти компьютера. Более экономичным решением является изменение размеров массива (его «наращивание») по мере необходимости уже в процессе работы программы, когда при поступлении очередного значения к массиву добавляются новые элементы, а в памяти выделяется дополнительное место для их хранения. Такие структуры данных, изменяемые в процессе работы программы, называют **динамическими**.

Цепочки данных. В массиве все элементы остаются независимыми друг от друга, они лишь собраны для удобства в единый набор по их смыслу, — как, например, листки анкет учеников могут быть собраны в одной папке. Нумерация элементов массива с помощью индексов позволяет обратиться к требуемому элементу, но порядок следования элементов в массиве в большинстве

случаев не важен. Скажем, удобно, когда хранящиеся в папке анкеты учеников расположены по алфавиту по фамилиям, но если порядок следования анкет в папке будет произвольным, то это не нарушит возможности пользования каждой отдельной анкетой. Однако в некоторых случаях важным является именно порядок следования элементов друг за другом: например, перестановка страниц в книге приведёт к искажению её текста.

Набор данных, в котором отдельные переменные — элементы обязательно должны следовать друг за другом в определённом порядке, называют цепочкой данных. Цепочка представляет собой конечную последовательность данных: она имеет начало (которым считается самый первый элемент цепочки), а в качестве конца цепочки используется особо оговорённое специальное значение, не встречающееся среди данных (например, значение `Nil` в языке Паскаль или `Null` — в языке Си). В более простых задачах может быть оговорено, что признаком конца цепочки данных является нулевое значение либо отрицательное число (если сами данные представляют собой положительные числа). При обработке цепочки данных требуемая операция поочерёдно выполняется над каждым элементом, пока не будет обнаружен конец цепочки, и тогда вычисления прекращаются.

Как правило, количество данных в цепочке заранее неизвестно, поэтому для её хранения и обработки необходимо использовать динамические структуры данных. При этом нужно учитывать, что для очередного элемента цепочки данных свободное место в памяти не обязательно отыщется сразу после предыдущего элемента. Скорее всего, компьютер сможет выделить под новый элемент цепочки некоторую произвольную ячейку памяти, которая на этот момент оказалась свободной. А значит, необходимо предусмотреть механизм, который обеспечивал бы указание: в какой именно ячейке памяти расположен следующий элемент цепочки.

Примером реализации хранения и обработки цепочки данных является **список**.

Началом (или, как ещё говорят, «головой») списка является его первый элемент, который содержит в себе два поля: **поле значения** и **поле указателя** на следующий элемент списка. Из них поле указателя как раз и содержит адрес ячейки памяти, в которой размещается следующий элемент списка.

В только что созданном (пустом) списке есть только один свободный элемент, поле значения которого остается пустым, а поле указателя содержит значение конца цепочки (в языке Паскаль — специальное значение `Nil` (рис. 2.4).

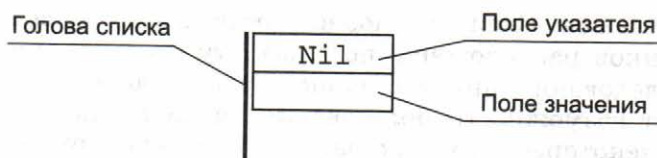


Рис. 2.4

При добавлении в пустой список первого значения выполняются следующие действия:

- само добавляемое значение записывается в поле значения свободного элемента (и тем самым он перестаёт быть свободным);
- в памяти выделяется место для ещё одного элемента списка, в его поле указателя записывается значение Nil, а адрес вновь выделенной ячейки памяти заносится в поле указателя заполненного первого элемента.

Теперь список состоит из двух элементов, из которых первый элемент хранит первое данное, а второй элемент является свободным; при этом записанное в его поле указателя значение Nil является признаком конца списка (рис. 2.5).



Рис. 2.5

Аналогичные действия с записью очередного данного, выделением нового свободного элемента списка и его связывания с предыдущим элементом через поле указателя выполняются и при дальнейшем заполнении списка. В результате образуется цепочка последовательно связанных друг с другом элементов списка, — например, как на рис. 2.6.

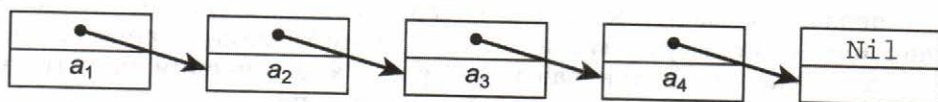


Рис. 2.6

Если требуется удалить какой-либо элемент из списка, достаточно изменить значение поля указателя в предшествующем ему

элементе, «перенацелив» это значение на последующий после него элемент (рис. 2.7).

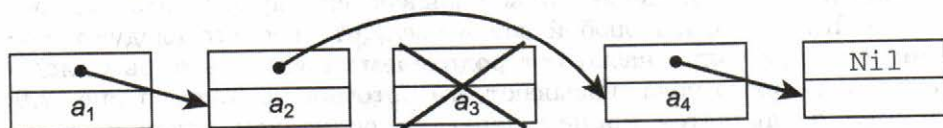


Рис. 2.7

Чтобы добавить новый элемент в середину списка, достаточно выделить для него новую ячейку памяти, а затем установить значения полей указателя в новом и в предшествующем ему элементах так, чтобы обеспечить непрерывность списка (рис. 2.8).

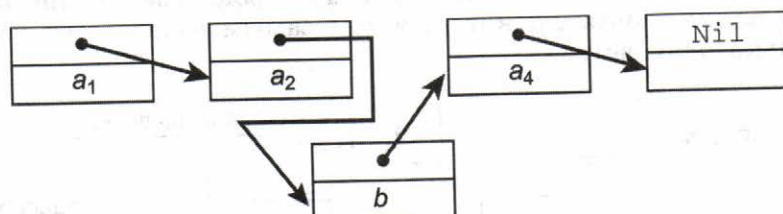


Рис. 2.8

Важным отличием списка от массива является принцип доступа к их элементам. В массиве возможен непосредственный (прямой) доступ к любому элементу по его индексу (индексам). В списке же возможен только последовательный доступ — список просматривается, начиная с его «головы», элемент за элементом по порядку, заданному значениями полей указателя, пока не будет найден нужный элемент. После этого найденный элемент может быть считан, удалён, изменён или же после него может быть вставлен новый элемент.

В списке после каждого элемента (кроме конечного) может следовать только один элемент. Такая структура данных называется **линейной односвязной**. В другом случае структура может быть разветвлённой: после некоторого элемента могут следовать два или более последующих элемента. Такую структуру данных называют **деревом** и обычно изображают графически, как на рис. 2.9.

Элементы дерева принято называть его **узлами**. Самый первый элемент дерева называют **корнем**. Связи между узлами называются **рёбрами**. Цепочки, образуемые последовательностями взаим-

мосвязанных узлов, называют **ветвями** дерева, а конечные узлы ветвей (после которых не следует ни одного узла) называются **листьями**. Все остальные узлы называются **внутренними узлами** дерева. Кроме того, в любой ветви узел, от которого следуют связанные с ним узлы, называют **родителем**, а узлы, которые следуют от некоторого узла, называют его **потомками**. Так, на рис. 2.9:

- узел a_1 является корнем дерева и родителем узлов a_2 и a_3 , которые образуют две ветви дерева;
- узел a_2 является потомком узла a_1 и родителем узлов a_4 , a_5 и a_6 , которые образуют от узла a_2 три ветви;
- узлы a_4 и a_5 представляют собой листья;
- аналогично, узел a_3 является потомком узла a_1 и родителем узлов a_7 и a_8 , которые представляют собой листья и образуют от узла a_3 две ветви;
- узел a_6 является потомком узла a_2 и родителем единственного узла — листа a_9 , который тем самым образует от узла a_6 только одну ветвь.

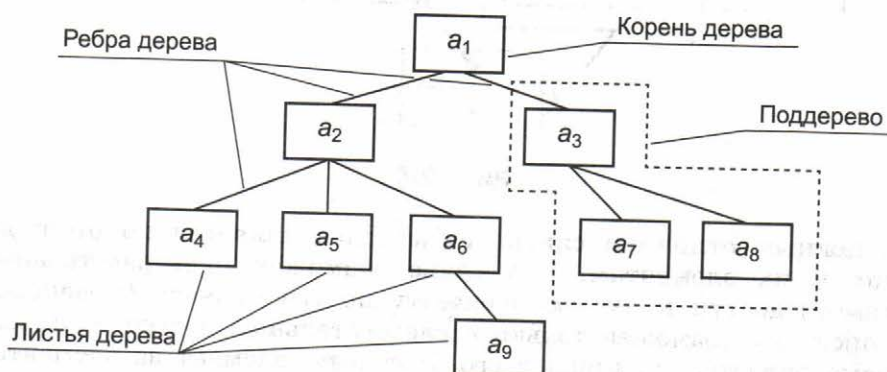


Рис. 2.9

Любая часть дерева, начинающаяся с некоторого внутреннего узла, называется **поддеревом** и может рассматриваться как самостоятельное дерево.

Существует несколько специфических разновидностей дерева, отличающихся особенностями их структуры и принципами обработки содержащихся в них данных. Из них мы отметим **бинарные деревья**, в которых каждый узел (кроме листьев) имеет ровно два потомка.

Дерево как графически изображаемый набор значений (узлов), связанных с другими узлами при помощи рёбер, является одной из разновидностей графа.

Элементы графа называют **узлами**, или **вершинами** графа, а взаимосвязи между ними называются **рёбрами**. Если рёбрам графа сопоставлены какие-либо числовые значения (например, длины дорог между городами, обозначенными узлами графа), то эти числа называют **весами** рёбер. Соответственно, граф, рёбрам которого назначены значения весов, называют **взвешенным**.

Обычно рассматриваются графы, в которых два каких-либо узла связаны только одним ребром и в которых не существует **петель** — рёбер, исходящих и заканчивающихся в одном и том же узле. Граф, содержащий петли и/или кратные рёбра, называют **псевдографом** (рис. 2.10).

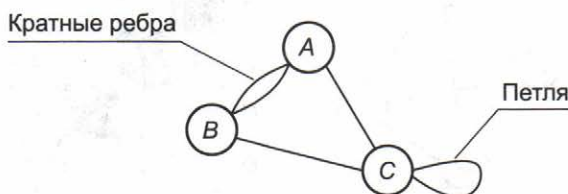


Рис. 2.10

Путь в графе — это конечная непрерывная последовательность узлов, соединённых друг с другом рёбрами. **Циклом** называют путь, который начинается и заканчивается в одном и том же узле. **Длиной пути** (или цикла) называют количество составляющих его рёбер.

В **неориентированном** графе связи между любыми узлами являются двунаправленными, т. е. эти узлы «равноправны» по отношению к этой связи.

В **ориентированном** графе (**орграфе**) связи между узлами являются направленными (рис. 2.11). Рёбра ориентированного графа называют **дугами**, а пути в нём — **маршрутами**. Ориентированный граф, в котором каждая пара узлов связана только одной дугой, называют **направленным**; в отличие от него в простом ориентированном графе какие-то вершины могут быть соединены двумя дугами, имеющими противоположные направления.

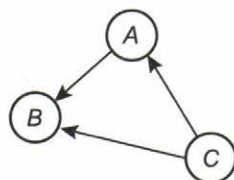


Рис. 2.11

Способы представления графов. Граф может быть представлен несколькими различными способами, из которых чаще всего используются:

- графический способ (наиболее наглядный, см. рис. 2.10, 2.11);

- **список рёбер** — перечисление всех рёбер графа как пар обозначений связываемых этими рёбрами узлов (например, список $\{A, B\}, \{A, D\}, \{A, C\}, \{B, C\}, \{C, D\}$ определяет граф, показанный на рис. 2.12);
- **матрица смежности** — квадратная симметричная таблица (матрица), в которой столбцы и строки соответствуют узлам графа, а в ячейках на их пересечении записываются числа 0 или 1, обозначающие наличие или отсутствие связей между соответствующими парами узлов (рис. 2.12).

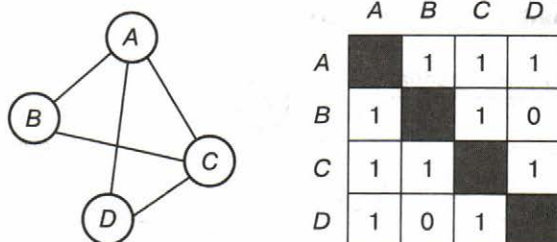


Рис. 2.12

Для взвешенного графа в ячейках матрицы смежности записываются значения весов, наличие которых указывает и на наличие ребра между соответствующими узлами (рис. 2.13). Если же какие-то два узла не связаны, то соответствующая ячейка матрицы смежности остается пустой. Ячейки для одноимённых строк и столбцов, обычно закрашивают и исключают из рассмотрения.

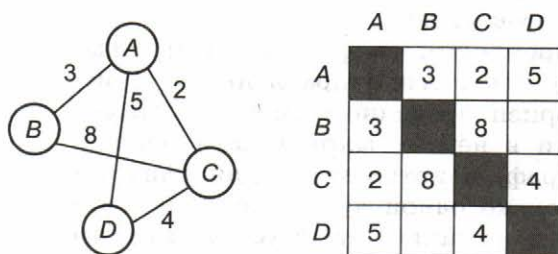


Рис. 2.13

Следует заметить, что для неориентированного графа матрица смежности симметрична относительно диагонали, образованной в ней закрашенными ячейками, так как, например, ребро между узлами A и B и его вес на рис. 2.13 те же, что между узлами B и A. Поэтому при решении задач с графами, заданными ма-

трицей смежности, достаточно рассматривать только часть этой матрицы над её диагональю из закрашенных ячеек.

Решение задач, связанных с анализом графов

Рассмотрим часто встречающиеся типы задач, связанных с анализом графов.

Сопоставление графа с матрицей смежности. В подобной задаче требуется определить, какая матрица смежности (из нескольких предполагаемых вариантов) соответствует заданному графу либо какой граф соответствует заданной матрице смежности. Это одна из наиболее простых задач. Для её решения достаточно сверить числовые значения в матрице смежности с имеющимися в графе рёбрами и их весами.

Поиск кратчайшего маршрута. Задана (в виде графа или матрицы смежности) схема путей между некоторыми населёнными пунктами с указанием длины каждого пути в виде веса соответствующего ребра. Требуется определить кратчайший путь между двумя заданными пунктами. При этом надо учитывать, что движение между ними возможно по разным рёбрам, в том числе в обход тех или иных узлов, но визуально по имеющемуся изображению графа «более короткий» путь может иметь большую длину, чем сумма нескольких более коротких путей.

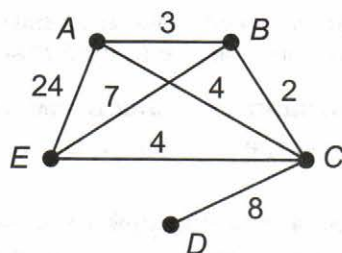
Пример 1

Задана таблица, в которой указаны длины путей между населёнными пунктами, обозначенными буквами. Определить длину кратчайшего пути от пункта А до пункта Е.

	A	B	C	D	E
A		3	4		24
B	3		2		7
C	4	2		8	4
D			8		
E	24	7	4		

Решение

Поскольку в условии задачи дана таблица (матрица смежности), прежде всего необходимо построить по ней граф.



Очевидно, поскольку пункт D является тупиком, его (как и ребро CD) можно игнорировать.

Теперь требуется найти и «измерить» все возможные пути из пункта A в пункт E :

$$AE = 24,$$

$$ABE = 3 + 7 = 10,$$

$$ABCE = 3 + 2 + 4 = 9,$$

$$ACE = 4 + 4 = 8.$$

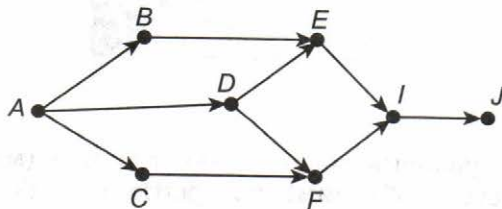
Остается найти кратчайший среди этих путей: это путь ACE , длина которого равна 8.

Ответ: 8.

Поиск количества различных путей в ориентированном графе. Задан ориентированный граф, определяющий сеть дорог между некоторыми населёнными пунктами, обозначенными буквами (движение между ними допустимо только в одном направлении, указанном стрелками). Требуется определить количество различных путей из одного пункта в другой. При этом пути считаются различными, если отличаются хотя бы на одну дугу.

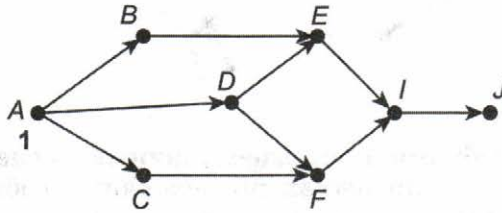
Пример 2

Задана сеть дорог между населёнными пунктами. Движение по этим дорогам возможно только в направлении, указанном стрелками. Требуется определить количество возможных путей из пункта A в пункт J .

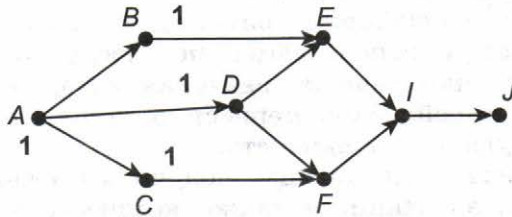


Решение

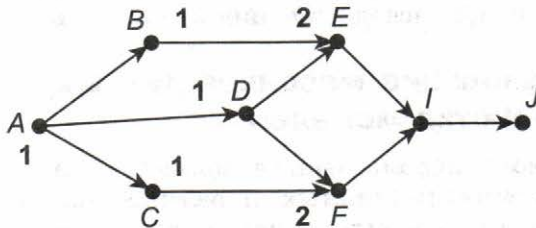
Прежде всего, учитываем, что из пункта A в пункт J точно есть по крайней мере один путь. Обозначим этот факт цифрой 1 рядом с пунктом A .



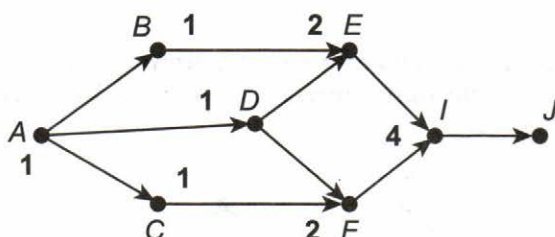
Из пункта A записанная для него единица «передвигается» во все соседние пункты, в которые ведут стрелки из пункта A , т. е. в пункты B , C и D .



Теперь обратим внимание на пункт E : в него по соответствующим стрелкам «приходят» единица из пункта B и единица из пункта D . Эти единицы нужно сложить (это показывает, что из пункта A в пункт D возможны два различных пути — ABE и ADE). Аналогичная ситуация — в пункте F .



Соответственно, в пункт I «приедут» две двойки — из пункта E и пункта F . Тогда рядом с пунктом I нужно записать их сумму — число 4. Наконец, в пункт J по единственной ведущей к нему стрелке «приходит» эта четвёрка из пункта I .



Ответ: 4.

Случайные события и псевдослучайные последовательности. Во многих реальных процессах происходящие события определяются случайностями. Например, процесс обслуживания в супермаркете зависит от количества покупателей, приходящих с товарами для их оплаты на кассы, которое изменяется случайным образом. Случайными можно считать и параметры, для которых неизвестна закономерность их изменения.

При моделировании подобных процессов в модели тоже требуется отражать случайность изменения соответствующих параметров. Для этого используются **псевдослучайные последовательности** — цепочки данных, значения которых изменяются в соответствии с особой закономерностью, в как можно большей степени имитирующей случайность.

Закономерность, позволяющая получить псевдослучайную последовательность значений, а также компьютерную программу, вычисляющую такие значения, называют **генератором случайных чисел**. Обычно в языке программирования имеется специальный оператор, позволяющий получить очередное числовое значение из генерируемой псевдослучайной последовательности. Кроме того, обычно предусматривается особый оператор инициализации генератора случайных чисел, при срабатывании которого соответствующая компьютерная программа переводится в исходное состояние для генерации новой псевдослучайной последовательности.

2.6.3. Использование массивов данных в разработке моделей

Для полноценного использования массивов при разработке моделей для решения различных практических задач обработки данных необходимо освоить типовые алгоритмы работы с массивами.

Вычисление суммы элементов массива

Для суммирования элементов массива необходимо в цикле перебирать последовательно все элементы массива с первого до

последнего и *прибавлять* значение каждого очередного элемента массива к значению заранее выделенной переменной. *Изначально* в эту переменную записывается *нулевое значение* (не влияющее на последующее вычисление суммы). После завершения работы цикла в этой переменной будет *накоплена сумма* всех элементов массива. Ниже алгоритм на псевдокоде (смесь языка программирования и естественного языка):

```
// задан массив M размером N
SUM = 0 // начальное значение суммы
ЦИКЛ ДЛЯ I ОТ 1 ДО N С ШАГОМ 1
    SUM = SUM + M(I)
КОНЕЦ ЦИКЛА
ВЫВЕСТИ SUM
```

Аналогичным способом выполняется суммирование значений элементов массива, удовлетворяющих некоторому условию (например, только положительных или только чётных). В этом случае внутри цикла размещается оператор ветвления, условие которого проверяет текущий элемент массива, и суммирование выполняется, только если это условие истинно. Пример:

```
ЦИКЛ ДЛЯ I ОТ 1 ДО N С ШАГОМ 1
    ЕСЛИ M(I) > 0 ТО SUM = SUM + M(I)
КОНЕЦ ЦИКЛА
```

Вычисление произведения элементов массива

Выполняется аналогично вычислению суммы, но в переменную, в которой будет накапливаться произведение, *изначально записывается значение 1* (не влияющее на произведение).

```
// задан массив M размером N
PR = 1 // изначально значение произведения
ЦИКЛ ДЛЯ I ОТ 1 ДО N С ШАГОМ 1
    PR = PR * M(I)
КОНЕЦ ЦИКЛА
ВЫВЕСТИ PR
```

При вычислении произведения элементов массива, удовлетворяющих некоторому условию, так же как и для суммы, используется оператор ветвления, проверяющий соответствие очередного элемента массива заданному условию. Пример:

```
ЦИКЛ ДЛЯ I ОТ 1 ДО N С ШАГОМ 1
    ЕСЛИ M(I) > 0 ТО PR = PR * M(I)
КОНЕЦ ЦИКЛА
```

Поиск в массиве

Поиск в массивах обычно реализуется в форме поиска индекса элемента массива, для которого значение элемента совпадает с заданным. В числовых массивах часто производится поиск минимального или максимального элемента.

Поиск максимального/минимального значения элемента массива

Вспомним рассмотренный ранее алгоритм поиска наибольшего/наименьшего числа из нескольких заданных чисел путём последовательного сравнения чисел с предполагаемым максимумом/минимумом. Этот же алгоритм может быть с ещё большим удобством реализован для массива при помощи цикла.

```
// задан массив M размером N
MAX = M(1) // изначально предполагаемый максимум
           // равен первому элементу массива
ЦИКЛ ДЛЯ I ОТ 1 ДО N С ШАГОМ 1
    ЕСЛИ M(I) > MAX ТО MAX = M(I)
КОНЕЦ ЦИКЛА
ВЫВЕСТИ MAX
```

Аналогично выполняется поиск минимума в массиве.

```
// задан массив M размером N
MIN = M(1) // изначально предполагаемый минимум
           // равен первому элементу массива
ЦИКЛ ДЛЯ I ОТ 1 ДО N С ШАГОМ 1
    ЕСЛИ M(I) < MIN ТО MIN = M(I)
КОНЕЦ ЦИКЛА
ВЫВЕСТИ MIN
```

Более сложной является задача, в которой нужно найти максимальное либо минимальное значение среди элементов массива, удовлетворяющих некоторому условию (например, среди чётных элементов). Как и при вычислении суммы или произведения элементов, удовлетворяющих заданному условию, нужно использовать в операторе ветвления дополнительное условие, проверяющее очередной элемент. Однако в этом случае мы не можем использовать в качестве предполагаемого максимума (либо минимума) первый элемент массива, так как заранее не знаем, будет ли он чётным.

В подобном случае можно использовать один из двух возможных способов:

- если заранее известен диапазон возможных значений элементов массива, то в качестве предполагаемого исходного значения берётся:

- для предполагаемого максимума — число, меньшее, чем самое маленькое из возможных значений элементов массива;
 - для предполагаемого минимума — число, большее, чем самое большое из возможных значений элементов массива;
- б) можно дополнить условия в операторе ветвления дополнительным условием — «ключом», которое должно сработать при первом обнаружении в массиве элемента, который удовлетворяет заданному условию.

Рассмотрим пример алгоритма поиска максимума среди нечётных натуральных чисел. В этом случае мы точно знаем, что элементы массива могут иметь значения из диапазона $[1, +\infty)$.

```
// задан массив M размером N
MAX = 0 // изначально предполагаемый максимум равен
        // нулю (числу, меньшему самого маленького
        // значения элементов массива)
ЦИКЛ ДЛЯ I ОТ 1 ДО N С ШАГОМ 1
    ЕСЛИ (M(I) > MAX) И (M(I) MOD 2 <> 0) ТО MAX = M(I)
КОНЕЦ ЦИКЛА
ВЫВЕСТИ MAX
```

Поиск второго по величине наибольшего/наименьшего значения элемента массива

Отличие от задачи поиска максимума/минимума в этой задаче заключается в том, что нужно одновременно отслеживать и «основной» максимум/минимум, и «следующий по порядку» второй максимум/минимум. При поиске «второго максимума» очередной элемент массива нужно сравнивать и с предполагаемым «основным» максимумом, и с предполагаемым «вторым» максимумом:

- если очередной элемент массива больше «основного» максимума, то значение этого элемента принимается в качестве нового «основного» максимума, а бывшее значение «основного» максимума становится новым «вторым» максимумом;
- если очередной элемент массива меньше «основного» максимума, но больше «второго» максимума, то значение этого элемента принимается в качестве нового «второго» максимума (значение «основного» максимума не изменяется).

```
// задан массив M размером N
ЕСЛИ M(1) > M(2) ТО MAX1 = M(1), MAX2 = M(2)
    ИНАЧЕ MAX1 = M(2), MAX2 = M(1)
// определяем начальные значения предполагаемых
// "основного" и "второго" максимумов
ЦИКЛ ДЛЯ I ОТ 3 ДО N С ШАГОМ 1 // рассматриваем
                                // остальные элементы
```

ЕСЛИ $M(I) > MAX1$ ТО $MAX2 = MAX1$, $MAX1 = M(I)$
 ЕСЛИ $(M(I) \leq MAX1)$ И $(M(I) > MAX2)$ ТО $MAX2 = M(I)$
 КОНЕЦ ЦИКЛА
 ВЫВЕСТИ $MAX2$

Поиск «второго минимума» производится аналогично, но знаки неравенств в условиях в операторах ветвления нужно изменить на противоположные.

Сортировка массива

Под сортировкой числового массива понимается процесс перестановки числовых значений элементов массива, целью которого является размещение элементов массива по возрастанию (или по убыванию).

Пусть у нас имеется целочисленный массив, состоящий из десяти элементов и заполненный случайными числами от 0 до 100. Представим массив и процесс его сортировки в виде таблицы (табл. 2.3). Значения индекса элементов массива разместим в первом вертикальном столбце таблицы, а значения элементов — во втором вертикальном столбце.

Таблица 2.3

Алгоритм сортировки массива по убыванию

Индекс	Массив	Шаг сортировки								
		1	2	3	4	5	6	7	8	9
0	44	84	84	84	84	84	84	84	84	84
1	9	9	79	79	79	79	79	79	79	79
2	39	39	39	69	69	69	69	69	69	69
3	69	69	69	39	46	46	46	46	46	46
4	41	41	41	41	41	44	44	44	44	44
5	84	44	44	44	44	41	41	41	41	41
6	46	46	46	46	39	39	39	41	41	41
7	79	79	9	9	9	9	9	9	39	39
8	41	41	41	41	41	41	41	39	9	15
9	15	15	15	15	15	15	15	15	15	9

Алгоритм сортировки массива по убыванию состоит в следующем:

- 1) На первом шаге сортировки провести поиск максимального элемента в массиве с 0 по 9 элемент. Поменять местами найденный максимальный элемент с элементом, имеющим индекс, равный нулю. Если максимальный элемент имеет индекс 0, то оставить его на месте.
- 2) На втором шаге сортировки провести поиск максимального элемента с 1 по 9 элемент. Поменять местами найденный максимальный элемент с элементом, имеющим индекс, равный единице.
- 3) Повторить шаги сортировки ещё семь раз. На каждом шаге осуществить поиск максимального элемента среди оставшихся неупорядоченных элементов и сделать перестановку элементов.

В таблице каждый шаг сортировки массива разместим в вертикальном столбце, отображающем порядок значений элементов массива. Перестановку значений элементов массива отобразим стрелками.

Вопросы и задания

1. Обоснуйте, почему при вычислении суммы элементов массива в отведенную для этого переменную нужно записать нуль, а при вычислении произведения элементов массива — единицу.
- *2. Укажите, какие изменения нужно внести в алгоритмы вычисления суммы и произведения элементов массива, если требуется сделать это только для указанной части массива (элементов с индексами из заданного диапазона).
- *3. Подумайте и укажите, как можно оптимизировать приведённые выше алгоритмы вычисления суммы и произведения элементов массива, уменьшив количество выполняемых в цикле сложений/умножений на одно.
4. Реализуйте алгоритм поиска минимума в массиве для элементов, удовлетворяющих условию делимости на 3.
5. Выполните трассировку описанных программ поиска максимума и минимума в массивах (в том числе для элементов удовлетворяющих предложенному вами условию) на примере нескольких конкретных массивов.
- *6. Подумайте и объясните, почему при поиске максимума нужно в качестве исходного предполагаемого максимума брать число, меньшее самого маленького из возможных в массиве значений элементов, а при поиске минимума — брать число, большее самого большого из возможных в массиве значений элементов. Обоснуйте своё мнение, проведя трассировку программ для соответствующих примеров массивов.

7. Реализуйте алгоритм поиска в массиве «второго минимума».
8. Выполните трассировку программы поиска в массиве «второго» максимума/минимума. Объясните смысл операций перезаписи значений предполагаемых максимумов/минимумов в операторе
ЕСЛИ $M(I) > MAX1$ **ТО** $MAX2 = MAX1$, $MAX1 = M(I)$.



Практическая работа 2.2

Проведение исследования на упорядочение и поиск экстремальных значений потока информации о температуре воздуха

Задание. Заполнить числовой массив показаниями температуры воздуха за десять дней (показания температуры целые числа, вводятся с клавиатуры). Найти максимальное значение температуры и в какой день была самая высокая температура. Осуществить упорядочение значений температуры по убыванию.



Электронное приложение к главе 2.



Создание проекта на языке Visual Basic .NET

1. Поместить на форму (рис. 2.14):
 - список `ListBox1` для вывода значений дня измерения температуры;
 - список `ListBox2`, который будет использоваться для вывода целочисленного массива, заполненного показаниями температуры;
 - список `ListBox3` для вывода упорядоченных значений температуры;
 - кнопку `Button1` для создания обработчика события заполнения массива;
 - кнопку `Button2` для создания обработчика события поиска максимального значения;
 - кнопку `Button3` для создания обработчика события упорядочения (сортировки) массива;
 - надпись `Label1` для вывода номера максимального значения температуры;
 - надпись `Label2` для вывода дня с максимальной температурой;
 - надписи `Label3–Label7` для вывода поясняющей информации.

Рис. 2.14

Программный модуль проекта будет содержать:

- обработчик события заполнения массива числами с клавиатуры;
 - обработчик события поиска максимального элемента и его номера.
 - обработчик события сортировки массива.
2. Объявить переменные для использования в программном модуле и создать обработчик события заполнения целочисленного массива числами с клавиатуры.

```
Dim A(9) As Byte, Max, i, j, R As Byte
```

```
Private Sub Button1_Click(sender As Object, e As  
EventArgs) Handles Button1.Click
```

```
'очистка полей списков дней измерения
```

```
'и значений элементов массива
```

```
ListBox1.Items.Clear()
```

```
ListBox2.Items.Clear()
```

```
ListBox3.Items.Clear()
```

```
'Заполнение массива числами, введенными
```

```
'с клавиатуры
```

```
For i = 0 To 9
```

```
ListBox1.Items.Add(i + 1)
```

```
A(i) = InputBox("Введите элемент массива", "Ввод  
элементов массива с клавиатуры")
```

```
ListBox2.Items.Add(A(i))
```

```
Next i
```

```
End Sub
```



3. Создать обработчик события для поиска максимального значения температуры и номера дня, в который была максимальная температура.

```
'Процедура поиска индекса
'максимального элемента массива и его номера
Private Sub Button2_Click(sender As Object, e As
    EventArgs) Handles Button2.Click
    Max = 0
    A(Max) = A(0)
    For i = 1 To 9
        If A(i) > A(Max) Then A(Max) = A(i) : Max = i
    Next i
    'Вывод максимального значения температуры,
    'и в какой день это было
    Label1.Text = A(Max)
    Label2.Text = Max + 1
End Sub
```

4. Создать обработчик события сортировки массива.

```
Private Sub Button3_Click(sender As Object, e As
    EventArgs) Handles Button3.Click
    'сортировка массива
    For i = 0 To 8
        For j = i + 1 To 9
            If A(i) < A(j) Then R = A(i) : A(i) = A(j) : A(j) = R
        Next j
    Next i
    'Вывод упорядоченного массива на экран
    For i = 0 To 9
        ListBox3.Items.Add(A(i))
    Next i
End Sub
```

5. Запустить проект (см. рис. 2.14).

Щёлкнуть по кнопке *Заполнить массив*. В первый список будут выведены дни измерения температуры воздуха, во второй список — значения элементов массива: числа, введённые пользователем с клавиатуры.

Щёлкнуть по кнопке *Поиск максимального*. На надписи будут выведены результаты поиска максимальной температуры и номер дня, когда температура воздуха была максимальной.

Щёлкнуть по кнопке *Упорядочить массив*. В третьем списке будут выведены отсортированные по убыванию значения температуры.



Создание проекта на языке Visual C#

1. Поместить на форму (см. рис. 2.14):

- список `ListBox1` для вывода значений дня измерения температуры;
- список `ListBox2`, который будет использоваться для вывода целочисленного массива, заполненного показаниями температуры;
- список `ListBox3` для вывода упорядоченных значений температуры;
- кнопку `Button1` для создания обработчика события заполнения массива;
- кнопку `Button2` для создания обработчика события поиска максимального значения;
- кнопку `Button3` для создания обработчика события упорядочения (сортировки) массива;
- надпись `Label1` для вывода максимального значения температуры;
- надпись `Label2` для вывода номера дня с максимальной температурой;
- надписи `Label3–Label7` для вывода поясняющей информации.

Программный модуль проекта будет содержать:

- обработчик события заполнения массива числами с клавиатуры;
- обработчик события поиска максимального элемента и его номера.
- обработчик события сортировки массива.

2. Подключить пространство имён *Microsoft.VisualBasic*. Для этого ввести команду [*Проект—Добавить ссылку...*]. В появившемся диалоговом окне *Менеджер ссылок—WindowsFormsApplication5* на вкладке *Сборки* из списка *Платформа* выбрать имя компонента *Microsoft Visual Basic*, поставить около компонента галочку и щёлкнуть по кнопке *ОК*.

В *Обозревателе решений* в разделе ссылок *References* добавится ссылка *Microsoft.VisualBasic* на соответствующее пространство имен.

Теперь можно добавить ссылку на пространство имён в программный код.

```
using Microsoft.VisualBasic;
```

3. Объявить переменные для использования в программном модуле и создать обработчик события заполнения целочисленного массива числами с клавиатуры.

```
int[] A = new int[10];
int i, j, Max, R;

private void button1_Click(object sender, EventArgs e)
{
    //очистка полей списков дней измерения
    //и значений элементов массива
    listBox1.Items.Clear();
    listBox2.Items.Clear();
    listBox3.Items.Clear();
    //Заполнение массива числами, введёнными
    //с клавиатуры
    for (i = 0; i < 10; i++)
    {
        listBox1.Items.Add(Convert.ToString(i+1));
        A[i] = Convert.ToInt16(Interaction.InputBox
            ("Введите элемент массива", "Ввод элементов
            массива с клавиатуры", "", 500, 500));
        listBox2.Items.Add(Convert.ToString(A[i]));
    }
}
```

4. Создать обработчик события поиска максимального значения температуры и номера дня, в который была максимальная температура.

```
private void button2_Click(object sender, EventArgs e)
{
    Max = 0;
    A[Max] = A[0];
    {
        for (i = 0; i < 10; i++)
        {
            if (A[i] > A[Max]) {A[Max] = A[i] ; Max = i;}
        }
    }
}
```



```
//Вывод максимального значения температуры,
//и в какой день это было
label1.Text = Convert.ToString(A[Max]);
label2.Text = Convert.ToString(Max + 1);
}
```

5. Создать обработчик события сортировки массива.

```
private void button3_Click(object sender, EventArgs e)
{
    //сортировка массива
    for (i = 0; i < 9; i++)
        for (j = i + 1; j < 10; j++)
            if (A[i] < A[j]) {R = A[i]; A[i] = A[j];
                            A[j] = R;}
    //вывод упорядоченного массива на экран
    for (i = 0; i < 10; i++)
        listBox3.Items.Add(Convert.ToString(A[i]));
}
```



6. Запустить проект (см. рис. 2.14).

Щёлкнуть по кнопке *Заполнить массив*. В первый список будут выведены номера дней измерения температуры воздуха, а во второй список — значения элементов массива: числа, введённые пользователем с клавиатуры.

Щёлкнуть по кнопке *Поиск максимальной*. На надписи будут выведены результаты поиска максимальной температуры и номер дня, когда температура воздуха была максимальной. Щёлкнуть по кнопке *Упорядочить массив*. В третьем списке будут выведены отсортированные по убыванию значения температуры.



Создание проекта на языке Lazarus



1. Поместить на форму (см. рис. 2.14):

- список `ListBox1` для вывода значений дня измерения температуры;
- список `ListBox2`, который будет использоваться для вывода целочисленного массива, заполненного показаниями температуры;

- список `ListBox3` для вывода упорядоченных значений температуры;
- кнопку `Button1` для создания обработчика события заполнения массива;
- кнопку `Button2` для создания обработчика события поиска максимального значения;
- кнопку `Button3` для создания обработчика события упорядочения (сортировки) массива;
- надпись `Label1` для вывода максимального значения температуры;
- надпись `Label2` для вывода номера дня с максимальной температурой;
- надписи `Label3`–`Label7` для вывода поясняющей информации.

Программный модуль проекта будет содержать:

- обработчик события заполнения массива числами с клавиатуры;
- обработчик события поиска максимального элемента и его номера.
- обработчик события сортировки массива

2. Объявить переменные для использования в программном модуле и создать обработчик события заполнения целочисленного массива числами с клавиатуры.

var

```
a: array[0..9] of integer;
i, j, Max, R :integer;
```

procedure TForm1.Button1Click(Sender: TObject);
begin

```
{очистка полей списков дней измерения
и значений элементов массива}
```

```
Listbox1.Items.Clear();
```

```
Listbox2.Items.Clear();
```

```
Listbox3.Items.Clear();
```

```
{Заполнение массива числами, введенными с клавиатуры}
```

```
For i := 0 to 9 do
```

```
begin
```

```
Listbox1.Items.Add(IntToStr(i+1));
```

```
A[i] := StrToInt(InputBox('Ввод элементов массива
с клавиатуры', 'Введите элемент массива', ''));
```

```
Listbox2.Items.Add(IntToStr(A[i]));
```

```
end;
```

```
end;
```



3. Создать обработчик события для поиска максимального значения температуры и номера дня, в который была максимальная температура.

```

procedure TForm1.Button2Click(Sender: TObject);
begin
    Max := 0;
    A[Max] := A[0];
    For i:= 1 To 9 do
        If A[i] > A[Max] Then
            begin
                A[Max] := A[i];
                Max := i;
            end;
    {Вывод максимального значения температуры,
    и в какой день это было}
    Label1.Caption := IntToStr(A[Max]);
    Label2.Caption := IntToStr(Max + 1);
end;

```



4. Создать обработчик события сортировки массива.

```

procedure TForm1.Button3Click(Sender: TObject);
begin
    {сортировка массива}
    For i := 0 To 8 do
        For j := i + 1 To 9 do
            If A[i] < A[j] Then
                begin
                    R := A[i];
                    A[i] := A[j];
                    A[j] := R;
                end;
    {вывод упорядоченного массива на экран}
    For i := 0 To 9 do
        ListBox3.Items.Add(IntToStr(A[i]));
end;

```



5. Запустить проект (см. рис. 2.14).
Щёлкнуть по кнопке *Заполнить массив*. В первый список будут выведены номера дней измерения температуры воздуха, а во второй список — значения элементов массива: числа, введённые пользователем с клавиатуры.

Щёлкнуть по кнопке *Поиск максимального*. На надписи будут выведены результаты поиска максимальной температуры и номер дня, когда температура воздуха была максимальной.

Щёлкнуть по кнопке *Упорядочить массив*. В третьем списке будут выведены отсортированные по убыванию значения температуры.

2.6.4. Использование элементов графики в разработке моделей

Графика в языках программирования Visual Basic .NET и Visual C#

Интегрированная система программирования Visual Studio для создания рисунков, рисования текста и отображения графических изображений на формах и элементах управления использует GDI (англ. *Graphics Device Interface* — графический интерфейс устройств).

Объект Graphics (Область рисования). Для использования GDI сначала необходимо создать область рисования, т. е. объект Graphics. Объект Graphics можно создать различными способами.

Первый способ используется для создания растрового изображения, которое можно сохранить как графический файл. Сначала необходимо объявить создание растрового изображения Image1 определённого размера, затем объявить создание объекта Graph1 типа Graphics из растрового изображения Image1 и, наконец, присвоить свойству Image формы или элемента управления (например, PictureBox1.Image) значение Image1.

Язык Visual Basic .NET:

```
Dim Image1 As New Bitmap(200, 200)
Dim Graph1 As Graphics = Graphics.FromImage(Image1)
PictureBox1.Image = Image1
```

Язык Visual C#:

```
Bitmap image1 = new Bitmap(200,200);
Graphics Graph1 = Graphics.FromImage(image1);
pictureBox1.Image = image1;
```

Второй способ использует событие Paint формы или элемента управления, которое происходит при их создании или обновлении. В обработчике этого события одним из аргументов является



е типа `System.Windows.Forms.PaintEventArgs`. В программном коде обработчика события можно объявить создание объекта `Graph1` типа `Graphics` как свойства аргумента `e`.

Язык Visual Basic .NET:

```
Private Sub PictureBox1_Paint(sender
As Object, e As PaintEventArgs)
Handles PictureBox1.Paint
Dim Graph1 As Graphics = e.Graphics
End Sub
```

Язык Visual C#:

```
private void pictureBox1_Paint(object sender,
PaintEventArgs e)
{
Graphics Graph1 = e.Graphics;
}
```

Перо. Объект `Pen` (Перо) определяет цвет и ширину линии рисования. В разделе объявления переменных необходимо определить имя объекта (например, `Pen1`), установить цвет (например, красный `Color.Red`) и ширину линии в пикселях (например, 3).

Язык Visual Basic .NET:

```
Dim Pen1 As New Pen(Color.Red, 3)
```

Язык Visual C#:

```
Pen Pen1 = new Pen(Color.Red, 3);
```

Кисть. Объект `Brush` (кисть) определяет цвет и стиль закрашивания прямоугольников, окружностей и других замкнутых фигур. В разделе объявления переменных необходимо определить имя объекта (например, `Brush1`) и установить тип закрашки и цвет (например, сплошная закрашка синего цвета `SolidBrush(Color.Blue)`).

Язык Visual Basic .NET:

```
Dim Brush1 As New SolidBrush(Color.Blue)
```

Язык Visual C#:

```
SolidBrush Brush1 = new SolidBrush(Color.Blue);
```



Цвет. Цвет устанавливается как значение свойства `Color`. Можно установить цвет с использованием нескольких десятков цветовых констант. Ниже приведены примеры установки зелёного цвета для объекта `Pen1` (перо) и жёлтого цвета для объекта `Brush1` (кисть).

Язык Visual Basic .NET:

```
Pen1.Color = Color.Green  
Brush1.Color = Color.Yellow
```

Язык Visual C#:

```
Pen1.Color = Color.Green;  
Brush1.Color = Color.Yellow;
```

Для установки цвета в 24-битовой палитре цветов RGB используется метод `Color.FromArgb(Red, Green, Blue)`, аргументами которого являются три числа в диапазонах 0 до 255 (интенсивности красного, зелёного и синего цветов). Например, так можно установить пурпурный цвет для кисти `Brush1`.

Язык Visual Basic .NET:

```
Brush1.Color = Color.FromArgb(255,0,255)
```

Язык Visual C#:

```
Brush1.Color = Color.FromArgb(255,0,255);
```

Цвет пера или кисти можно также установить с использованием элемента управления `ColorDialog`. Для этого данный элемент управления необходимо поместить на форму и ввести в программный код следующие строки:

Языки Visual Basic .NET:

```
ColorDialog1.ShowDialog()  
Pen1.Color = ColorDialog1.Color
```

Язык Visual C#:

```
ColorDialog1.ShowDialog();  
Pen1.Color = ColorDialog1.Color;
```

После выполнения первой строки программного кода появится диалоговое окно *Цвет* (рис. 2.15). Вторая строка программного кода присваивает перу цвет, который можно выбрать в диалоговом окне *Цвет* с помощью мыши из палитры 48 основных

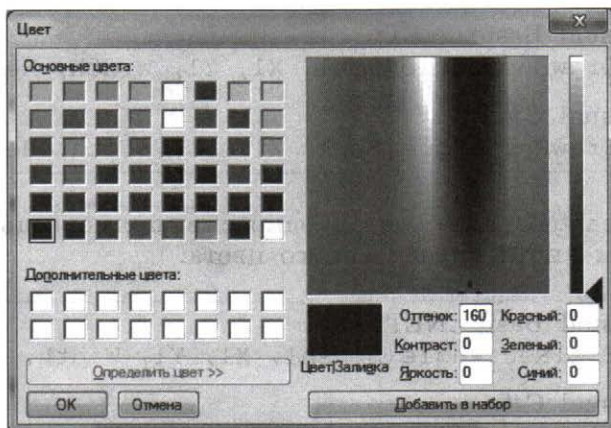


Рис. 2.15

цветов. После щелчка по кнопке *Определить цвет* в раскрывшемся диалоговом окне цвет можно выбрать из полной палитры 16 777 216 цветов, а также установить путём ввода в текстовые поля интенсивностей базовых цветов в системе цветопередачи RGB или оттенка, контрастности и яркости в системе цветопередачи HSB.

Графические методы. Графические фигуры рисуются с использованием графических методов. Замкнутые фигуры, такие как прямоугольники или эллипсы, состоят из двух частей — контура и внутренней области. Контур рисуется с использованием заданного пера, а внутренняя область закрашивается с использованием заданной кисти.

`DrawLine()` — метод рисования линии, аргументами которого являются перо определённого цвета и толщины (например, `Pen1`), а также координаты концов линии `X1`, `Y1` и `X2`, `Y2`.

Язык Visual Basic .NET:

```
Graph1.DrawLine(Pen1, X1, Y1, X2, Y2)
```

Язык Visual C#:

```
Graph1.DrawLine(Pen1, X1, Y1, X2, Y2);
```

`DrawRectangle()` — метод рисования прямоугольника, аргументами которого являются перо определённого цвета и толщины (например, `Pen1`), а также координаты левого верхнего угла `X1`, `Y1`, ширина `Width` и высота `Height`.



Язык Visual Basic .NET:

```
Graph1.DrawRectangle(Pen1, X1, Y1, Width, Height)
```

Язык Visual C#:

```
Graph1.DrawRectangle(Pen1, X1, Y1, Width, Height);
```

FillRectangle() — метод закрашки прямоугольника с использованием кисти определённого цвета.

Язык Visual Basic .NET:

```
Graph1.FillRectangle(Brush1, X1, Y1, Width, Height)
```

Язык Visual C#:

```
Graph1.FillRectangle(Brush1, X1, Y1, Width, Height);
```

DrawEllipse() — метод рисования окружности или эллипса, аргументами которого являются перо определённого цвета и толщины (например, Pen1), а также координаты левого верхнего угла описанного прямоугольника X1, Y1, ширина Width и высота Height.

Язык Visual Basic .NET:

```
Graph1.DrawEllipse(Pen1, X1, Y1, Width, Height)
```

Язык Visual C#:

```
Graph1.DrawEllipse(Pen1, X1, Y1, Width, Height);
```

FillEllipse() — метод закрашки окружности или эллипса с использованием кисти определённого цвета.

Язык Visual Basic .NET:

```
Graph1.FillEllipse(Brush1, X1, Y1, Width, Height)
```

Язык Visual C#:

```
Graph1.FillEllipse(Brush1, X1, Y1, Width, Height);
```

Для рисования точки с заданными координатами X1 и Y1 можно использовать методы **DrawRectangle(Pen1, X1, Y1, 1, 1)** или **DrawEllipse(Pen1, X1, Y1, 1, 1)**, в которых аргументы Width и Height равны 1.

Graph1.Clear() — метод, заданным цветом (например, белым) стирающий изображения в области рисования.

Язык Visual Basic .NET:

```
Graph1.Clear(Color.White)
```

Язык Visual C#:

```
Graph1.Clear(Color.White);
```

Рисование текста. Метод DrawString() позволяет выводить текст в область рисования. Аргументами метода является строка текста, шрифт, кисть и координаты начала строки. Объекты шрифт (например, drawFont) и кисть (например, drawBrush) необходимо объявить.

Язык Visual Basic .NET:

```
Dim drawFont As New Font("Arial", 12)
```

```
Dim drawBrush As New SolidBrush(Color.Black)
```

Язык Visual C#:

```
Font drawFont = new Font("Arial", 12);
```

```
Brush drawBrush = new SolidBrush(Color.Black);
```

Рисование текста в левом верхнем углу области рисования можно осуществить так:

Язык Visual Basic .NET:

```
Graph1.DrawString("Текст", drawFont, drawBrush, 0, 0)
```

Язык Visual C#:

```
Graph1.DrawString("Текст", drawFont, drawBrush, 0, 0);
```

Графика в языке программирования Lazarus

Холст. Язык программирования Lazarus для создания рисунков, рисования текста и отображения графических изображений на формах и элементах управления использует в качестве области рисования объект Canvas (Холст). Объект Canvas можно создать для формы или элемента управления (чаще всего объект Canvas создаётся для области рисования PaintBox).

```
PaintBox1.Canvas;
```

Перо. Объект Pen (Перо) определяет цвет (например, красный clRed) и ширину линии в пикселях (например, 3) рисования контуров графических объектов.

```
PaintBox1.Canvas.Pen.Color := clRed;
PaintBox1.Canvas.Pen.Width := 3;
```

Кисть. Объект `Brush` (Кисть) определяет цвет (например, закраска синего цвета `clBlue`) и стиль закрашивания прямоугольников, окружностей и других замкнутых фигур (например, сплошная закрашка `bsSolid`).

```
PaintBox1.Canvas.Brush.Color := clBlue;
PaintBox1.Canvas.Brush.Style := bsSolid;
```

Цвет. Цвет устанавливается как значение свойства `Color`, которое можно задать различными способами:

- с помощью одной из нескольких десятков констант, определяющих цвет (`clBlack` — чёрный, `clBlue` — синий, `clGreen` — зелёный, `clRed` — красный, `clYellow` — жёлтый, `clWhite` — белый и т. д.);
- с помощью функции `RgbToColor()`, реализующей цветовую модель RGB (красный, зелёный, синий). Аргументом этой функции являются десятичные значения трёх базовых цветов. Значение каждого цвета может изменяться в пределах от 0 до 255. Например, `RgbToColor(0, 0, 255)` — синий, `RgbToColor(0, 255, 0)` — зелёный, `RgbToColor(255, 0, 0)` — красный; `RgbToColor(0, 0, 0)` — чёрный и `RgbToColor(255, 255, 255)` — белый.

```
PaintBox1.Canvas.Pen.Color := RgbToColor(0, 255, 0);
```

- цвет пера или кисти можно также установить с использованием элемента управления `ColorDialog`. Для этого данный элемент управления необходимо поместить на форму и ввести в программный код следующие строки.

```
ColorDialog1.Execute;
PaintBox1.Canvas.Pen.Color := ColorDialog1.Color;
```

Графические методы. Графические фигуры рисуются с использованием графических методов.

`MoveTo(X1, Y1)` — метод осуществляет переход (без рисования линии) в точку с требуемыми текущими координатами (например, в точку с координатами `X1, Y1`).


```
PaintBox1.Canvas.MoveTo(X1,Y1);
```

`LineTo(X2,Y2)` — метод рисует прямую линию от точки с текущими координатами до точки с заданными координатами (например, в точку с координатами `X2, Y2`). Цвет и толщина линии определяются свойствами объекта `Pen`.

```
PaintBox1.Canvas.LineTo(X2,Y2);
```

`Rectangle(X1, Y1, X2, Y2)` — метод рисует прямоугольник с координатами левого верхнего и правого нижнего углов. Цвет и толщина линии контура определяются свойствами объекта `Pen`, а цвет и стиль заливки — свойствами объекта `Brush`.

```
PaintBox1.Canvas.Rectangle(X1,Y1, X2,Y2);
```

`Ellipse(X1,Y1, X2,Y2)` — метод рисует окружность, вписанную в прямоугольник с заданными координатами левого верхнего и правого нижнего углов. Цвет и толщина линии контура определяются свойствами объекта `Pen`, а цвет и стиль заливки — свойствами объекта `Brush`.

```
PaintBox1.Canvas.Ellipse(X1,Y1, X2,Y2);
```

`Pixels[X,Y]` — свойство, которое закрашивает точку холста с заданными координатами определённым цветом.

```
PaintBox1.Canvas.Pixels[X,Y] := color;
```

Рисование текста. Метод `TextOut()` позволяет выводить текст в область рисования. Аргументами метода являются координаты начала строки и строка текста. Название шрифта, его размер в пунктах, начертание (полужирное, курсив или подчёркивание) и

цвет текста определяются свойствами объекта `Font`. Рисование текста в левом верхнем углу области рисования шрифтом `Arial` размером 12 пунктов можно осуществить так.

```
PaintBox1.Canvas.Font.Name := 'Arial';
PaintBox1.Canvas.Font.Size := 12;
PaintBox1.Canvas.TextOut(0, 0, 'Текст');
```

Практическая работа 2.3

Проектирование простого графического редактора

Задание. Создать проект, который позволит рисовать линии, закрашенные прямоугольники, закрашенные окружности. Ввод координат осуществлять щелчками по графическому полю. Для рисования фигур использовать меню и панель инструментов. Обеспечить возможность открытия и сохранения графических файлов.

Электронное приложение к главе 2.



Создание проекта на языке Visual Basic .NET

1. Поместить на форму (рис. 2.18):

- графическое поле `PictureBox1`, которое будет использоваться в качестве области рисования;
- четыре надписи `Label1`, `Label2`, `Label3` и `Label4` для вывода координат;
- четыре надписи для вывода имён координат;
- меню `menuStrip1` для создания меню проекта;
- панель инструментов `ToolStrip1` для создания четырёх кнопок, обеспечивающих рисование графических примитивов и очистку поля рисования;
- диалог `colorDialog1`, который позволяет выбрать цвет с использованием диалогового окна *Цвет*;
- диалог `openFileDialog1`, который позволяет выбрать файл для открытия с использованием диалогового окна *Открыть*;
- диалог `saveFileDialog1`, который позволяет выбрать имя файла при его сохранении с использованием диалогового окна *Сохранить*.

Элементы управления `menuStrip1`, `colorDialog1`, `openFileDialog1` и `saveFileDialog1` будут видны только в процессе создания проекта (рис. 2.16).



Рис. 2.16

2. Объявить переменные, которые будут содержать координаты двух точек, а также перо и кисть, которые будут использоваться для рисования графических примитивов.

```
Dim X1, Y1, X2, Y2 As Integer
Dim Pen1 As New Pen(Color.Black, 5)
Dim Brush1 As New SolidBrush(Color.Red)
```



Будем использовать первый способ для создания области рисования с использованием растрового изображения `image1`, которое можно сохранить как графический файл. Объявить `image1` как растровое изображение.

```
Dim image1 As New Bitmap(200, 200)
Dim Graph1 As Graphics = Graphics.FromImage(image1)
```



3. Создать программный код обработчика события, который обеспечивает запоминание и вывод на надписи координат после нажатия (событие `PictureBox1_MouseDown`) и отпускания (событие `PictureBox1_MouseUp`) кнопки мыши на графическом поле. В качестве значений аргумента `e` обработчика события выступает та или иная пара координат.

```
Private Sub PictureBox1_MouseDown(sender As Object,
e As MouseEventArgs) Handles PictureBox1.MouseDown
    X1 = e.X
    Y1 = e.Y
    Label1.Text = X1
    Label2.Text = Y1
End Sub
```



```
Private Sub PictureBox1_MouseUp(sender As Object,
e As MouseEventArgs) Handles PictureBox1.MouseUp
    X2 = e.X
    Y2 = e.Y
    Label3.Text = X2
    Label4.Text = Y2
End Sub
```

Создадим меню графического редактора, для чего используем элемент управления `menuStrip1`:

<i>Файл</i>	<i>Графические примитивы</i>
<i>Открыть</i>	<i>Линия</i>
<i>Сохранить</i>	<i>Прямоугольник</i>
	<i>Окружность</i>
	<i>Очистить</i>

4. Разместить на форме элемент управления `menuStrip1`.
5. В появившемся в левом верхнем углу формы в редакторе меню создать заголовок первого уровня. В поле *Вводить здесь* внести пункт меню *Файл*.
6. Для создания пункта меню перейти на следующую строку в редакторе меню. Ввести пункты меню *Открыть* и *Сохранить*.
7. Аналогично создать пункты меню верхнего уровня *Графические примитивы* и его подпункты.
8. Двойным щелчком по пунктам меню последовательно создать заготовки обработчиков событий. Ввести программный код обработчика события рисования линии. Для выбора цвета используем диалог `colorDialog1`, а для вывода диалогового окна *Цвет* — метод `ShowDialog()`.

```
Private Sub линияToolStripMenuItem_
Click(sender As Object, e As EventArgs) Handles
ЛинияToolStripMenuItem.Click
    'цвет линии
    colorDialog1.ShowDialog()
    Pen1.Color = ColorDialog1.Color
    'вывод линии
    PictureBox1.Image = image1
    Graph1.DrawLine(Pen1, X1, Y1, X2, Y2)
End Sub
```

9. Создать программный код обработчика события рисования закрашенного прямоугольника. Для вычисления ширины и высоты прямоугольника использовать абсолютное значение разности соответствующих координат.

```
Private Sub прямоугольникToolStripMenuItem_
Click(sender As Object, e As EventArgs) Handles
ПрямоугольникToolStripMenuItem.Click
```



```
'цвет контура
colorDialog1.ShowDialog()
Pen1.Color = ColorDialog1.Color
'цвет заливки фигуры
colorDialog1.ShowDialog()
Brush1.Color = ColorDialog1.Color
PictureBox1.Image = image1
'вывод прямоугольника с заданным цветом контура
Graph1.DrawRectangle(Pen1, X1, Y1,
    Math.Abs(X2 - X1), Math.Abs(Y2 - Y1))
'вывод закрашенного прямоугольника
Graph1.FillRectangle(Brush1, X1, Y1,
    Math.Abs(X2 - X1), Math.Abs(Y2 - Y1))
```

End Sub

10. Создать программный код обработчика события рисования закрашенной окружности.

```
Private Sub ОкружностьToolStripMenuItem_
Click(sender As Object, e As EventArgs) Handles
ОкружностьToolStripMenuItem.Click
'цвет контура
ColorDialog1.ShowDialog()
Pen1.Color = ColorDialog1.Color
'цвет заливки фигуры
ColorDialog1.ShowDialog()
Brush1.Color = ColorDialog1.Color
PictureBox1.Image = image1
'вывод окружности с заданным цветом контура
Graph1.DrawEllipse(Pen1, X1, Y1,
    Math.Abs(X2 - X1), Math.Abs(Y2 - Y1))
'вывод закрашенной окружности
Graph1.FillEllipse(Brush1, X1, Y1,
    Math.Abs(X2 - X1), Math.Abs(Y2 - Y1))
```

End Sub

11. Создать программный код обработчика события очистки области рисования.

```
Private Sub ОчиститьToolStripMenuItem_
Click(sender As Object, e As EventArgs) Handles
ОчиститьToolStripMenuItem.Click
PictureBox1.Image = image1
Graph1.Clear(Color.Empty)
End Sub
```

Создадим панель инструментов графического редактора. В панель инструментов должны входить кнопки *Линия*, *Прямоугольник*, *Окружность* и *Очистить*.

Для создания панели инструментов используем элемент управления ToolStrip1.

12. Разместить на форме элемент управления ToolStrip1.
13. С помощью появившегося в левом верхнем углу панели ToolStrip1 раскрывающегося списка выбрать кнопку Button.
14. Вызвать контекстное меню для вставленной кнопки Button и выполнить команду *Загрузить рисунок...*
15. В диалоговом окне *Выбор ресурса* нажать кнопку *Импорт...* и открыть нужный рисунок (рис. 2.17).

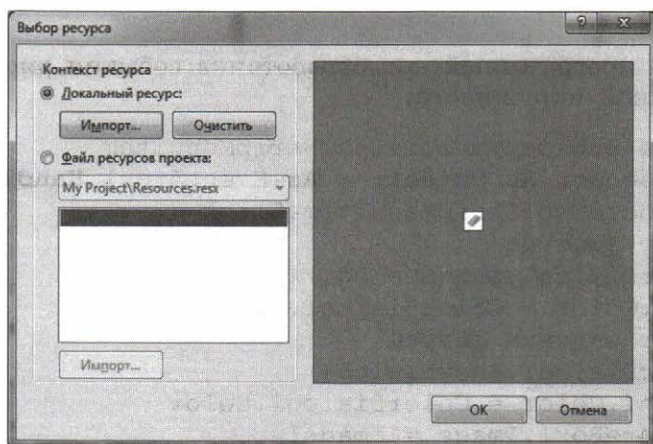


Рис. 2.17

16. Аналогично создать ещё три кнопки.
17. Создать программный код обработчика события щелчка по кнопкам панели инструментов.
18. Создать обработчик события открытия растрового графического файла.



```
Private Sub ОткрытьToolStripMenuItem_
Click(sender As Object, e As EventArgs) Handles
ОткрытьToolStripMenuItem.Click
    OpenFileDialog1.ShowDialog()
    PictureBox1.Image = Image.FromFile(OpenFileDialog1.
                                                FileName)
End Sub
```


19. Создать обработчик события сохранения растрового графического файла с расширением bmp.



```
Private Sub СохранитьToolStripMenuItem_
Click(sender As Object, e As EventArgs) Handles
СохранитьToolStripMenuItem.Click
    SaveFileDialog1.Filter = "BMP (*.bmp) | *.bmp"
    SaveFileDialog1.ShowDialog()
    image1.Save(SaveFileDialog1.FileName,
                System.Drawing.Imaging.ImageFormat.Bmp)
End Sub
```

20. Запустить проект (см. рис. 2.18).

Осуществить нажатие и отпускание кнопки мыши на графическом поле. На надписи будут выведены координаты двух точек.

С использованием меню *Графические примитивы* нарисовать графические фигуры.

С использованием меню *Файл* сохранить рисунок как растровый графический файл.

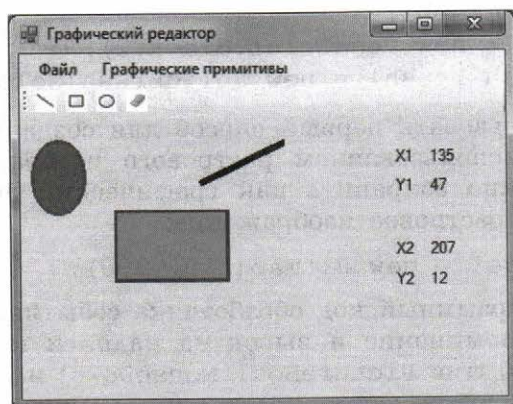


Рис. 2.18



Создание проекта на языке Visual C#

1. Поместить на форму (см. рис. 2.18):

- графическое поле PictureBox1, которое будет использоваться в качестве области рисования;
- четыре надписи Label1, Label2, Label3 и Label4 для вывода координат;

- четыре надписи для вывода имён координат;
- меню `menuStrip1` для создания меню проекта;
- панель инструментов `ToolStrip1` для создания четырёх кнопок, обеспечивающих рисование графических примитивов и очистку поля рисования;
- диалог `colorDialog1`, который позволяет выбрать цвет с использованием диалогового окна *Цвет*;
- диалог `openFileDialog1`, который позволяет выбрать файл для открытия с использованием диалогового окна *Открыть*;
- диалог `saveFileDialog1`, который позволяет выбрать имя файла при его сохранении с использованием диалогового окна *Сохранить*.

Элементы управления `menuStrip1`, `colorDialog1`, `openFileDialog1` и `saveFileDialog1` будут видны только в процессе создания проекта (см. рис. 2.15).

2. Объявить переменные, которые будут содержать координаты двух точек, а также перо и кисть, которые будут использоваться для рисования графических примитивов.

```
int X1,X2,Y1,Y2;
Graphics Graph1;
Pen Pen1 = new Pen(Color.Black,2);
SolidBrush Brush1 = new SolidBrush(Color.Blue);
```

Будем использовать первый способ для создания области рисования с использованием растрового изображения `image1`, которое можно сохранить как графический файл. Объявить `image1` как растровое изображение.

```
Bitmap image1 = new Bitmap(200, 200);
```

3. Создать программный код обработчика события, который обеспечивает запоминание и вывод на надписи координат после нажатия (событие `PictureBox1_MouseDown`) и отпускания (событие `PictureBox1_MouseUp`) кнопки мыши на графическом поле. В качестве значений аргумента `e` обработчика события выступает та или иная пара координат.

```
private void pictureBox1_MouseDown(object sender,
MouseEventArgs e)
{
    X1 = e.X;
    Y1 = e.Y;
    label1.Text = Convert.ToString(X1);
    label2.Text = Convert.ToString(Y1);
}
```



```
private void pictureBox1_MouseUp(object sender,
MouseEventArgs e)
{
    X2 = e.X;
    Y2 = e.Y;
    label3.Text = Convert.ToString(X1);
    label4.Text = Convert.ToString(Y1);
}
```

Создадим меню графического редактора, для чего используем элемент управления menuStrip1:

Файл	Графические примитивы
Открыть	Линия
Сохранить	Прямоугольник
	Окружность
	Очистить

4. Разместить на форме элемент управления menuStrip1.
5. В появившемся в левом верхнем углу формы в редакторе меню создать заголовок первого уровня. В поле *Вводить здесь* внести пункт меню *Файл*.
6. Для создания пункта меню перейти на следующую строку в редакторе меню. Ввести пункты меню *Открыть* и *Сохранить*.
7. Аналогично создать пункты меню верхнего уровня *Графические примитивы* и его подпункты.
8. Двойным щелчком по пунктам меню последовательно создать заготовки обработчиков событий. Ввести программный код обработчика события рисования линии. Для выбора цвета используем диалог colorDialog1, а для вывода диалогового окна *Цвет* — метод ShowDialog().

```
private void линияtoolStripButton1_Click(object
sender, EventArgs e)
{
    // цвет линии
    colorDialog1.ShowDialog();
    Pen1.Color = colorDialog1.Color;
    //вывод линии
    Graphics Graph1 = Graphics.FromImage(image1);
    pictureBox1.Image = image1;
    Graph1.DrawLine(Pen1, X1, Y1, X2, Y2);
}
```



9. Создать программный код обработчика события рисования закрашенного прямоугольника. Для вычисления ширины и высоты прямоугольника использовать абсолютное значение разности соответствующих координат.

```
private void прямоугольникToolStripMenuItem_
Click(object sender, EventArgs e)
{
    //цвет контура
    colorDialog1.ShowDialog();
    Pen1.Color = colorDialog1.Color;
    //цвет заливки фигуры
    colorDialog1.ShowDialog();
    Brush1.Color = colorDialog1.Color;
    Graphics Graph1 = Graphics.FromImage(image1);
    pictureBox1.Image = image1;
    //вывод прямоугольника с заданным цветом контура
    Graph1.DrawRectangle(Pen1, X1, Y1,
        Math.Abs(X2 - X1), Math.Abs(Y2 - Y1));
    //вывод закрашенного прямоугольника
    Graph1.FillRectangle(Brush1, X1, Y1,
        Math.Abs(X2 - X1), Math.Abs(Y2 - Y1));
}
```

10. Создать программный код обработчика события рисования закрашенной окружности.

```
private void окружностьToolStripMenuItem_Click(object
sender, EventArgs e)
{
    //цвет контура
    colorDialog1.ShowDialog();
    Pen1.Color = colorDialog1.Color;
    //цвет заливки фигуры
    colorDialog1.ShowDialog();
    Brush1.Color = colorDialog1.Color;
    Graphics Graph1 = Graphics.FromImage(image1);
    pictureBox1.Image = image1;
    //вывод окружности с заданным цветом контура
    Graph1.DrawEllipse(Pen1, X1, Y1,
        Math.Abs(X2 - X1), Math.Abs(Y2 - Y1));
    //вывод закрашенной окружности
    Graph1.FillEllipse(Brush1, X1, Y1,
        Math.Abs(X2 - X1), Math.Abs(Y2 - Y1));
}
```


11. Создать программный код обработчика события очистки области рисования.

```
private void очиститьToolStripMenuItem_Click(object
sender, EventArgs e)
{
    Graphics Graph1 = Graphics.FromImage(imagel);
    pictureBox1.Image = imagel;
    Graph1.Clear(Color.Empty);
}
```



Создадим панель инструментов графического редактора. В панель инструментов должны входить кнопки *Линия*, *Прямоугольник*, *Окружность* и *Очистить*.

Для создания панели инструментов используем элемент управления ToolStrip1.

12. Разместить на форме элемент управления ToolStrip1.
13. С помощью появившегося в левом верхнем углу панели ToolStrip1 раскрывающегося списка выбрать кнопку Button.
14. Вызвать контекстное меню для вставленной кнопки Button и выполнить команду *Загрузить рисунок...*
15. В диалоговом окне *Выбор ресурса* нажать кнопку *Импорт...* и открыть нужный рисунок (рис. 2.19).

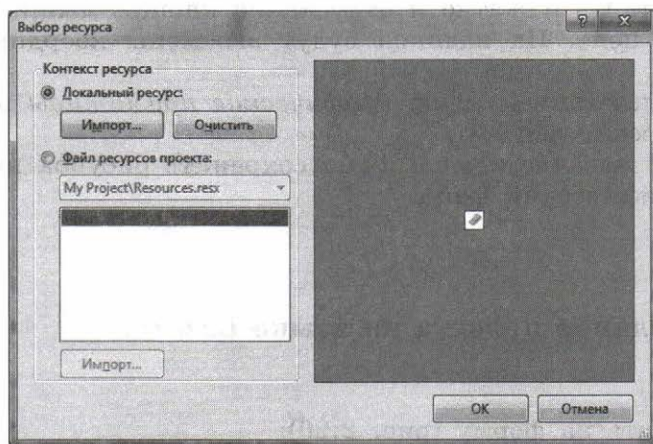


Рис. 2.19

16. Аналогично создать ещё три кнопки.
17. Создать программный код обработчика события щелчка по кнопкам панели инструментов.

18. Создать обработчик события открытия растрового графического файла.



```
private void открытьToolStripMenuItem_Click(object
sender, EventArgs e)
{
    openFileDialog1.ShowDialog();
    pictureBox1.Image =
        Image.FromFile(openFileDialog1.
        FileName);
}
```

19. Создать обработчик события сохранения растрового графического файла с расширением bmp.



```
private void сохранитьToolStripMenuItem_Click(object
sender, EventArgs e)
{
    saveFileDialog1.Filter = "BMP (*.bmp) | *.bmp";
    saveFileDialog1.ShowDialog();
    image1.Save(saveFileDialog1.FileName,
        System.Drawing.Imaging.ImageFormat.Bmp);
}
```

20. Запустить проект (см. рис. 2.18).

Осуществить нажатие и отпускание кнопки мыши на графическом поле. На надписи будут выведены координаты двух точек.

С использованием меню *Графические примитивы* нарисовать графические фигуры.

С использованием меню *Файл* сохранить рисунок как растровый графический файл.



Создание проекта на языке Lazarus



1. Поместить на форму (рис. 2.20):

- область рисования PictureBox1, которое будет использоваться в качестве области рисования;
- четыре надписи Label1, Label2, Label3 и Label4 для вывода координат;
- четыре надписи для вывода имён координат;
- меню MainMenu1 для создания меню проекта;

- панель инструментов `ToolBar1` для создания четырёх кнопок, обеспечивающих рисование графических примитивов и очистку поля рисования;
- коллекцию изображений `ImageList1` для хранения изображений, которые будут помещены на кнопки панели инструментов;
- диалог `ColorDialog1`, который позволяет выбрать цвет с использованием диалогового окна *Цвет*;
- диалог `OpenDialog1`, который позволяет выбрать файл для открытия с использованием диалогового окна *Открыть*;
- диалог `SaveDialog1`, который позволяет выбрать имя файла при его сохранении с использованием диалогового окна *Сохранить*.

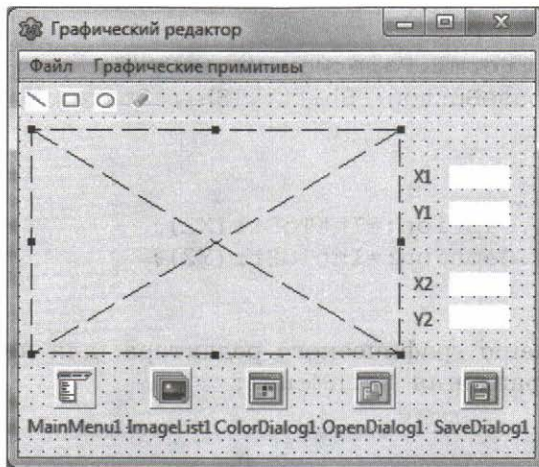


Рис. 2.20

Элементы управления `MainMenu1`, `ImageList1`, `OpenDialog1`, `SaveDialog1` и `ColorDialog1` будут видны только в процессе создания проекта.

2. Объявить переменные, которые будут содержать координаты двух точек на холсте и `Bitmap` как растровое изображение.

```
var
  X1: integer;
  Y1: integer;
  X2: integer;
  Y2: integer;
  Bitmap : TBitmap;
```



3. Создать программные коды обработчиков событий, которые обеспечивают запоминание и вывод на надписи координат после нажатия (событие `PictureBox1MouseDown`) и отпускания (событие `PictureBox1MouseUp`) кнопки мыши на графическом поле.



```
procedure TForm1.PaintBox1MouseDown(Sender: TObject;
Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    X1:=X;
    Y1:=Y;
    Label1.Caption:=IntToStr(X1);
    Label2.Caption:=IntToStr(Y1);
end;

procedure TForm1.PaintBox1MouseUp(Sender: TObject;
Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    X2:=X;
    Y2:=Y;
    Label3.Caption:=IntToStr(X2);
    Label4.Caption:=IntToStr(Y2);
end;
```

Создадим меню графического редактора, для чего используем элемент управления `MainMenu1`:

Файл	Графические примитивы
Открыть	Линия
Сохранить	Прямоугольник
	Окружность
	Очистить

4. Разместить на форме элемент управления `MainMenu1`.
 5. Осуществить двойной щелчок по элементу управления `MainMenu1`. В появившемся диалоговом окне *Редактор меню* создать меню проекта (рис. 2.21).

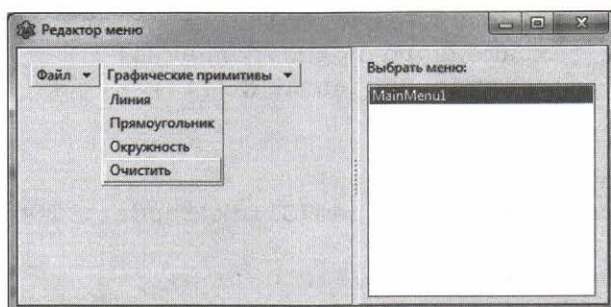


Рис. 2.21

6. Двойным щелчком по пунктам меню последовательно создать заготовки обработчиков событий рисования графических примитивов. Ввести программный код обработчика события рисования линий.

```
procedure TForm1.MenuItem5Click(Sender: TObject);
begin
    PaintBox1.Canvas.Pen.Width := 3;
    {цвет линии}
    ColorDialog1.Execute;
    PaintBox1.Canvas.Pen.Color := ColorDialog1.Color;
    {вывод линии}
    PaintBox1.Canvas.MoveTo(X1,Y1);
    PaintBox1.Canvas.LineTo(X2,Y2);
end;
```

7. Создать программные коды обработчиков событий рисования закрашенного прямоугольника, закрашенной окружности и очистки области рисования.

Для установки цвета контура графической фигуры (цвета пера) и цвета её заливки (цвета кисти) использовать диалоговое окно *Цвет*, которое вызывается с помощью элемента управления ColorDialog1.

```
{закрашенный прямоугольник}
procedure TForm1.MenuItem6Click(Sender: TObject);
begin
    {цвет контура}
    ColorDialog1.Execute;
    PaintBox1.Canvas.Pen.Color := ColorDialog1.Color;
    {цвет заливки фигуры}
    ColorDialog1.Execute;
```

```

    PaintBox1.Canvas.Brush.Color := ColorDialog1.Color;
    {вывод закрашенного прямоугольника}
    PaintBox1.Canvas.Rectangle(X1,Y1,X2,Y2);
end;

{ закрашенная окружность}
procedure TForm1.MenuItem7Click(Sender: TObject);
begin
    PaintBox1.Canvas.Pen.Width := 3;
    {цвет контура}
    ColorDialog1.Execute;
    PaintBox1.Canvas.Pen.Color := ColorDialog1.Color;
    {цвет заливки фигуры}
    ColorDialog1.Execute;
    PaintBox1.Canvas.Brush.Color := ColorDialog1.Color;
    {вывод закрашенной окружности}
    PaintBox1.Canvas.Ellipse(X1,Y1,X2,Y2);
end;

{очистка области рисования}
procedure TForm1.MenuItem8Click(Sender: TObject);
begin
    PaintBox1.Canvas.Brush.Color := clDefault;
    PaintBox1.Canvas.Clear;
end;

```

Создадим панель инструментов графического редактора. В панель инструментов должны входить кнопки *Линия*, *Прямоугольник*, *Окружность* и *Очистить*. Для создания панели инструментов используем элемент управления *ToolBar1*.

8. Разместить на форме элемент управления *ToolBar1*.

Создадим четыре кнопки на панели инструментов.

9. Выделить элемент управления *ToolBar1*, щёлкнуть правой кнопкой мыши и в контекстном меню выбрать пункт *Новая кнопка*. Прodelать эти действия четыре раза.

На каждой кнопке панели инструментов разместим соответствующее изображение, которое хранится в элементе управления *ImageList1*.

10. Разместить на форме элемент управления *ImageList1*.

11. Выделить элемент управления *ImageList1*, щёлкнуть правой кнопкой мыши и в контекстном меню выбрать пункт *Редактор ImageList...*

12. В появившемся окне *Редактор ImageList* (рис. 2.22) добавить четыре изображения для размещения на кнопках панели инструментов, нажав четыре раза кнопку *Добавить...*

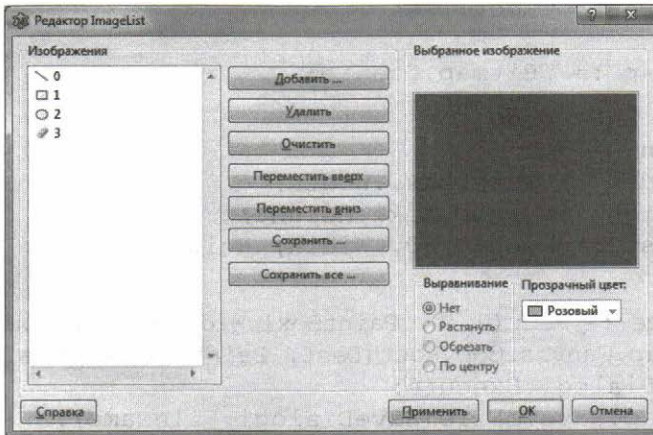


Рис. 2.22

- Установим соответствие между коллекцией кнопок на панели инструментов *ToolBar1* и коллекцией изображений *ImageList1*.
13. Выделить элемент управления *ToolBar1* и в окне *Инспектор объектов* для свойства *Images* установить значение *ImageList1*.
14. Последовательно выделить кнопки *ToolButton1*, *ToolButton2*, *ToolButton3* и *ToolButton4* на панели инструментов и в окне *Инспектор объектов* для свойства *ImageIndex* установить соответствующие порядковые номера изображения в коллекции изображений *ImageList1*.
15. Создать программный код обработчика события щелчка по кнопкам панели инструментов.
16. Создать обработчик события открытия растрового графического файла.

```
procedure TForm1.MenuItem3Click(Sender: TObject);
begin
  Bitmap := TBitmap.Create;
  OpenDialog1.Execute;
  Bitmap.LoadFromFile(OpenDialog1.FileName);
  PaintBox1.Canvas.Draw(0,0,Bitmap);
  Bitmap.Free;
end;
```





17. Создать обработчик события сохранения растрового графического файла.

```

procedure TForm1.MenuItem4Click(Sender: TObject);
    var Source: TRect;
        Dest: TRect;
begin
    Bitmap := TBitmap.Create;
    with Bitmap do
    begin
        Width := PaintBox1.Width;
        Height := PaintBox1.Height;
        Dest := Rect(0, 0, Width, Height);
    end;

    Source := Rect(0, 0, PaintBox1.Width, PaintBox1.Height);
    Bitmap.Canvas.CopyRect(Dest, PaintBox1.Canvas, Source);
    SaveDialog1.Execute;
    Bitmap.SaveToFile(SaveDialog1.FileName);
    Bitmap.Free;
end;

```

18. Запустить проект (см. рис. 2.1).

Осуществить нажатие и отпускание кнопки мыши на графическом поле. На надписи будут выведены координаты двух точек.

С использованием меню *Графические примитивы* нарисовать графические фигуры.

С использованием меню *Файл* сохранить рисунок как растровый графический файл.

2.6.5. Исследование математических моделей

На языке алгебры формальные модели записываются с помощью уравнений, точное решение которых основывается на поиске равносильных преобразований алгебраических выражений, позволяющих выразить переменную величину с помощью формулы.

Точные решения существуют только для некоторых уравнений определённого вида (линейные, квадратные, тригонометрические и др.), поэтому для большинства уравнений приходится использовать методы приближённого решения с заданной точностью (графические или численные).

Например, нельзя найти корень уравнения $x^3 - \cos x = 0$ путём равносильных алгебраических преобразований. Однако такие уравнения можно решать приближенно графическими и численными методами.

Графические методы решения уравнений. Построение графиков функций может использоваться для грубо приближённого решения уравнений. Для уравнений вида $f(x) = 0$, где $f(x)$ — некоторая непрерывная функция, корень (или корни) этого уравнения являются точкой (или точками) пересечения графика функции с осью X .

Графическое решение таких уравнений можно осуществить путём построения компьютерных моделей:

- построением графика функции в системе объектно-ориентированного программирования Visual Basic .NET, Visual C# или Lazarus;
- в электронных таблицах Microsoft Excel или OpenOffice Calc путём построения диаграммы типа *График*.

Численные методы решения уравнений. Для решения уравнений с заданной точностью можно применить разработанные в вычислительной математике численные методы решения уравнений путём последовательных приближений. Самый простой из них — метод половинного деления. Если мы определим числовой отрезок аргумента x , на котором существует корень, и функция на краях этого отрезка принимает значения разных знаков, то можно использовать метод половинного деления.

Численный метод половинного деления. Идея метода состоит в выборе точности решения и сведении первоначального числового отрезка $[A; B]$, на котором существует корень уравнения, к отрезку заданной точности. Процесс заключается в последовательном делении отрезков пополам точкой $C = (A + B)/2$ и отображению той половины отрезка ($[A; C]$ или $[C; B]$), на котором корня нет.

Выбор нужной половины отрезка основывается на проверке знаков значений функции на его краях. Выбирается та половина, на которой произведение значений функции на краях отрицательно, т. е. когда функция имеет разные знаки — пересекает ось абсцисс.

Процесс продолжается до тех пор, пока длина числового отрезка не станет меньше заданной удвоенной точности. Деление этого отрезка пополам дает значение корня с заданной точностью $x \approx (A + B)/2$.

Вопросы и задания

1. Для каких уравнений используются приближённые методы решения?
2. Почему целесообразно сначала использовать графический метод приближённого решения уравнения?
3. Когда применяется метод половинного деления?

Практическая работа 2.4

Графическое решение уравнения

Задание. Найти корень уравнения $x^3 - \cos x = 0$ приближёнными методами (графическим и численным методами деления пополам числового отрезка аргумента).

Формальная модель задана уравнением, для нахождения корня уравнения разработаем компьютерную модель на языке программирования.

Электронное приложение к главе 2.



Создание проекта на языке Visual Basic .NET

1. Поместить на форму (рис. 2.23):

- графическое поле PictureBox1, которое будет использоваться в качестве области рисования;
- текстовые поля TextBox1 и TextBox2 для ввода числовых значений концов отрезка A и B ;
- текстовое поле TextBox3 для ввода точности вычислений P ;
- надпись Label1 для вывода значений корня;
- четыре надписи для вывода обозначений;
- кнопки Button1 и Button2 для запуска обработчиков событий.

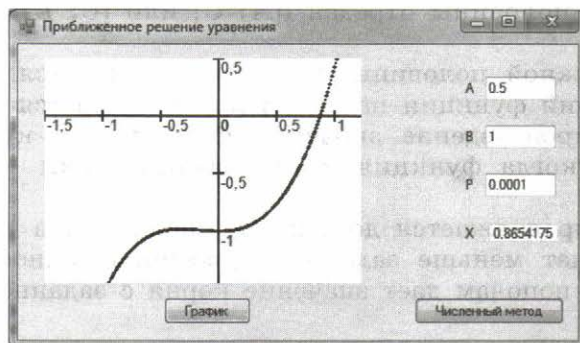


Рис. 2.23

2. Создать обработчик события для кнопки Button1 (графический метод решения уравнения).

Примечание. В Visual Basic ключевое слово `Me` указывает на текущую форму.



```

Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 2)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
Dim X, Y As Single
Private Sub Button1_Click(sender As Object,
e As EventArgs) Handles Button1.Click
    Graph1 = Me.PictureBox1.CreateGraphics()
    Graph1.Clear(Color.White)
    'Печать шкал математической системы координат
    'в компьютерной системе координат
    For X = -150 To 150 Step 50
        Graph1.DrawString(X / 100, drawFont, drawBrush,
                           X + 150, 50)
    Next X
    For Y = 0 To 200 Step 50
        Graph1.DrawString((Y - 150) / 100, drawFont,
                           drawBrush, 150, 200 - Y)
    Next Y
    'Преобразование компьютерной системы
    'координат в математическую систему координат
    Graph1.ScaleTransform(1, -1) 'Поворот оси Y
    Graph1.TranslateTransform(150, -50)
    'Сдвиг по осям X и Y
    'Рисование осей математической системы координат
    Graph1.DrawLine(Pen1, -150, 0, 150, 0) 'Ось X
    Graph1.DrawLine(Pen1, 0, -150, 0, 50) 'Ось Y
    For X = -150 To 150 Step 50 'Засечки на оси X
        Graph1.DrawLine(Pen1, X, -5, X, 5)
    Next X
    For Y = -100 To 100 Step 50 'Засечки на оси Y
        Graph1.DrawLine(Pen1, -5, Y, 5, Y)
    Next Y
    'График функции
    For X = -1.5 To 1.5 Step 0.01
        Y = X ^ 3 - Math.Cos(X)
        Graph1.DrawEllipse(Pen1, X * 100, Y * 100, 1, 1)
    Next X
End Sub

```

3. Для кнопки Button2 ввести программный код, позволяющий вычислить корень уравнения методом половинного деления с использованием цикла с постусловием, который будет выполняться, пока выполняется условие $(B - A)/2 > P$:



```

Dim A, B, C, P As Single
Private Sub Button2_Click(sender As Object,
e As EventArgs) Handles Button2.Click
    A = Val(TextBox1.Text)
    B = Val(TextBox2.Text)
    P = Val(TextBox3.Text)
    Do
        C = (A + B) / 2
        If (A ^ 3 - Math.Cos(A)) * (C ^ 3 - Math.Cos(C)) < 0
            Then B = C Else A = C
        Loop While (B - A) / 2 > P
        Label1.Text = (A + B) / 2
    End Sub

```

4. Запустить проект (см. рис. 2.23).

Щёлкнуть по кнопке *График*, в графической области появится график функции.

График функции пересекает ось X один раз, следовательно, уравнение имеет один корень. По графику грубо приближённо можно определить, что $x \approx 0,8$ (см. рис. 2.23).

Из графика функции видно, что корень находится на отрезке $[0,5; 1]$. Введём в текстовые поля значения концов числового отрезка, а также точность вычислений (например, 0,0001).

Щёлкнуть по кнопке *Численный метод*. На надпись будет выведено значения корня: $x \approx 0,8654175$ (см. рис. 2.23).

Точность вычисления корня зависит не только от параметров используемого численного метода, но и от типа переменной. В нашем случае имеет смысл говорить о математической точности результата, которая не может превышать точность числового метода, т. е. $x \approx 0,8654$.



Создание проекта на языке Visual C#

1. Поместить на форму (см. рис. 2.23):

- графическое поле PictureBox1, которое будет использоваться в качестве области рисования;
- текстовые поля TextBox1 и TextBox2 для ввода числовых значений концов отрезка A и B ;
- текстовое поле TextBox3 для ввода точности вычислений P ;
- надпись Label1 для вывода значений корня;
- четыре надписи для вывода обозначений;
- кнопки Button1 и Button2 для запуска обработчиков событий.

2. Создать обработчик события для кнопки Button1 (графический метод решения уравнения).

Примечание. В Visual C# ключевое слово `this` ссылается на текущий экземпляр класса.

```
Graphics Graph1;
Pen Pen1 = new Pen(Color.Black, 2);
Font drawFont = new Font("Arial", 10);
Brush drawBrush = new SolidBrush(Color.Black);
int X, Y;
double X1, Y1;
private void button1_Click(object sender, EventArgs e)
{
    Graph1 = this.pictureBox1.CreateGraphics();
    Graph1.Clear(Color.White);
    //Вывод шкал математической системы координат в
    //компьютерной системе координат
    for (X = -150; X < 150; X += 50)
    {
        Graph1.DrawString(System.Convert.ToString(X/100.0),
                           drawFont, drawBrush, X + 150, 50);
    }
    for (Y = 0; Y <= 200; Y += 50)
    {
        Graph1.DrawString(System.Convert.ToString(Y - 150/
        100.0),drawFont, drawBrush, 150, 200 - Y);
    }
    //Преобразование компьютерной системы координат
    //в математическую систему координат
    Graph1.ScaleTransform(1, -1); //Поворот оси Y
    Graph1.TranslateTransform(150, -50);
    //Сдвиг по //осям X и Y
    //Рисование осей в математической системе координат
    Graph1.DrawLine(Pen1, -150, 0, 150, 0); //Ось X
    Graph1.DrawLine(Pen1, 0, -150, 0, 50); //Ось Y
    //Засечки на оси X
    for (X = -150; X < 150; X += 50)
    {
        Graph1.DrawLine(Pen1, X, -5, X, 5);
    }
    //Засечки на оси Y
    for (Y = -100; Y < 100; Y += 50)
    {
```



```

    Graph1.DrawLine(Pen1, -5, Y, 5, Y);
}
for (X1 = -1.5; X1 < 1.5; X1 += 0.01)
{
    Y1 = X1 * X1 * X1 - Math.Cos(X1);
    Graph1.DrawEllipse(Pen1, Convert.ToInt16(X1 * 100),
        Convert.ToInt16(Y1 * 100), 1, 1);
}
}

```

3. Для кнопки Button2 ввести программный код, позволяющий вычислить корень уравнения методом половинного деления с использованием цикла с постусловием, который будет выполняться, пока выполняется условие $(B - A)/2 > P$:



```

double A, B, C, P;
private void button2_Click(object sender, EventArgs e)
{
    A = Convert.ToDouble(textBox1.Text);
    B = Convert.ToDouble(textBox2.Text);
    P = Convert.ToDouble(textBox3.Text);
do
{
    C = (A + B) / 2;
    if ((A * A * A - Math.Cos(A)) *
        (C * C * C - Math.Cos(C)) < 0)
    { B = C; } else { A = C; }
}
while ((B - A) / 2 > P);
label1.Text = Convert.ToString((A + B) / 2);
}

```

4. Запустить проект (см. рис. 2.23).

Щёлкнуть по кнопке *График*, в графической области появится график функции.

График функции пересекает ось X один раз, следовательно, уравнение имеет один корень. По графику грубо приближённо можно определить, что $x \approx 0,8$ (см. рис. 2.23).

Из графика функции видно, что корень находится на отрезке $[0,5; 1]$. Введём в текстовые поля значения концов числового отрезка, а также точность вычислений (например, 0,0001).

Щёлкнуть по кнопке *Численный метод*. На надпись будет выведено значения корня: $x \approx 0,86541$ (см. рис. 2.23).

Точность вычисления корня зависит не только от параметров используемого численного метода, но и от типа переменной. В нашем случае имеет смысл говорить о математической точности результата, которая не может превышать точность числового метода, т. е. $x \approx 0,8654$.



Создание проекта на языке Lazarus



1. Поместить на форму (рис. 2.24):

- графическое поле PaintBox1, которое будет использоваться в качестве области рисования;
- текстовые поля Edit1 и Edit2 для ввода числовых значений концов отрезка A и B ;
- текстовое поле Edit3 для ввода точности вычислений P ;
- надпись Label1 для вывода значений корня;
- четыре надписи для вывода обозначений;
- кнопки Button1 и Button2 для запуска обработчиков событий.

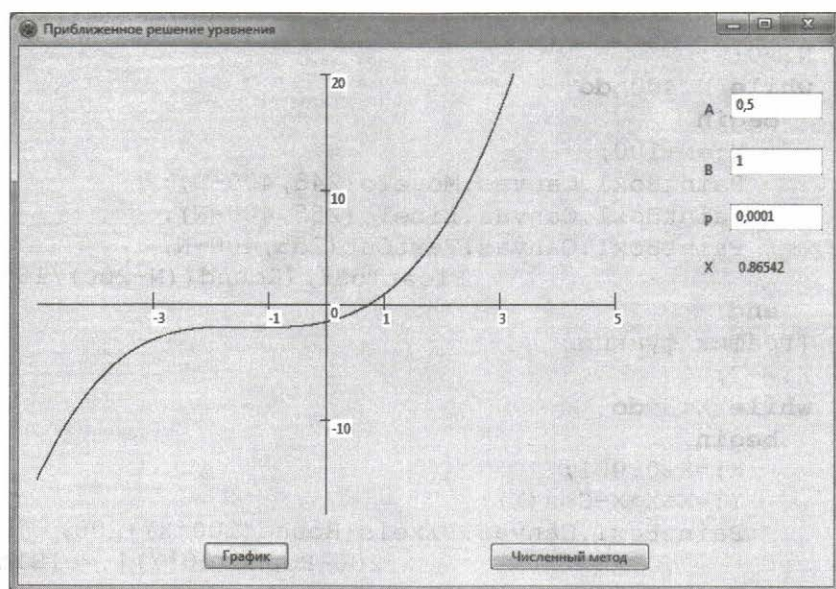


Рис. 2.24

2. Создать обработчик события для кнопки Button1 (графический метод решения уравнения).



```

var
    X, Y: real;
    N: integer;
procedure TForm1.Button1Click(Sender: TObject);
begin
    {Ось X }
    PaintBox1.Canvas.MoveTo(0,200);
    PaintBox1.Canvas.LineTo(500,200);
    {Ось Y }
    PaintBox1.Canvas.MoveTo(250,0);
    PaintBox1.Canvas.LineTo(250,500);
    {Шкала оси X}
    N:=0;
    while N<500 do
        begin
            N:=N+100;
            PaintBox1.Canvas.MoveTo(N,190);
            PaintBox1.Canvas.LineTo(N,210);
            PaintBox1.Canvas.TextOut(N,205,
                                     FloatToStr(Round(N-250)/50));
        end;
    {Шкала оси Y}
    N:=0;
    while N<400 do
        begin
            N:=N+100;
            PaintBox1.Canvas.MoveTo(245,400-N);
            PaintBox1.Canvas.LineTo(255,400-N);
            PaintBox1.Canvas.TextOut(255,400-N,
                                     FloatToStr(Round((N-200)/10)));
        end;
    {график функции }
    X:=-3;
    while X<3 do
        begin
            X:=X+0.001;
            Y:=X*X*X-Cos(X);
            PaintBox1.Canvas.Pixels[Round(100*X)+200,
                                     200-Round(20*Y)]:=clBlack;
        end;
    end;
end;

```


3. Создать обработчик события для кнопки Button2. Ввести программный код, позволяющий вычислить корень уравнения методом половинного деления с использованием цикла с постусловием, который будет выполняться, пока не станет истинным условие $(B - A)/2 < P$:

```

procedure TForm1.Button2Click(Sender: TObject);
var A, B, C, P: real;
    St:string;
begin
    A:=StrToFloat(Edit1.Text);
    B:=StrToFloat(Edit2.Text);
    P:=StrToFloat(Edit3.Text);
    Repeat
        C:=(A+B)/2;
        If (A*A-A-Cos(A))*(C*C-C-Cos(C))<0
            Then B:=C
            Else A:=C;
    Until (B-A)/2<P;
    str((A+B)/2:7:5,St);
    Label1.Caption:=St;
end;

```

4. Запустить проект (см. рис. 2.24).

Щёлкнуть по кнопке *График* — в графической области появится график функции.

График функции пересекает ось X один раз, следовательно, уравнение имеет один корень. По графику грубо приближённо можно определить, что $x \approx 0,8$ (см. рис. 2.24).

Из графика функции видно, что корень находится на отрезке $[0,5; 1]$. Введём в текстовые поля значения концов числового отрезка, а также точность вычислений (например, 0,0001).

Щёлкнуть по кнопке *Численный метод*. На надпись будет выведено значения корня: $x \approx 0,86542$ (см. рис. 2.24).

Точность вычисления корня зависит не только от параметров используемого численного метода, но и от типа переменной. В нашем случае имеет смысл говорить о математической точности результата, которая не может превышать точность числового метода, т. е. $x \approx 0,8654$.

Домашний проект

Графическое решение уравнения

Найти корень уравнения $x^3 - \cos x = 0$ приближёнными методами (графическим и численным методами деления пополам числового отрезка аргумента) с использованием электронных таблиц.

2.6.6. Оптимизационное моделирование в экономике

В сфере управления сложными системами (например, в экономике) применяется оптимизационное моделирование, в процессе которого осуществляется поиск наиболее оптимального пути развития системы.

Критериями оптимальности могут быть различные параметры, например в экономике можно стремиться к максимальному количеству выпускаемой продукции, а можно — к её низкой себестоимости. Оптимальное развитие соответствует экстремальному (максимальному или минимальному) значению выбранного целевого параметра.

Развитие сложных систем зависит от множества факторов (параметров), следовательно, целевой параметр зависит от множества параметров. Выражением такой зависимости является целевая функция

$$K = F(X_1, X_2, \dots, X_n),$$

где K — целевой параметр;

$X_1, X_2, \dots, X_n$ — параметры, влияющие на развитие системы.

Цель исследования состоит в нахождении экстремума этой функции и определении значений параметров, при которых этот экстремум достигается. Если целевая функция нелинейная, то она имеет экстремумы, которые находятся определёнными методами.

Однако часто целевая функция линейна и, соответственно, экстремумов не имеет. Задача поиска оптимального режима при линейной зависимости приобретает смысл только при наличии определённых ограничений на параметры. Если ограничения на параметры (система неравенств) также имеют линейный характер, то такие задачи являются задачами **линейного программирования**. (То есть термин «линейное программирование» в оптимизационном моделировании понимается как поиск экстремумов линейной функции, на которую наложены ограничения.)

Рассмотрим в качестве примера экономического моделирования поиск вариантов оптимального раскроя листов материала на заготовки определённого размера.

Содержательная постановка задачи. В ходе производственного процесса из листов материала получают заготовки деталей двух типов А и Б тремя различными способами, при этом количества получаемых заготовок при этих способах различны. В таблице 2.3 на пересечении строк и столбцов записаны количества заготовок типов А и Б при соответствующих способах раскроя.



Таблица 2.3

Способы раскроя заготовок

Тип заготовки	Способы раскроя		
	1-й	2-й	3-й
А	10	3	8
Б	3	6	4

Необходимо выбрать оптимальное сочетание способов раскроя, для того чтобы получить 500 заготовок типа А и 300 заготовок типа Б при расходовании наименьшего количества листов материала.

Формальная модель «Оптимизация раскроя». Параметрами, значения которых требуется определить, являются количества листов материала, которые будут раскроены различными способами:

X_1 — количество листов, раскроенное способом 1;

X_2 — количество листов, раскроенное способом 2;

X_3 — количество листов, раскроенное способом 3.

Целевая функция, выражающая количество листов материала, которое надо минимизировать, примет вид:

$$F = X_1 + X_2 + X_3.$$

Ограничения накладываются требуемыми количествами заготовок типов А и Б, тогда с учётом количества заготовок, получаемых различными способами, должны выполняться два равенства:

$$10 \cdot X_1 + 3 \cdot X_2 + 8 \cdot X_3 = 500;$$

$$3 \cdot X_1 + 6 \cdot X_2 + 4 \cdot X_3 = 300.$$

Кроме того, количества листов не могут быть отрицательными, поэтому должны выполняться неравенства:

$$X_1 \geq 0; \quad X_2 \geq 0; \quad X_3 \geq 0.$$

Таким образом, необходимо найти удовлетворяющие ограничениям значения параметров, при которых целевая функция принимает минимальное значение.

Вопросы и задания

Почему для определения минимума или максимума целевой функции, линейно зависящей от параметров, необходимо наличие условий?



Практическая работа 2.5

Построение и исследование оптимизационной модели

Задание. Создать проект «Оптимизация раскроя», который позволит выбрать оптимальное сочетание способов раскроя, для того чтобы получить 500 заготовок типа А и 300 заготовок типа Б при расходовании наименьшего количества листов материала. Способы раскроя представлены в табл. 2.3.

Электронное приложение к главе 2.



Создание проекта на языке Visual Basic .NET

Набор параметров X_1 , X_2 и X_3 (количества листов материала, которые должны быть раскроены разными способами) должен удовлетворять одновременно двум условиям:

$$10 \cdot X_1 + 3 \cdot X_2 + 8 \cdot X_3 = 500 \text{ And } 3 \cdot X_1 + 6 \cdot X_2 + 4 \cdot X_3 = 300$$

Необходимо произвести перебор всех возможных вариантов с помощью трёх вложенных циклов. С помощью оператора условного перехода надо вывести значения набора параметров и значение целевой функции на надписи.

1. Поместить на форму (рис. 2.25):

- три надписи Label1, Label2 и Label3 для вывода значений параметров;
- надпись Label4 для вывода значения целевой функции;
- четыре надписи для вывода обозначений;
- кнопку Button1 для запуска обработчика события.

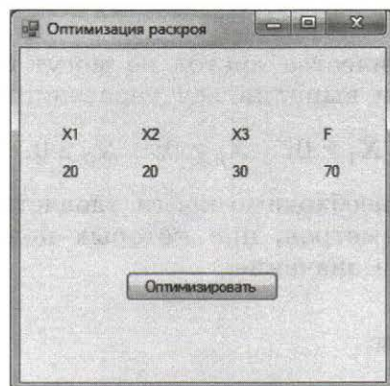


Рис. 2.25

2. Создать обработчик события:

```
Dim X1, X2, X3, F As Byte
Private Sub Button1_Click(sender As Object,
e As EventArgs) Handles Button1.Click
    F = 250
    For X1 = 0 To 100
        For X2 = 0 To 100
            For X3 = 0 To 100
                If 10 * X1 + 3 * X2 + 8 * X3 = 500
                    And 3 * X1 + 6 * X2 + 4 * X3 = 300 Then
                    If X1 + X2 + X3 < F Then
                        F = X1 + X2 + X3
                        Label1.Text = X1
                        Label2.Text = X2
                        Label3.Text = X3
                        Label4.Text = F
                    End If
                End If
            Next X3
        Next X2
    Next X1
End Sub
```

3. Запустить проект.

Щёлкнуть по кнопке *Оптимизировать*.

На надписи будет выведен набор параметров (см. рис. 2.25):

X_1 (количество листов, раскроенное способом 1) — 20;

X_2 (количество листов, раскроенное способом 2) — 20;

X_3 (количество листов, раскроенное способом 3) — 30.

Значение целевой функции F (количество листов материала) — 70.



Создание проекта на языке Visual C#

Набор параметров X_1 , X_2 и X_3 (количества листов материала, которые должны быть раскроены разными способами) должен удовлетворять одновременно двум условиям. На языке Visual C# это запишется следующим образом:

```
10 * X1 + 3 * X2 + 8 * X3 == 500
&& 3 * X1 + 6 * X2 + 4 * X3 == 300
```

Для того чтобы найти наборы значений параметров, удовлетворяющих этому условию, необходимо произвести перебор всех возможных вариантов с помощью трёх вложенных циклов. С помощью оператора условного перехода надо вывести значения набора параметров и значение целевой функции на надписи.

1. Поместить на форму (см. рис. 2.25):

- три надписи Label1, Label2 и Label3 для вывода значений параметров;
- надпись Label4 для вывода значения целевой функции;
- четыре надписи для вывода обозначений;
- кнопку Button1 для запуска обработчика события.

2. Создать обработчик события:



```

int X1, X2, X3, F;
private void button1_Click(object sender, EventArgs e)
{
    F = 250;
    for (X1 = 0; X1 < 100; X1++)
    {
        for (X2 = 0; X2 < 100; X2++)
        {
            for (X3 = 0; X3 < 100; X3++)
            {
                if (10 * X1 + 3 * X2 + 8 * X3 == 500
                    && 3 * X1 + 6 * X2 + 4 * X3 == 300)
                {
                    if (X1 + X2 + X3 < F)
                    {
                        F = X1 + X2 + X3;
                        label1.Text = Convert.ToString(X1);
                        label2.Text = Convert.ToString(X2);
                        label3.Text = Convert.ToString(X3);
                        label4.Text = Convert.ToString(F);
                    }
                }
            }
        }
    }
}

```

3. Запустить проект.

Щёлкнуть по кнопке *Оптимизировать*.

На надписи будет выведен набор параметров (см. рис. 2.25):
 X_1 (количество листов, раскроенное способом 1) — 20;
 X_2 (количество листов, раскроенное способом 2) — 20;
 X_3 (количество листов, раскроенное способом 3) — 30.
 Значение целевой функции F (количество листов материала) — 70.



Создание проекта на языке Lazarus



Набор параметров X_1 , X_2 и X_3 (количества листов материала, которые должны быть раскроены разными способами) должен удовлетворять одновременно двум условиям. На языке Lazarus это запишется следующим образом:

$(10 \cdot X_1 + 3 \cdot X_2 + 8 \cdot X_3 = 500)$ And $(3 \cdot X_1 + 6 \cdot X_2 + 4 \cdot X_3 = 300)$

Для того чтобы найти наборы значений параметров, удовлетворяющих этому условию, необходимо произвести перебор всех возможных вариантов с помощью трёх вложенных циклов. С помощью оператора условного перехода надо вывести значения набора параметров и значение целевой функции на надписи.

1. Поместить на форму (см. рис. 2.25):

- три надписи Label1, Label2 и Label3 для вывода значений параметров;
- надпись Label4 для вывода значения целевой функции;
- четыре надписи для вывода обозначений;
- кнопку Button1 для запуска обработчика события.

2. Создать обработчик события:

```
procedure TForm1.Button1Click(Sender: TObject);
var X1, X2, X3, F: integer;
begin
    F:=250;
    For X1:=0 To 100 Do
    begin
        For X2:=0 To 100 Do
        begin
            For X3:=0 To 100 Do
            begin
                If (10*X1+3*X2+8*X3=500)
                    And (3*X1+6*X2+4*X3=300)
```



```

Then
begin
  If  $X_1 + X_2 + X_3 < F$ 
  Then
  begin
     $F := X_1 + X_2 + X_3$ ;
    Label1.Caption := IntToStr(X1);
    Label2.Caption := IntToStr(X2);
    Label3.Caption := IntToStr(X3);
    Label4.Caption := IntToStr(F);
  end;
end;
end;
end;
end;
end;
end;

```

3. Запустить проект.

Щёлкнуть по кнопке *Оптимизировать*.

На надписи будет выведен набор параметров (см. рис. 2.25):

X_1 (количество листов, раскроенное способом 1) — 20;

X_2 (количество листов, раскроенное способом 2) — 20;

X_3 (количество листов, раскроенное способом 3) — 30.

Значение целевой функции F (количество листов материала) — 70.



Домашний проект

Оптимизация раскроя

В электронных таблицах создать проект «Оптимизация раскроя», который позволит выбрать оптимальное сочетание способов раскроя, для того чтобы получить 500 заготовок типа А и 300 заготовок типа Б при расходовании наименьшего количества листов материала. Способы раскроя представлены в табл. 2.3.

2.7. Исследование интерактивных компьютерных моделей

2.7.1. Исследование физических и астрономических моделей

Рассмотрим процесс построения и исследования модели на конкретном примере движения тела, брошенного под углом к горизонту.



Содержательная постановка задачи «Бросание мячика в стенку». В процессе тренировок теннисистов используются автоматы по бросанию мячика. Необходимо задать автомату необходимую скорость и угол бросания мячика для попадания в стенку определённой высоты, находящуюся на известном расстоянии.

Описательная модель. Сначала построим описательную модель процесса движения тела с использованием объектов, понятий и законов физики, т. е. в данном случае идеализированную модель движения объекта. Из условия задачи можно сформулировать следующие основные допущения:

- мячик мал по сравнению с Землей, поэтому его можно считать материальной точкой;
- изменение высоты мячика мало, поэтому ускорение свободного падения можно считать постоянной величиной $g = 9,8 \text{ м/с}^2$, следовательно, движение по оси Y можно считать равноускоренным;
- скорость бросания мячика мала, поэтому сопротивлением воздуха можно пренебречь, следовательно, движение по оси X можно считать равномерным.

Формальная модель. Для формализации модели обозначим величины:

- начальную скорость мячика — v_0 ;
- угол бросания мячика — α ;
- высоту стенки — h ;
- расстояние до стенки — s .

Изобразим график движения мячика (рис. 2.26).

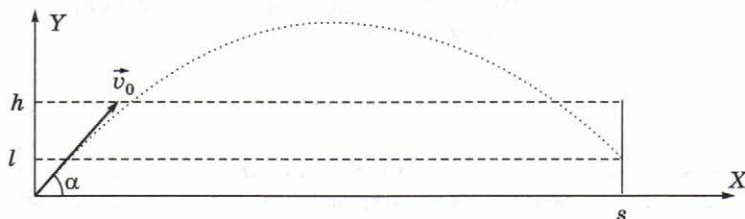


Рис. 2.26

Используем известные из курса физики формулы равномерного и равноускоренного движения для определения координат мячика. Дальность x и высоту y при заданной начальной скорости v_0 и угле бросания α для любого момента времени t можно вычислить по формулам:

$$\begin{aligned} x &= v_0 \cdot \cos \alpha \cdot t, \\ y &= v_0 \cdot \sin \alpha \cdot t - g \cdot t^2 / 2. \end{aligned} \quad (2.1)$$

Чтобы определить, попадёт ли мячик в стенку, необходимо вычислить его координату y в момент времени, когда он будет находиться на расстоянии s . Из первой формулы выражаем время, которое понадобится мячику, чтобы преодолеть расстояние s :

$$t = s / (v_0 \cdot \cos \alpha).$$

Подставляем это значение времени t в формулу для вычисления координаты y . Получаем l — высоту мячика над поверхностью земли на расстоянии s (см. рис. 2.26):

$$l = s \cdot \operatorname{tg} \alpha - g \cdot s^2 / (2 \cdot v_0^2 \cdot \cos^2 \alpha). \quad (2.2)$$

Формализуем теперь условие попадание мячика в мишень. Попадание произойдёт, если значение высоты мячика l будет удовлетворять условию в форме неравенства:

$$0 \leq l \leq h.$$

Если $l < 0$, то это означает «недолёт», а если $l > h$, то это означает «перелёт».

Рассмотрим теперь гелиоцентрическую модель Солнечной системы.

Качественная описательная модель. Гелиоцентрическая модель мира Коперника на естественном языке формулировалась следующим образом:

- Земля вращается вокруг своей оси и Солнца;
- все планеты вращаются вокруг Солнца.

Формальная модель. Ньютон формализовал гелиоцентрическую систему мира, открыв закон всемирного тяготения и законы механики и записав их в виде формул:

$$F = \gamma \cdot \frac{m_1 \cdot m_2}{r^2}; \quad \vec{F} = m \cdot \vec{a}. \quad (2.2)$$

Интерактивная компьютерная модель (рис. 2.27). Трёхмерная динамическая модель (электронный образовательный ресурс с сайта ФЦИОР: fcior.edu.ru) показывает вращение планет Солнечной системы. В центре модели изображено Солнце, вокруг него — планеты Солнечной системы.

В модели выдержаны реальные отношения орбит планет и их эксцентриситеты. Солнце находится в фокусе орбиты каждой планеты. Изобразить в небольшом окне все планеты сразу достаточно сложно, поэтому предусмотрены режимы *Меркурий...*, *Марс* и *Юпитер...Нептун*, а также режим *Все планеты*. Выбор нужного режима производится при помощи соответствующего переключателя.

Во время движения можно менять значение угла зрения в окне ввода. Получить представление о реальных эксцентриситетах орбит можно, выставив значение угла зрения 90° .

Можно изменить внешний вид модели, отключив отображение названий планет, их орбит или системы координат, показываемой в левом верхнем углу. Кнопка *Старт* запускает модель, а *Сброс* — возвращает в исходное состояние.



Рис. 2.27

Вопросы и задания

1. Какие условия входят в построение качественной модели бросания мячика в стенку?
2. Проведите компьютерный эксперимент с интерактивной компьютерной моделью «Строение Солнечной системы», размещённой на сайте ФЦИОР fcior.edu.ru (нужно ввести в строке поиска запрос «Строение Солнечной системы», далее учитель поможет вам скачать модуль ресурса).

Практическая работа 2.6

Построение и исследование модели «Бросание мячика в стенку»

Задание. Создать проект «Бросание мячика в стенку», который позволит построить и исследовать движение тела, брошенного под углом к горизонту.

Электронное приложение к главе 2.





Создание проекта на языке Visual Basic .NET

1. Поместить на форму (рис. 2.28):

- четыре текстовых поля для ввода значений: TextBox1 — начальной скорости, TextBox2 — угла бросания мячика, TextBox3 — расстояния до стенки и TextBox4 — высоты стенки;
- надпись Label1 для вывода высоты мячика на заданном расстоянии;
- надпись Label2 для вывода текстового сообщения о результатах броска;
- десять надписей для вывода имён переменных и единиц измерения;
- кнопку Button1 для запуска событийной процедуры вычисления результатов бросания мячика;
- кнопку Button2 для демонстрации траектории движения мячика;
- графическое поле PictureBox1. С помощью диалогового окна *Свойства* установить размер поля, например 400; 220.

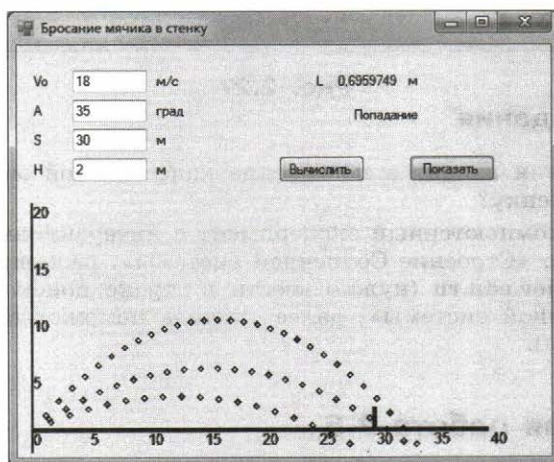


Рис. 2.28

2. Создать программный код обработчика события, который определяет попадание мячика в стенку. В этом коде:

- объявить вещественные константы одинарной точности G (ускорение свободного падения g) и Pi (число π);

- объявить вещественные переменные одинарной точности V_0 (начальная скорость v_0), A (угол бросания α), S (расстояние до стенки s), H (высота стенки h) и L (высота мячика l);
- присвоить переменным V_0 , A , S , H значения, введенные в текстовые поля, с использованием функции преобразования строки в вещественное число `Val()`;
- вычислить высоту мячика L на заданном расстоянии по формуле (2.2);
- вывести высоту мячика L на надпись `Label1`;
- вывести текстовое сообщение о результатах броска на надпись `Label2` с использованием условного оператора, в котором в качестве условия проверяется значение переменной L .

В языке программирования Visual Basic .NET аргументы тригонометрических функций `sin()`, `cos()` задаются в радианах, а угол бросания мячика мы будем вводить в градусах. Поэтому необходимо преобразовать значения углов из градусов в радианы с использованием константы `Pi`.

```
Const G As Single = 9.81
Const Pi As Single = 3.14
Dim V0, A, S, L, H As Single
Private Sub Button1_Click(sender As Object,
    e As EventArgs) Handles Button1.Click
    'Ввод начальных значений
    V0 = Val(TextBox1.Text)
    A = Val(TextBox2.Text)
    S = Val(TextBox3.Text)
    H = Val(TextBox4.Text)
    'Попадание в стенку
    L = S * Math.Tan(A * Pi / 180) - (G * S ^ 2) / (2 *
        V0 ^ 2 * Math.Cos(A * Pi / 180) ^ 2)
    Label1.Text = L
    Select Case L
        Case Is < 0
            Label2.Text = "Недолёт"
        Case Is > H
            Label2.Text = "Перелёт"
        Case Else
            Label2.Text = "Попадание"
    End Select
End Sub
```

Для визуализации компьютерной модели построим в графическом поле траекторию движения мячика.



В обработчике события осуществим преобразование компьютерной системы координат графического поля в математическую систему координат, удобную для построения траектории движения. Нарисуем оси координат и нанесем на них шкалы.

3. В математической системе координаты находятся в диапазонах $0 \leq X \leq 400$ и $-20 \leq Y \leq 200$. Траектория движения мячика, скорее всего, будет в диапазоне координат $0 \leq X \leq 40$ м и $0 \leq Y \leq 20$ м. Следовательно, необходимо увеличить масштаб графика в 10 раз:

- координаты точек графика необходимо умножить на 10;
- значения шкал осей разделить на 10.

```

Dim Graph1 As Graphics
Dim Pen1 As New Pen(Color.Black, 4)
Dim drawBrush As New SolidBrush(Color.Black)
Dim drawFont As New Font("Arial", 10)
Dim X, Y, T As Single
Private Sub Button2_Click(sender As Object,
e As EventArgs) Handles Button2.Click
    Graph1 = Me.PictureBox1.CreateGraphics()
    'Печать шкал математической системы координат
    'в компьютерной системе координат
    For X = 0 To 400 Step 50
        Graph1.DrawString(X / 10, drawFont, drawBrush, X, 200)
    Next X
    For Y = 20 To 220 Step 50
        Graph1.DrawString((Y - 20) / 10, drawFont,
            drawBrush, 0, 220 - Y)
    Next Y
    'Преобразование компьютерной системы
    'координат в математическую систему координат
    Graph1.ScaleTransform(1, -1) 'Поворот оси Y
    Graph1.TranslateTransform(0, -200)
    'Сдвиг по оси Y
    'Рисование осей математической системы координат
    Graph1.DrawLine(Pen1, 0, 0, 400, 0) 'Ось X
    Graph1.DrawLine(Pen1, 0, -20, 0, 200) 'Ось Y
    'Стенка
    Graph1.DrawLine(Pen1, S * 10, 0, S * 10, H * 10)
    'Построение траектории движения мячика
    For T = 0 To 10 Step 0.1
        Y = V0 * Math.Sin(A * Pi / 180) * T - G * T * T / 2
        X = V0 * Math.Cos(A * Pi / 180) * T
    
```



```
Graph1.DrawEllipse(Pen1, X * 10, Y * 10, 1, 1)
Next T
End Sub
```

Компьютерный эксперимент. Введём произвольные значения начальной скорости и угла бросания мячика. Скорее всего, попадания в стенку не будет. Меняя один из параметров, например угол, произведём пристрелку, используя известный артиллерийский приём «взятие в вилку», в котором применяется эффективный метод «деление пополам». Сначала найдем угол, при котором мячик перелетит стенку, затем угол, при котором мячик не долетит до стенки. Вычислим среднее значение углов, составляющих «вилку», и проверим, попадёт ли при этом значении мячик в стенку. Если он попадёт в стенку, то задача выполнена, если не попадёт, то рассмотрим новую «вилку» и т. д.

4. Запустить проект и ввести значения начальной скорости, угла бросания, расстояния до стенки и ее высоты.

Щёлкнуть по кнопкам *Вычислить* и *Показать*.

На надписи будут выведены результаты, а в графическом поле появится траектория движения тела (см. рис. 2.28).

Подобрать значения начальной скорости и угла бросания мячика, обеспечивающие его попадание в стенку.

Например, при скорости бросания мячика $v_0 = 18$ м/с и угле бросания $\alpha = 35^\circ$ мячик попадёт в стенку высотой $h = 2$ м и находящуюся на расстоянии $s = 30$ м на высоте $l = 0,6959749$ м.

Анализ результатов. Полученная точность высоты попадания мячика в стенку $l = 0,6959749$ м не имеет физического смысла и определяется типом переменной. Так как L является переменной одинарной точности, то её значение вычисляется с точностью семи значащих цифр. Исходные данные заданы с точностью две значащие цифры, поэтому необходимо и результат округлить до двух значащих цифр: $l = 0,70$ м.



Создание проекта на языке Visual C#

1. Поместить на форму (см. рис. 2.28):

- четыре текстовых поля для ввода значений: TextBox1 — начальной скорости, TextBox2 — угла бросания мячика, TextBox3 — расстояния до стенки и TextBox4 — высоты стенки;

- надпись Label1 для вывода высоты мячика на заданном расстоянии;
- надпись Label2 для вывода текстового сообщения о результатах броска;
- десять надписей для вывода имён переменных и единиц измерения;
- кнопку Button1 для запуска событийной процедуры вычисления результатов бросания мячика;
- кнопку Button2 для демонстрации траектории движения мячика;
- графическое поле PictureBox1. С помощью диалогового окна *Свойства* установить размер поля, например 400; 220.

2. Создать программный код обработчика события, который определяет попадание мячика в стенку. В этом коде:

- объявить вещественные константы одинарной точности G (ускорение свободного падения g) и Pi (число π);
- объявить вещественные переменные одинарной точности $V0$ (начальная скорость v_0), A (угол бросания α), S (расстояние до стенки s), H (высота стенки h) и L (высота мячика l);
- присвоить переменным $V0$, A , S , H значения, введенные в текстовые поля, с использованием функции преобразования строки в вещественное число `Val()`;
- вычислить высоту мячика L на заданном расстоянии по формуле (2.2);
- вывести высоту мячика L на надпись Label1;
- вывести текстовое сообщение о результатах броска на надпись Label2 с использованием условного оператора, в котором в качестве условия проверяется значение переменной L .

В языке программирования Visual C# аргументы тригонометрических функций `sin()`, `cos()` задаются в радианах, а угол бросания мячика мы будем вводить в градусах. Поэтому необходимо преобразовать значения углов из градусов в радианы с использованием константы Pi .



```
const double G = 9.81;
const double Pi = 3.14;
double V0, A, S, L, H;
private void button1_Click(object sender, EventArgs e)
{
    //Ввод начальных значений
    V0 = Convert.ToDouble(textBox1.Text);
    A = Convert.ToDouble(textBox2.Text);
    S = Convert.ToDouble(textBox3.Text);
```



```
H = Convert.ToDouble(textBox4.Text);
//Попадание в стенку
L = S * Math.Tan(A * Pi / 180) - (G * S * S) / (2 * V0
*V0 * Math.Cos(A * Pi / 180) * Math.Cos(A * Pi / 180));
label1.Text = Convert.ToString(L);
if (L<0) {label2.Text = "Недолёт";}
if (L>H) {label2.Text = "Перелёт";}
if (L==H) {label2.Text = "Попадание";}
}
```

Для визуализации компьютерной модели построим в графическом поле траекторию движения мячика.

В обработчике события осуществим преобразование компьютерной системы координат графического поля в математическую систему координат, удобную для построения траектории движения. Нарисуем оси координат и нанесём на них шкалы.

3. В математической системе координаты находятся в диапазонах $0 \leq X \leq 400$ и $-20 \leq Y \leq 200$. Траектория движения мячика, скорее всего, будет в диапазоне координат $0 \leq X \leq 40$ м и $0 \leq Y \leq 20$ м. Следовательно, необходимо увеличить масштаб графика в 10 раз:

- координаты точек графика необходимо умножить на 10;
- значения шкал осей разделить на 10.

```
Graphics Graph1;
Pen Pen1 = new Pen(Color.Black, 4);
Font drawFont = new Font("Arial", 10);
Brush drawBrush = new SolidBrush(Color.Black);
double X,Y,T;
private void button2_Click(object sender, EventArgs e)
{
    Graph1 = this.pictureBox1.CreateGraphics();
    //Печать шкал математической системы координат
    //в компьютерной системе координат
    for (X = 0; X < 400; X += 50)
    {
        Graph1.DrawString(System.Convert.ToString(X / 10),
            drawFont, drawBrush, Convert.ToInt16(X), 200);
    }
    for (Y = 20; Y < 220; Y += 50)
    {
        Graph1.DrawString(System.Convert.ToString((Y - 20) / 10),
            drawFont, drawBrush, 0, Convert.ToInt16(220 - Y));
    }
}
```



```

//Преобразование компьютерной системы
//координат в математическую систему координат
Graph1.ScaleTransform(1, -1); //Поворот оси Y
Graph1.TranslateTransform(0, -200);
//Сдвиг по оси Y
//Рисование осей математической системы
//координат
Graph1.DrawLine(Pen1, 0, 0, 400, 0); //Ось X
Graph1.DrawLine(Pen1, 0, -20, 0, 200); //Ось Y
//Стенка
Graph1.DrawLine(Pen1, Convert.ToInt16(S * 10), 0,
Convert.ToInt16(S * 10), Convert.ToInt16(H * 10));
//Построение траектории движения мячика
for (T = 0; T < 10; T += 0.1)
{
    Y = V0 * Math.Sin(A * Pi / 180) * T - G * T * T / 2;
    X = V0 * Math.Cos(A * Pi / 180) * T;
    Graph1.DrawEllipse(Pen1, Convert.ToInt16(X * 10),
Convert.ToInt16(Y * 10), 1, 1);
}
}

```

Компьютерный эксперимент. Введём произвольные значения начальной скорости и угла бросания мячика. Скорее всего, попадания в стенку не будет. Меняя один из параметров, например угол, произведём пристрелку, используя известный артиллерийский приём «взятие в вилку», в котором применяется эффективный метод «деление пополам». Сначала найдём угол, при котором мячик перелетит стенку, затем угол, при котором мячик не долетит до стенки. Вычислим среднее значение углов, составляющих «вилку», и проверим, попадёт ли при этом значении мячик в стенку. Если он попадёт в стенку, то задача выполнена, если не попадёт, то рассмотрим новую «вилку» и т. д.

4. Запустить проект и ввести значения начальной скорости, угла бросания, расстояния до стенки и её высоты.

Щёлкнуть по кнопкам *Вычислить* и *Показать*.

На надписи будут выведены результаты, а в графическом поле появится траектория движения тела (см. рис. 2.28).

Подобрать значения начальной скорости и угла бросания мячика, обеспечивающие его попадание в стенку.

Например, при скорости бросания мячика $v_0 = 18$ м/с и угле бросания $\alpha = 35^\circ$ мячик попадёт в стенку высотой $h = 2$ м и находящуюся на расстоянии $s = 30$ м на высоте $l = 0,6959749$ м.

Анализ результатов. Полученная точность высоты попадания мячика в стенку $l = 0,6959749$ м не имеет физического смысла и определяется типом переменной. Так как L является переменной одинарной точности, то ее значение вычисляется с точностью семи значащих цифр. Исходные данные заданы с точностью две значащие цифры, поэтому необходимо и результат округлить до двух значащих цифр: $l = 0,70$ м.



Создание проекта на языке Lazarus



1. Поместить на форму (рис. 2.29):

- четыре текстовых поля для ввода значений: Edit1 — начальной скорости, Edit2 — угла бросания мячика, Edit3 — расстояния до стенки и Edit4 — высоты стенки;
- надпись Label1 для вывода высоты мячика на заданном расстоянии;
- надпись Label2 для вывода текстового сообщения о результатах броска;
- десять надписей для вывода имён переменных и единиц измерения;
- кнопку Button1 для запуска событийной процедуры вычисления результатов бросания мячика;
- кнопку Button2 для демонстрации траектории движения мячика;
- графическое поле PaintBox1. С помощью диалогового окна *Свойства* установить размер поля, например 400; 220.

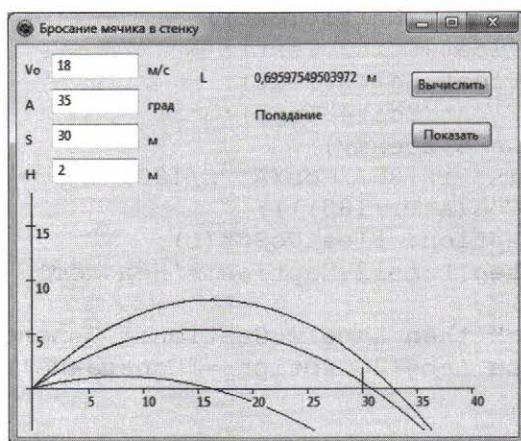


Рис. 2.29

2. Создать программный код обработчика события, который определяет попадание мячика в стенку. В этом коде:

- объявить вещественные константы одинарной точности G (ускорение свободного падения g) и Pi (число π);
- объявить вещественные переменные одинарной точности $V0$ (начальная скорость v_0), A (угол бросания α), S (расстояние до стенки s), H (высота стенки h) и L (высота мячика l);
- присвоить переменным $V0$, A , S , H значения, введенные в текстовые поля, с использованием функции преобразования строки в вещественное число `StrToFloat()`;
- вычислить высоту мячика L на заданном расстоянии по формуле (2.2);
- вывести высоту мячика L на надпись `Label1`;
- вывести текстовое сообщение о результатах броска на надпись `Label2` с использованием условного оператора, в котором в качестве условия проверяется значение переменной L .

В языке программирования Lazarus аргументы тригонометрических функций `sin()`, `cos()` задаются в радианах, а угол бросания мячика мы будем вводить в градусах. Поэтому необходимо преобразовать значения углов из градусов в радианы с использованием константы Pi .

```
var V0, A, S, H, L: real;
const
  G = 9.81;
  Pi = 3.14;
procedure TForm1.Button1Click(Sender: TObject);
begin
  V0:=StrToFloat(Edit1.Text);
  A:=StrToFloat(Edit2.Text);
  S:=StrToFloat(Edit3.Text);
  H:=StrToFloat(Edit4.Text);
  {Попадание в стенку}
  L:=S*Sin(A*Pi/180)/Cos(A*Pi/180)-G*Sqr(S)/
  (2*Sqr(V0*Cos(A*Pi/180)));
  Label1.Caption:=FloatToStr(L);
  if L<0 then Label2.Caption:='Недолёт'
  else
    if L>H then Label2.Caption:='Перелёт'
    else Label2.Caption:='Попадание';
end;
```

Для визуализации компьютерной модели построим в графическом поле траекторию движения мячика.

В обработчике события осуществим преобразование компьютерной системы координат графического поля в математическую систему координат, удобную для построения траектории движения. Нарисуем оси координат и нанесем на них шкалы.

3. В математической системе координаты находятся в диапазонах $0 \leq X \leq 400$ и $-20 \leq Y \leq 200$. Траектория движения мячика, скорее всего, будет в диапазоне координат $0 \leq X \leq 40$ м и $0 \leq Y \leq 20$ м. Следовательно, необходимо увеличить масштаб графика в 10 раз:

- координаты точек графика необходимо умножить на 10;
- значения шкал осей разделить на 10.

```

procedure TForm1.Button2Click(Sender: TObject);
var X, Y: integer;
    T: real;
    N: integer; {счётчик}
begin
    {Ввод начальных значений}
    V0:=StrToFloat(Edit1.Text);
    A:=StrToFloat(Edit2.Text);
    S:=StrToFloat(Edit3.Text);
    L:=StrToFloat(Edit4.Text);
    PaintBox1.Canvas.Brush.Color := clDefault;
    {рисование траектории}
    with PaintBox1.Canvas do
        begin
            T:=0;
            while T<10 do
                begin
                    T:=T+0.005;
                    Y:=180-Round(10*(V0*Sin(A*Pi/180)*T-G*T*T/2));
                    X:=5+Round(10*(V0*Cos(A*Pi/180)*T));
                    Pixels[X,Y]:=clBlack;
                end;
            MoveTo(0,180); LineTo(410,180); //ось X
            MoveTo(5,0); LineTo(5,400); //ось Y

            {стенка}
            MoveTo(Round(10*S),Round(180));
            LineTo(Round(10*S),180-Round(10*N));
            {шкала оси X}
            N:=0;
            while N<=400 do

```



```

begin
    N:=N+50;
    MoveTo(N,180); LineTo(N,200);
    TextOut(N,185,IntToStr(Round(N/10)));
end;
{шкала оси Y}
N:=0;
while N<=200 do
begin
    N:=N+50;
    MoveTo(0,180-N); LineTo(10,180-N);
    TextOut(10,180-N,IntToStr(Round(N/10)));
end;
end;
end;

```

Компьютерный эксперимент. Введём произвольные значения начальной скорости и угла бросания мячика. Скорее всего, попадания в стенку не будет. Меняя один из параметров, например угол, произведём пристрелку, используя известный артиллерийский приём «взятие в вилку», в котором применяется эффективный метод «деление пополам». Сначала найдём угол, при котором мячик перелетит стенку, затем угол, при котором мячик не долетит до стенки. Вычислим среднее значение углов, составляющих «вилку», и проверим, попадёт ли при этом значении мячик в стенку. Если он попадёт в стенку, то задача выполнена, если не попадёт, то рассмотрим новую «вилку» и т. д.

4. Запустить проект и ввести значения начальной скорости, угла бросания, расстояния до стенки и её высоты.

Щёлкнуть по кнопкам *Вычислить* и *Показать*.

На надписи будут выведены результаты, а в графическом поле появится траектория движения тела (см. рис. 2.29).

Подобрать значения начальной скорости и угла бросания мячика, обеспечивающие его попадание в стенку.

Например, при скорости бросания мячика $v_0 = 18$ м/с и угле бросания $\alpha = 35^\circ$ мячик попадёт в стенку высотой $h = 2$ м и находящуюся на расстоянии $s = 30$ м на высоте $l = 0,6959749$ м.

Анализ результатов. Полученная точность высоты попадания мячика в стенку $l = 0,6959749$ м не имеет физического смысла и определяется типом переменной. Так как L является переменной одинарной точности, её значение вычисляется с точностью семи значащих цифр. Исходные данные заданы с точностью две

значащие цифры, поэтому необходимо и результат округлить до двух значащих цифр: $l = 0,70$ м.

Домашний проект

Построение и исследование модели «Бросание мячика в стенку»

В электронных таблицах создать проект «Бросание мячика в стенку», который позволит построить и исследовать движение тела, брошенного под углом к горизонту.

2.7.2. Исследование химических моделей

В школьном курсе встречается достаточно много учебных ситуаций, когда ученик должен принять решение, например должен распознать (идентифицировать) тот или иной объект. Обычно такие задачи выполняются учеником методом проб и ошибок, без осознания и фиксации стратегии поиска. Создание модели принятия решения как раз и является осознанием и фиксацией последовательности рассуждений (действий), которая приводит к распознаванию того или иного объекта среди некоторой совокупности.

В качестве примера такой модели рассмотрим лабораторную работу по химии «Распознавание волокон». Учащемуся даются образцы волокон, газовая горелка и справочная таблица по свойствам горения каждого образца волокон (табл. 2.4). Предлагается распознать каждое из волокон.

Таблица 2.4

Свойства волокон

№	Горит быстро	Растворяется в ацетоне	Из расплава можно вытянуть нити	При горении ощущается запах	Волокно (результат распознавания)
1	Да	Нет	–	Да	Хлопок
2	Нет	Нет	Нет	Да	Шерсть
3	Да	Да	–	Нет	Ацетатное волокно
4	Нет	Нет	Да	Да	Капрон
5	Нет	Нет	Да	Нет	Лавсан

Стратегия распознавания может быть представлена в виде дерева поиска на основе структуры «если-то-иначе», причём может быть множество различных деревьев с различным количеством шагов. Наша цель — нахождение оптимальной стратегии распознавания (достижения цели за минимальное число шагов). Такая стратегия будет реализована, если каждый шаг будет максимально уменьшать неопределённость (нести максимальное количество информации).

Формальная модель «Распознавание волокон». На первом шаге разделим пять волокон на две группы по условию *Горит быстро*. Если условие истинно, то это волокна первой группы под номерами 1 и 3, если ложно, то это волокна второй группы под номерами 2, 4 и 5.

Для идентификации волокон первой группы достаточно проверить справедливость условия *Растворяется в ацетоне*. Если условие истинно, то это волокно 3 — ацетатное волокно, если ложно, то это волокно 1 — хлопок.

Для идентификации волокон второй группы сначала необходимо проверить справедливость условия *Из расплава можно вытянуть нити*. Если условие истинно, то это волокна 4 и 5, если ложно, то это волокно 2 — шерсть.

Для идентификации волокон 4 и 5 достаточно проверить справедливость условия *При горении ощущается запах*. Если условие истинно, то это волокно 4 — капрон, если ложно, то это волокно 5 — лавсан.

Целесообразно представить иерархическую модель распознавания (принятия решения) в виде блок-схемы (рис. 2.30).

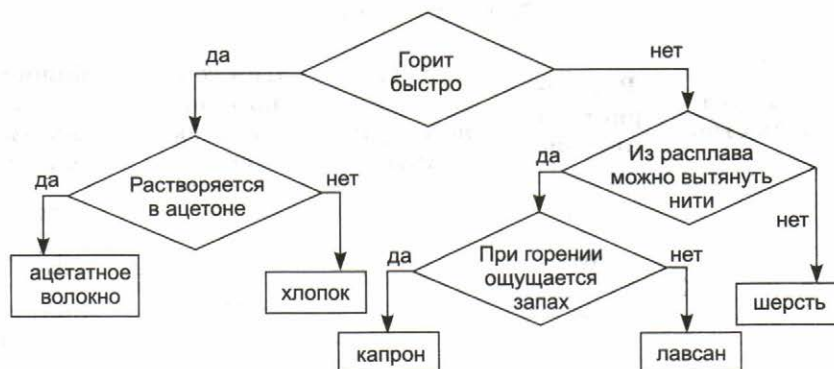


Рис. 2.30

Вопросы и задания

1. Могут ли существовать несколько моделей распознавания для одной и той же ситуации?
2. Какая стратегия построения модели распознавания быстрее всего приведёт к цели?

Практическая работа 2.7

Построение и исследование модели «Распознавание волокон»

Задание. Создать проект «Распознавание волокон». В проекте пользователю задаётся серия вопросов, анализируются ответы и сравниваются с имеющимися факторами. При этом производится логический вывод и формируется ответ на интересующий пользователя вопрос.

Результат распознавания выводится в виде списка названий волокон.

Электронное приложение к главе 2.



Создание проекта на языке Visual Basic .NET

1. Поместить на форму (рис. 2.31) кнопку Button1 и список ListBox1.

Первую развилку (условие *Горит быстро*) реализуем в форме обработчика события, а остальные — в форме общих процедур:

- для идентификации волокон первой группы (1-го и 3-го) создадим процедуру `Раствор()` (условие *Растворяется в ацетоне*);
 - для идентификации волокон второй группы сначала необходимо создать процедуру `Нити()` (условие *Из расплава можно вытянуть нити*);
 - для идентификации 4-го и 5-го волокон надо создать процедуру `Запах()` (условие *При горении ощущается запах*).
2. Создать обработчик события `Button1_Click()`, который содержит вызовы общих процедур `Раствор()` и `Нити()`:

```
Dim a As Byte
```

```
Private Sub Button1_Click(sender As Object, e As  
EventArgs) Handles Button1.Click
```

```
    a = MsgBox("Горит быстро?", 36, "Первый вопрос")
```

```
    If a = 6 Then Раствор() Else Нити()
```

```
End Sub
```

3. Создать общую процедуру `Раствор()`, которая позволяет распознать 1-е и 3-е волокна.

```
Sub Раствор()
    a = MsgBox("Растворяется в ацетоне?", 36,
        "Второй вопрос")
    If a = 6 Then ListBox1.Items.Add("3.Ацетатное волокно")
    Else ListBox1.Items.Add("1.Хлопок")
End Sub
```

4. Создать общую процедуру `Нити()`, которая позволяет распознать 2-е волокно и содержит вызов общей процедуры `Запах()`.

```
Sub Нити()
    a = MsgBox("Из расплава можно вытянуть нити?",
        36, "Второй вопрос")
    If a = 6 Then Запах()
    Else ListBox1.Items.Add("2.Шерсть")
End Sub
```

5. Создать общую процедуру `Запах()`, которая позволяет распознать 4-е и 5-е волокна.

```
Sub Запах()
    a = MsgBox("При горении ощущается запах?",
        36, "Третий вопрос")
    If a = 6 Then ListBox1.Items.Add("4.Капрон")
    Else ListBox1.Items.Add("5.Лавсан")
End Sub
```

Компьютерный эксперимент (см. рис. 2.31). Работа с моделью позволит более эффективно спланировать и провести распознавание волокон в процессе выполнения лабораторной работы по химии.

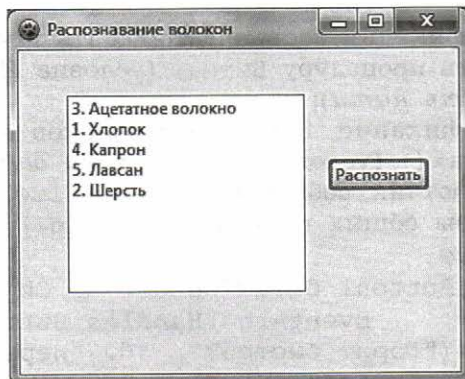


Рис. 2.31

6. Запустить проект и проводить химические опыты в соответствии с задаваемыми вопросами.
Выполнить процедуру распознавания для каждого волокна.



Создание проекта на языке Visual C#



```
int a;
private void button1_Click(object sender, EventArgs e)
{
    DialogResult a = MessageBox.Show("Горит быстро?",
        "Первый вопрос", MessageBoxButtons.YesNo,
        MessageBoxIcon.None);

    if (a == DialogResult.Yes) { Раствор(); } else {
        Нити(); };
}

void Раствор()
{
    DialogResult a = MessageBox.Show("Растворяется в
        ацетоне?", "Второй вопрос", MessageBoxButtons.
        YesNo, MessageBoxIcon.None);

    if (a == DialogResult.Yes) { listBox1.Items.Add("3.
        Ацетатное волокно"); } else { listBox1.Items.
        Add("1. Хлопок"); };
}

void Нити()
{
    DialogResult a = MessageBox.Show("Из расплава можно
        вытянуть нити?", "Третий вопрос", MessageBoxButtons.
        YesNo, MessageBoxIcon.None);
    if (a == DialogResult.Yes) { Запах(); } else {
        listBox1.Items.Add("2. Шерсть"); };
}

void Запах()
{
    DialogResult a = MessageBox.Show("При горении
        ощущается запах?", "Четвёртый вопрос",
        MessageBoxButtons.YesNo, MessageBoxIcon.None);
}
```

```

    if (a == DialogResult.Yes) { listBox1.Items.Add("4.
    Капрон"); } else { listBox1.Items.Add("5. Лавсан");
  };
}

```



Создание проекта на языке Lazarus

```

var a: integer;

procedure Rastvor;
begin
  a:=MessageDlg('Растворяется в ацетоне?', mtCustom,
                mbYesNo, 0); {'Второй вопрос'}
  If a=mrYes Then Form1.ListBox1.Items.Add('3.
                                Ацетатное волокно')
                Else Form1.ListBox1.Items.Add('1.
                                Хлопок');
end;

procedure Zapax;
begin
  a:=MessageDlg('При горении ощущается запах?',
                mtCustom, mbYesNo, 0); {'Четвёртый вопрос'}
  If a=mrYes Then Form1.ListBox1.Items.Add('4. Капрон')
                Else Form1.ListBox1.Items.Add('5. Лавсан');
end;

procedure Niti;
begin
  a:=MessageDlg('Из расплава можно вытянуть нити?',
                mtCustom, mbYesNo, 0); {'Третий вопрос'}
  If a=mrYes Then Zapax Else Form1.ListBox1.Items.
                Add('2. Шерсть');
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  a:=MessageDlg('Горит быстро?', mtCustom, mbYesNo, 0);
                {'Первый вопрос'}
  If a=mrYes Then Rastvor Else Niti;
end;

```


2.7.3. Исследование биологических моделей



В биологии при исследовании развития биосистем строятся динамические модели изменения численности популяций различных живых существ (бактерий, рыб, зверей и т. д.) с учётом различных факторов. Взаимовлияние популяций рассматривается в моделях типа «жертва–хищник».

Формальная модель «Численность популяций». Изучение динамики популяций естественно начать с простейшей модели **неограниченного роста**, в которой численность популяции ежегодно увеличивается на определённый процент. Математическую модель можно записать с помощью рекуррентной формулы, связывающей численность популяции следующего года с численностью популяции текущего года, с использованием коэффициента роста a :

$$x_{n+1} = a \cdot x_n.$$

Например, если ежегодный прирост численности популяции составляет 5%, то $a = 1,05$.

В модели **ограниченного роста** учитывается эффект перенаселённости, связанный с нехваткой питания, болезнями и т. д., который замедляет рост популяции с увеличением её численности. Введём коэффициент перенаселённости b , значение которого обычно существенно меньше. Тогда коэффициент ежегодного увеличения численности равен $(a - b \cdot x_n)$, и формула принимает вид:

$$x_n + 1 = (a - b \cdot x_n) \cdot x_n.$$

В модели **ограниченного роста с отловом** учитывается, что на численность популяций промысловых животных оказывает влияние величина ежегодного отлова. Если величина ежегодного отлова равна c , то формула принимает вид:

$$x_n + 1 = (a - b \cdot x_n) \cdot x_n - c.$$

Популяции обычно существуют не изолированно, а во взаимодействии с другими популяциями. Наиболее важным типом является взаимодействие между жертвами и хищниками (например, караси–щуки, зайцы–волки и т. д.). В модели **«жертва–хищник»** количество жертв x_n и количество хищников y_n связаны между собой. Количество встреч жертв с хищниками можно считать пропорциональным произведению собственно количеств жертв и хищников, а коэффициент f характеризует возможность гибели жертвы при встрече с хищниками. В этом случае численность популяции жертв уменьшается на величину $f \cdot x_n \cdot y_n$, и формула для расчёта численности жертв принимает вид:

$$x_{n+1} = (a - b \cdot x_n) \cdot x_n - c - f \cdot x_n \cdot y_n.$$

Численность популяции хищников в отсутствие жертв (в связи с отсутствием пищи) уменьшается, что можно описать рекуррентной формулой

$$y_{n+1} = d \cdot y_n,$$

где значение коэффициента $d < 1$ характеризует скорость уменьшения численности популяции хищников.

Увеличение популяции хищников можно считать пропорциональной произведению собственно количеств жертв и хищников, а коэффициент g характеризует величину роста численности хищников за счёт жертв. Тогда для численности хищников можно использовать формулу:

$$y_{n+1} = d \cdot y_n + g \cdot x_n \cdot y_n.$$



Вопросы и задания

Приведите примеры изменения численности популяций, соответствующих моделям неограниченного роста, ограниченного роста, ограниченного роста с отловом и «жертва–хищник» в природе.



Практическая работа 2.8

Построение и исследование модели «Популяция»

Задание. В электронных таблицах создать компьютерную модель, позволяющую исследовать численность популяций с использованием различных моделей: неограниченного роста, ограниченного роста, ограниченного роста с отловом и «жертва–хищник».



Компьютерная модель «Популяция» в электронных таблицах



Электронное приложение к главе 2.

1. В ячейки B2 и B7 ввести начальные значения численности популяций жертв и хищников. Для простоты примем начальные количества жертв X_1 и хищников Y_1 за единицу (рис. 2.32).
В ячейки B3:B6 ввести значения коэффициентов a , b , c и f , влияющих на изменение численности жертв.

В ячейки B8 и B9 внести значения коэффициентов d и g , влияющих на изменение численности хищников.

В столбце D будем вычислять численность популяции в соответствии с моделью неограниченного роста, в столбце E — ограниченного роста, в столбце F — ограниченного роста с отловом, в столбцах G и H численность популяций жертв и хищников.

	A	B
1		
2	X1=	1,00
3	a=	1,60
4	b=	0,05
5	c=	0,30
6	f=	0,06
7	Y1=	1,00
8	d=	0,90
9	g=	0,05

Рис. 2.32

- В ячейки D2, E2, F2, G2 и H2 внести значения начальной численности популяций.

В ячейку D3 внести рекуррентную формулу неограниченного роста $=B\$3*D2$.

В ячейку E3 внести рекуррентную формулу ограниченного роста $=(B\$3-B\$4*E2)*E2$.

В ячейку F3 внести рекуррентную формулу ограниченного роста с отловом $=(B\$3-B\$4*F2)*F2-B\$5$.

В ячейку G3 внести рекуррентную формулу роста жертв $=(B\$3-B\$4*G2)*G2-B\$5-B\$6*G2*H2$.

В ячейку H3 внести рекуррентную формулу роста хищников $=B\$8*H2+B\$9*G2*H2$.

- Скопировать внесённые формулы в ячейки столбцов командами в OpenOffice Calc [*Правка—Заполнить—Вниз*], в Microsoft Excel [*Главная—Заполнить—Вниз*].

В ячейках столбцов ознакомиться с динамикой изменения численности популяций (рис. 2.33).

Для визуализации компьютерной модели построим графики изменения популяций с течением времени.

- Выделить столбцы данных D, E, F, G, H и построить диаграмму. В OpenOffice Calc выбрать [*Диаграмма XY—Линии с точками*], в Microsoft Excel — [*График—Нормированный график с маркерами и накоплением*].

Появятся графики изменения численности популяций в соответствии с моделями неограниченного роста, ограниченного роста, ограниченного роста с отловом, количества жертв и хищников в модели «жертва—хищник» (рис. 2.34).

- Для того чтобы график был более удобен для исследования, удалить значения из ячеек B10:B22. В результате пролучим следующий график (рис. 2.35).

D	E	F	G	H
Неогр. рост	Огран. рост	С отловом	Жертвы	Хищники
1	1	1	1	1
1,60	1,55	1,25	1,19	0,95
2,56	2,36	1,62	1,47	0,91
4,10	3,50	2,16	1,86	0,89
6,55	4,98	2,93	2,40	0,88
10,49	6,73	3,96	3,13	0,90
16,78	8,51	5,25	4,04	0,95
26,84	9,99	6,72	5,12	1,05
42,95	11,00	8,19	6,26	1,21
68,72	11,55	9,45	7,30	1,47
109,95	11,81	10,36	8,08	1,86
175,92	11,92	10,91	8,46	2,42
281,47	11,97	11,20	8,43	3,20
450,36	11,99	11,35	8,02	4,23
720,58	11,99	11,42	7,28	5,50
1152,92	12,00	11,45	6,29	6,96
1844,67	12,00	11,47	5,16	8,45
2951,48	12,00	11,47	4,01	9,78
4722,37	12,00	11,47	2,96	10,77
7555,79	12,00	11,48	2,08	11,28
12089,26	12,00	11,48	1,41	11,33

Рис. 2.33

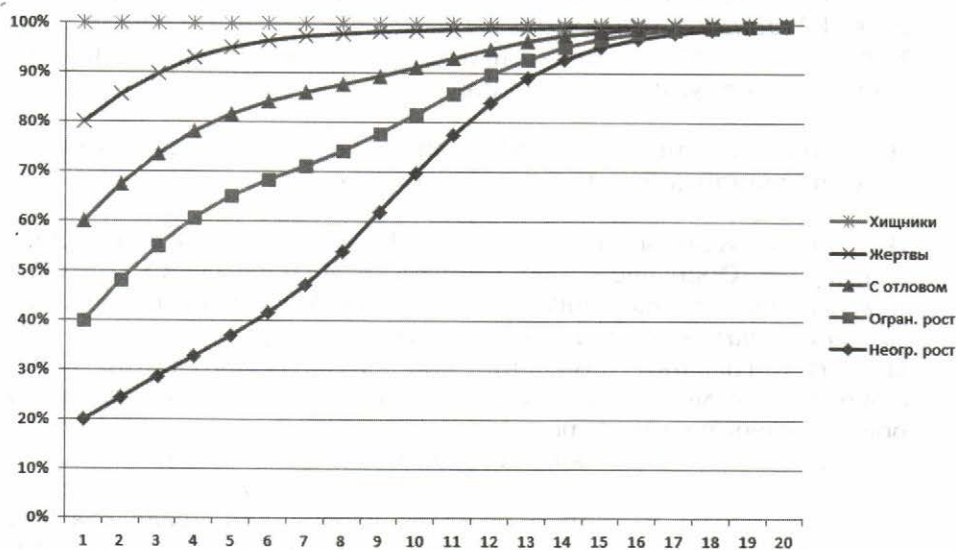


Рис. 2.34

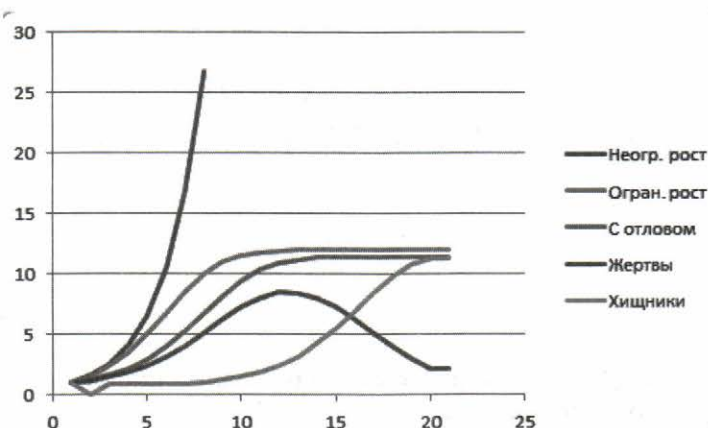


Рис. 2.35

Исследование модели. Изменяя значения начальной численности популяций, а также коэффициенты, можно получать различные варианты изменения численности популяций в зависимости от времени. При заданных значениях коэффициентов по графикам видно, что, начиная примерно с пятнадцатого жизненного цикла (года):

- отлов на численность популяции не влияет;
- количество хищников резко возрастает, а количество жертв уменьшается практически до нуля.

ЗОР к главе 2 на сайте ФЦИОР (<http://fcior.edu.ru>)

www

- Движение тела под углом к горизонту
- Движение тела, брошенного под углом к горизонту
- Двумерные и многомерные массивы
- Идентификаторы, ключевые слова, литералы
- Назначение и виды информационных моделей
- Общий обзор .NET и синтаксиса C#
- Одномерные массивы данных
- Поиск и преобразование информации в массиве
- Популяционно-видовой уровень организации живой материи на примере отношений хищник-жертва
- Построение информационных моделей ИС
- Применение методов численного решения уравнений
- Сортировка информации в массиве. Классификация методов
- Строение Солнечной системы
- Формализация задач из различных предметных областей. Формирование требований к ИС

Глава 3

БАЗЫ ДАННЫХ. СИСТЕМЫ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ (СУБД)

www

При изучении данной главы рекомендуется установить следующее программное обеспечение:

- для операционной системы Windows:
 - программу составления генеалогических деревьев GenoPro;
- для операционных систем Windows и Linux:
 - систему управления базами данных (СУБД) OpenOffice Base.



3.1. Базы данных

Для упорядоченного хранения и обработки связанных между собой данных используются базы данных.



База данных (БД) представляет собой определённым образом организованную совокупность данных некоторой предметной области, хранящуюся в компьютере и постоянно используемую.

Библиотечный каталог — пример базы данных. Он хранит информацию о книгах, каждая из которых имеет название, автора, год издания и т. д.

Информация в базах данных хранится в упорядоченном виде. Так, в библиотечном каталоге все записи упорядочены либо по алфавиту (алфавитный каталог), либо по области знания (предметный каталог).

База данных является информационной моделью организации данных предметной области.

Базы данных можно классифицировать по различным признакам. Рассмотрим их классификацию по используемой модели

данных. Принято выделять реляционные, иерархические и сетевые модели данных. В настоящее время реляционные БД являются доминирующими, поэтому о них в основном и пойдёт речь в учебнике.

Реляционные базы данных. Реляционная модель базы данных была предложена в 1969 г. математиком, научным сотрудником фирмы IBM Э. Ф. Коддом.

Слово «реляционная» происходит от английского *relation* — отношение. Это строгое математическое понятие, относящееся к теории множеств. Для пользователя базы данных отношения удобно представлять в виде неупорядоченных таблиц. Таблицы состоят из столбцов и строк и содержат данные.

В виде двумерной таблицы удобно представлять базу данных, хранящую данные о группе объектов с одинаковыми свойствами. В каждой строке таблицы последовательно размещаются значения свойств одного из объектов; каждое значение свойства — в своём столбце, озаглавленном именем свойства.

Столбцы такой таблицы называют полями; каждое поле характеризуется своим именем (именем соответствующего свойства) и типом данных, представляющих значения данного свойства.

Поле базы данных — это столбец таблицы, содержащий значения определённого свойства.



Строки таблицы являются записями об объекте; эти записи разбиты на поля столбцами таблицы, поэтому каждая запись представляет набор значений, содержащихся в полях.

Запись базы данных — это строка таблицы, содержащая набор значений свойств, размещённый в полях базы данных.



Каждая таблица должна содержать, по крайней мере, одно ключевое поле, содержимое которого уникально для каждой записи в этой таблице. Ключевое поле позволяет однозначно идентифицировать запись в таблице.

Ключевое поле — это поле, значения которого однозначно определяют запись в таблице.



В качестве ключевого поля чаще всего используют поле, содержащее порядковый номер. Однако иногда удобнее в качестве ключевого поля таблицы использовать другие поля, код объекта, например инвентарный номер, и т. п.

Тип поля. Тип поля определяется типом данных, которые оно содержит. В OpenOffice Base поля могут содержать данные следующих основных типов:

- *Текст* — содержит текст длиной до 255 символов;
- *Число* — содержит числа;
- *Целое* — содержит целые числа;
- *Вещественное* — содержит вещественные числа;
- *Дата/Время* — содержит информацию о дате и времени;
- *Логическое* — содержит значения TRUE (*истина*) или FALSE (*ложь*).

Каждый тип поля имеет свой набор свойств. Наиболее важными свойствами полей являются:

- *Длина* — определяет максимальную длину (текстовые и числовые поля);
- *Пример формата* — устанавливает формат представления данных;
- *Обязательное* — указывает, что данное поле должно быть заполнено.

База данных «Процессоры». Рассмотрим, например, базу данных «Процессоры» (табл. 3.1), которая содержит перечень объектов (процессоров), каждый из которых имеет имя (название). В качестве характеристик (свойств) можно рассмотреть количество элементов в процессоре и частоту системной шины. Поле *Процессор* является текстовым, а поля *Технологический процесс*, *Частота системной шины* и, естественно, *Счётчик* — числовыми полями.

При этом каждое поле обладает определённым набором свойств. Например, для поля *Счётчик* задан формат данных *целое число*.

Таблица 3.1

Табличная база данных «Процессоры»

Счётчик	Процессор	Технологический процесс, нм	Частота системной шины, МГц
1	Intel Pentium III	250	100
2	Intel Core 2	65	533
3	Celeron M	90	533
4	Intel Core i7	32	2400

Вопросы и задания



1. В чём заключается разница между записью и полем в базе данных?
2. Поля каких типов могут присутствовать в базе данных?
3. Чем отличается ключевое поле от остальных полей?

3.2. Система управления базами данных

3.2.1. Основные объекты СУБД: таблицы, формы, запросы, отчёты

Системы управления базами данных (СУБД). Развитие информационных технологий привело к созданию компьютерных баз данных. Создание баз данных, а также операции поиска и сортировки данных выполняются специальными программами — системами управления базами данных (СУБД). Таким образом, необходимо различать собственно базы данных (БД), которые являются упорядоченными наборами данных, и системы управления базами данных (СУБД) — программы, управляющие хранением и обработкой данных.

Система управления базами данных (СУБД) — это программа, позволяющая создавать базы данных, а также обеспечивающая обработку (сортировку) и поиск данных.



В реляционных СУБД Microsoft Access и OpenOffice Base используется стандартный для операционных систем многооконный интерфейс, но, в отличие от других приложений, не многодокументный. Единовременно может быть открыта только одна база данных, содержащая обязательное *окно базы данных* и *окна для работы с объектами базы данных*. В каждый момент времени одно из окон является активным, и в нём курсором отмечается активный объект.

Окно базы данных — один из главных элементов интерфейса СУБД. Здесь систематизированы все объекты базы данных: *таблицы, запросы, формы, отчеты*.

Таблицы. В реляционных базах данных вся информация хранится в двумерных таблицах. Это *базовый* объект базы данных, все остальные объекты создаются на основе существующих таблиц (производные объекты). Каждая строка в таблице — *запись* базы данных, а столбец — *поле*. Запись содержит набор данных об одном объекте, а поле — однородные данные обо всех объектах.

Запросы. В СУБД запросы являются важнейшим инструментом. Главное предназначение запросов — это отбор данных на основании заданных условий.

Формы. Формы позволяют отображать данные, содержащиеся только в одной записи. При помощи форм можно добавлять в таблицы новые данные, а также редактировать или удалять существующие. Форма может содержать рисунки, графики и другие внедрённые объекты.

Отчёты. Они предназначены для печати данных, содержащихся в таблицах и запросах, в красиво оформленном виде.

Существует достаточно много различных СУБД, но для первого знакомства мы рекомендуем СУБД OpenOffice Base.



Вопросы и задания

1. Какую функцию выполняют СУБД?
2. Можно ли открыть в СУБД несколько разных баз данных?
3. Какие основные объекты входят в СУБД и какие функции они выполняют?



Практическая работа 3.1

Создание базы данных

Задание. В СУБД OpenOffice Base создать базу данных «Процессоры» (см. табл. 3.1).

Варианты выполнения работы:

- включить в базу данных различные поля;
- заполнить базу данных различными записями.



Создание базы данных «Процессоры»



Электронное приложение к главе 3.

1. В операционной системе Windows или в операционной системе Linux запустить СУБД OpenOffice Base.
2. В появившемся диалоговом окне *Мастер баз данных* (шаг 1) выбрать с помощью переключателя пункт *Создать новую базу данных* (рис. 3.1).
Щёлкнуть по кнопке *Далее >>*.

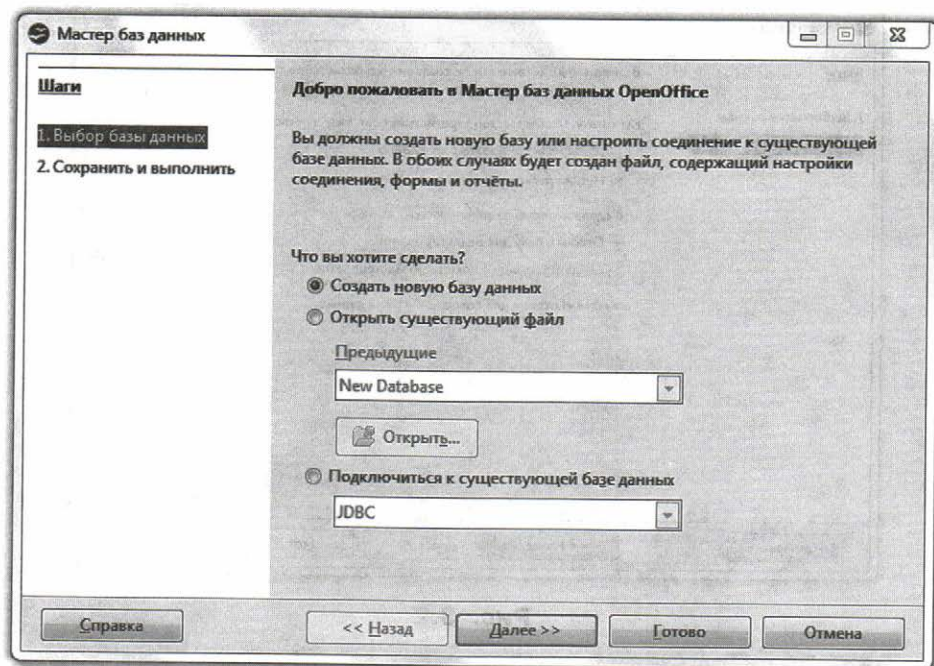


Рис. 3.1

3. В появившемся диалоговом окне *Мастер баз данных* (шаг 2) установить флажок *Открыть базу для редактирования* (рис. 3.2).
Щёлкнуть по кнопке *Готово*.
4. Откроется диалоговое окно *Сохранить как*, в котором выбрать место сохранения и имя базы данных («Процессоры»).
5. В левой части появившегося диалогового окна *Процессоры* в разделе *База данных* выбрать пункт *Таблицы*, в средней части окна в разделе *Задачи* выбрать пункт *Создать таблицу в режиме дизайна...* (рис. 3.3).
6. В появившемся окне *Таблица 1* в столбце *Название поля* ввести имена полей базы данных «Процессоры».
В столбце *Тип поля* с помощью раскрывающегося списка выбрать для каждого поля его тип (рис. 3.4).
7. Сохранить структуру базы данных [*Файл—Сохранить*]: в диалоговом окне *Сохранить как* ввести *Имя таблицы* — «Таблица 1»; нажать кнопку *ОК*.
8. Закрыть окно *Таблица1—OpenOffice.org Base: Table Design*.

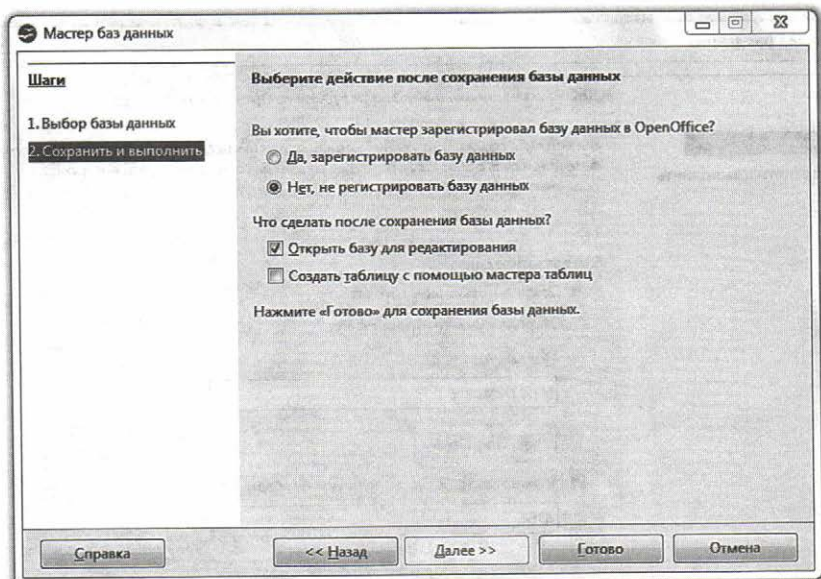


Рис. 3.2

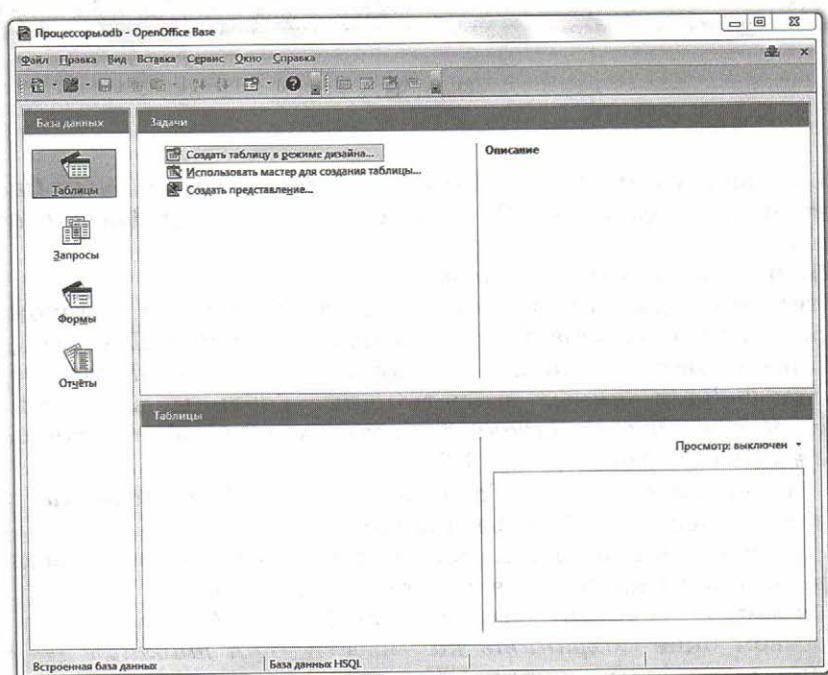


Рис. 3.3

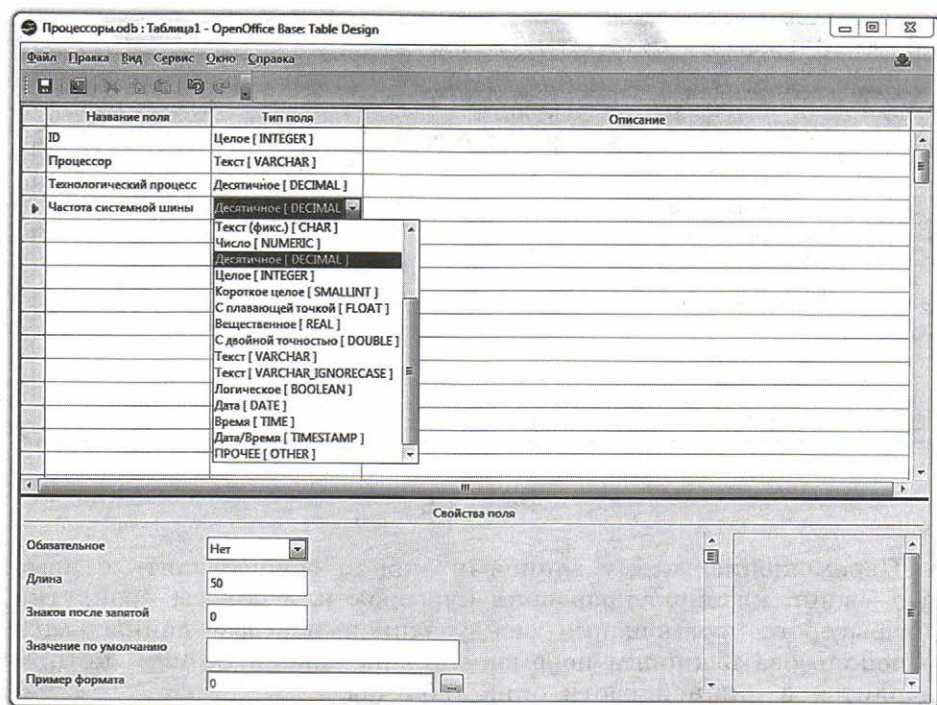


Рис. 3.4

9. В диалоговом окне *Процессоры* в разделе *Таблицы* появится пункт *Таблица 1*.

Ввод данных в таблицу базы данных и их редактирование мало чем отличаются от аналогичных действий в других офисных приложениях.

При вводе данных в режиме *Таблица* в поле маркера записи, которое расположено слева от полей таблицы, может отображаться один из следующих символов:



(звёздочка) — обозначает пустую запись в конце таблицы;



(стрелка) — обозначает выделенную (активную) запись;



(карандаш) — обозначает, что в записи производятся изменения.

10. Осуществить двойной щелчок по пункту *Таблица 1* и в появившемся диалоговом окне *Таблица 1—Процессоры—OpenOffice Base: Table Data View* ввести записи базы данных «Процессоры» (рис. 3.5).

Таблица1 - Процессоры - OpenOffice Base: Table Data View

Файл Правка Вид Вставка Сервис Окно Справка

	ID	Процессор	Технологический процесс нм	Частота системной шины МГц
▶	1	Intel Pentium III	250	100
	2	Intel Core 2	65	533
	3	Celeron M	90	533
	4	Intel Core i7	32	2400

Запись 1 из 4

Рис. 3.5

Перемещение между записями можно осуществлять с помощью мыши, клавиш управления курсором или полосы прокрутки. Для быстрого перемещения между записями в базе данных можно использовать кнопки перемещения на панели *Запись*, которая находится в нижней части окна таблицы.

3.2.2. Использование формы для просмотра и редактирования записей в базе данных

Записи базы данных можно просматривать и редактировать в виде **таблицы** или в виде **формы**. Выше мы работали с базой данных, представленной в виде таблицы, когда запись образует строку в этой таблице. Такое представление базы данных позволяет наблюдать несколько записей одновременно, и в этом состоит достоинство табличного представления.

Однако часто вид *Таблица* не позволяет видеть полностью всю информацию на экране. Если база данных содержит достаточно много полей, а значения полей содержат много символов, то не все поля таблицы могут уместиться на экране, а значения полей могут быть видны не полностью.

Форма отображает одну запись в удобном для пользователя виде. В процессе создания формы можно указать, какие поля базы данных включить в форму, как расположить поля в окне формы, а также как можно сделать форму визуально привлекательной.

Фактически с помощью формы создаётся графический интерфейс доступа к базе данных, который может содержать различ-

ные элементы управления (*текстовые поля, кнопки, переключатели* и т. д.), а также *надписи*. Обычно на форме размещаются *надписи*, являющиеся именами полей базы данных, и *текстовые поля*, содержащие данные из базы данных.

Пользователь может изменять **дизайн** формы (размер, цвет и т. д.), элементов управления и надписей.

Примерами форм могут являться *Визитка* в базе данных «Записная книжка» или *Карточка* в базе данных «Библиотечный каталог», которые содержат лишь одну запись базы данных, зато представленную в удобном для пользователя виде.

Вопросы и задания

В чём заключается разница между представлением базы данных с помощью таблицы и формы?

Практическая работа 3.2

Создание формы в базе данных

Задание. В СУБД OpenOffice Base создать форму для базы данных «Процессоры».

Варианты выполнения работы:

- задать различный дизайн формы и расположение полей и надписей на форме.



Создание формы для базы данных «Процессоры»



Электронное приложение к главе 3.

Создание формы можно проводить различными способами:

- самостоятельно (в *Режиме дизайна*) — этот способ позволяет начать создание формы «с нуля»;
- с использованием *Мастера форм* (более простой путь), который с помощью серии диалоговых окон помогает пользователю в создании формы.

Создадим форму для базы данных «Процессоры» с помощью *Мастера форм*.

1. В левой части появившегося диалогового окна *Процессоры* в разделе *База данных* выбрать пункт *Формы*, в средней части окна в разделе *Задачи* выбрать пункт *Использовать мастер для создания формы* ...

2. В появившемся диалоговом окне *Мастер форм* (шаг 1: *Выбор поля*) выбрать поля базы данных из *Таблицы 1*, которые будут отображаться на форме (рис. 3.6).

Щёлкнуть по кнопке *Дальше >*.

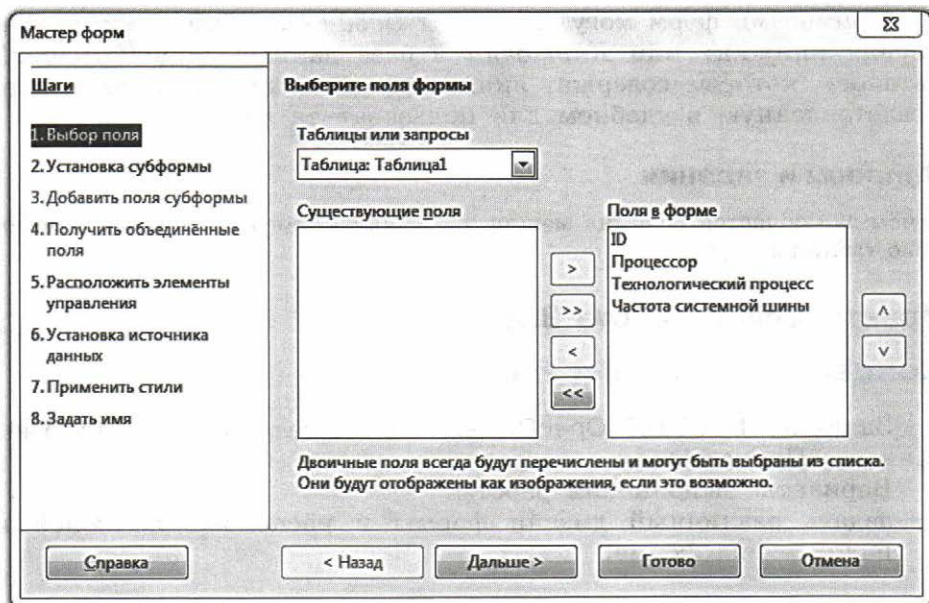


Рис. 3.6

3. В диалоговом окне *Мастер форм* (шаг 5: *Расположить элементы управления*) выбрать расположение полей, которые будут отображаться на форме (рис. 3.7).
Щёлкнуть по кнопке *Дальше >*.
4. В диалоговом окне *Мастер форм* (шаг 6: *Установка источника данных*) установить переключатель в положение *Форма для отображения всех данных*.
Щёлкнуть по кнопке *Дальше >*.
5. В диалоговом окне *Мастер форм* (шаг 7: *Применить стили*) выбрать стиль и обрамление поля (рис. 3.8).
Щёлкнуть по кнопке *Дальше >*.
6. В диалоговом окне *Мастер форм* (шаг 8: *Задать имя*) ввести название формы (например, *Форма 1*) и установить переключатель в положение *Модифицировать форму*.
Щёлкнуть по кнопке *Готово*.

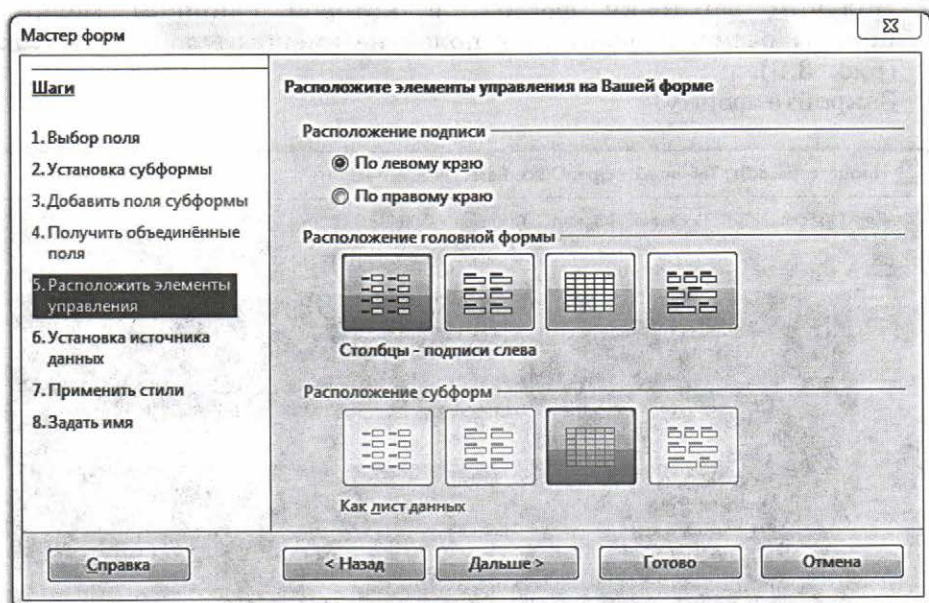


Рис. 3.7

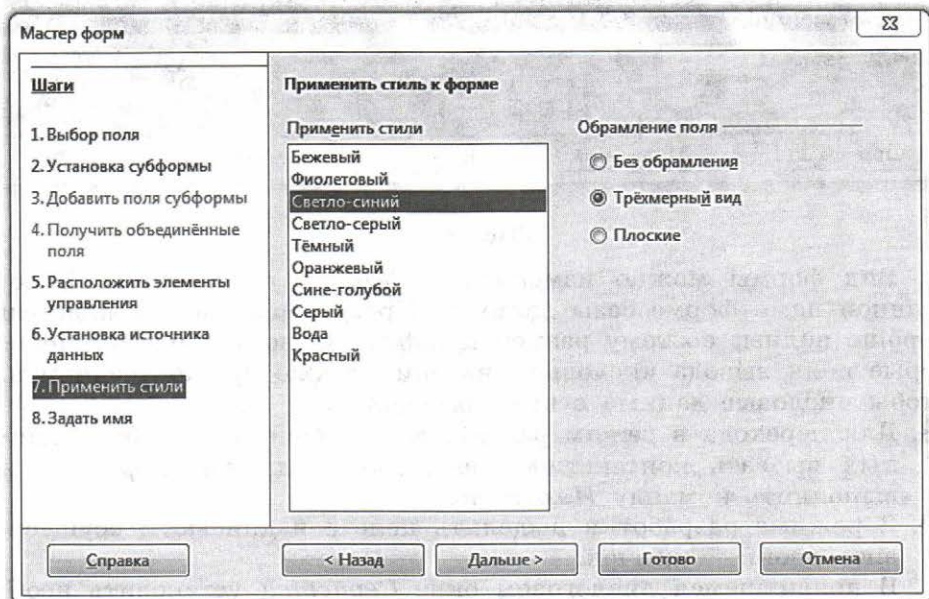


Рис. 3.8

7. Получим заготовку формы, в которой слишком мелкий шрифт, размеры некоторых полей не вмещают данные и т. д. (рис. 3.9).

Закройте форму.

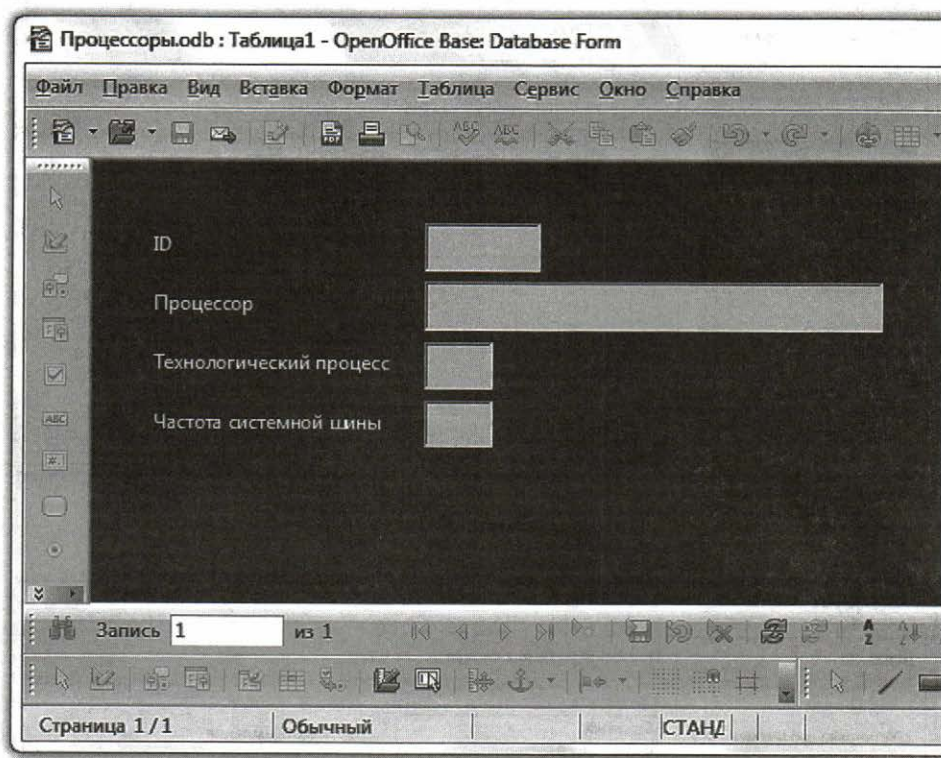


Рис. 3.9

Вид формы можно изменять в *Режиме разработки*. В созданной нами форме базы данных «Процессоры» не все надписи хорошо видны, поэтому размер шрифта нужно увеличить. Некоторые поля вывода числовых значений также нужно увеличить, чтобы числовые данные в них помещались.

8. Для перехода в режим разработки в главном окне базы данных вызвать контекстное меню для созданной *Формы 1* и выполнить команду *Изменить*.
9. В режиме разработки выделить поле с надписью и осуществить двойной щелчок.

В появившемся диалоговом окне *Свойства* установить необходимые параметры (в том числе размер шрифта).

Проделать это для всех полей (рис. 3.10).

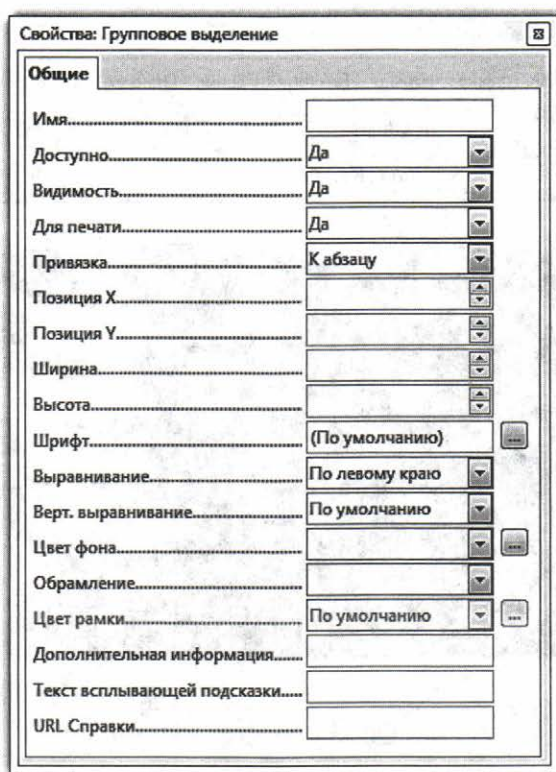


Рис. 3.10

Для изменения параметров надписи и поля по отдельности их нужно разгруппировать.

10. Выделить надпись и поле, в контекстном меню выбрать пункт [Группировка—Разгруппировать]. Прodelать это для всех полей.

Теперь форма для базы данных «Процессоры» готова. С её помощью можно просматривать записи, редактировать их или вводить новые записи.

После открытия форма содержит запись № 1. При работе с формой для перехода от одной записи к другой необходимо воспользоваться панелью *Запись*, которая находится в нижней части окна формы.

11. В окне *Форма 1* щёлкнуть по кнопке *Режим разработки*.

С использованием панели *Запись*, которая содержит кнопки со стрелками, можно перемещаться по записям.

Также можно в поле номера записи ввести номер искомой записи (рис. 3.11).

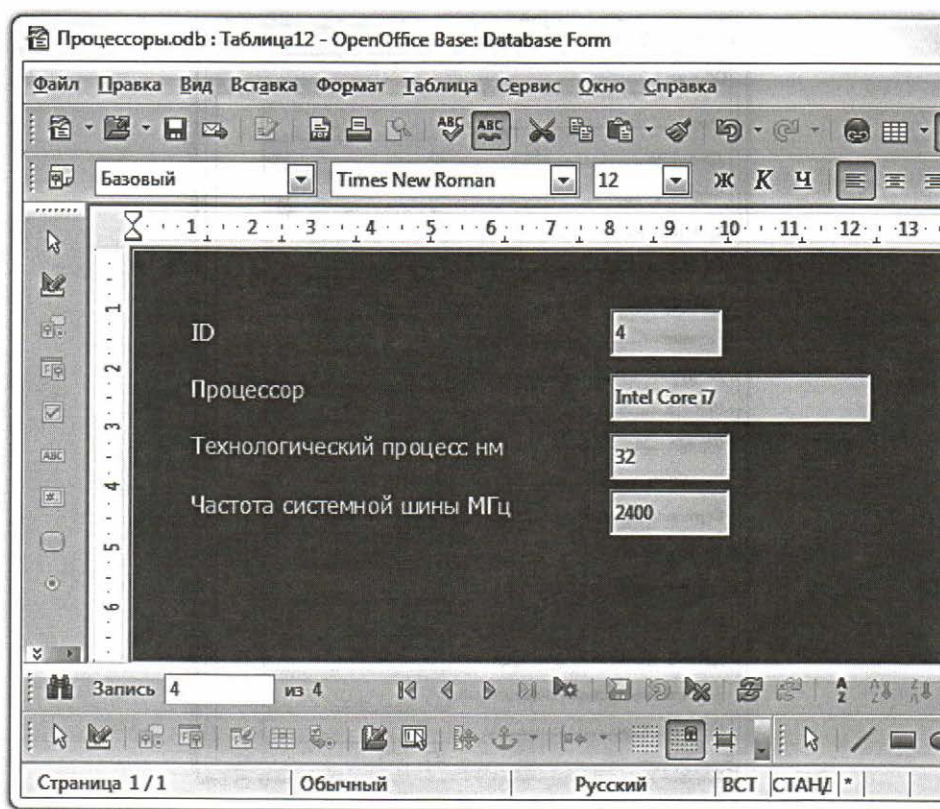


Рис. 3.11

3.2.3. Поиск записей в базе данных с помощью фильтров и запросов

Поиск записей с помощью фильтров. Фильтры позволяют отбирать записи, которые удовлетворяют заданным условиям. Условия отбора записей создаются с использованием операторов сравнения (=, >, < и т. д.).

Простой фильтр содержит условие отбора записей только для одного поля. **Сложный фильтр** содержит несколько условий для различных полей. В результате применения сложного фильтра будут отобраны только те записи, которые удовлетворяют всем условиям одновременно.

Поиск записей с помощью запросов. Запросы осуществляют поиск записей в табличной базе данных так же, как и фильт-

ры. Различие между ними состоит в том, что запрос является самостоятельным объектом базы данных, а фильтр привязан к конкретной таблице.

Запрос является производным объектом от таблицы. Однако результатом выполнения запроса является также таблица, т. е. запросы могут использоваться вместо таблиц. Например, форма может быть создана как для таблицы, так и для запроса.

Запросы позволяют отобрать те записи, которые удовлетворяют заданным условиям. Запросы, как и фильтры, бывают простые и сложные. **Простой запрос** содержит одно условие, а **сложный запрос** содержит несколько условий для различных полей.

В процессе создания запроса можно отбирать не только записи, но и поля, которые будут присутствовать в запросе.

Алгебра логики. Создание (формулирование) запросов тесно связано с понятиями алгебры логики.

Методы алгебры логики помогают нам формализовать логические рассуждения и с их помощью реализовать компьютерную обработку и поиск информации. Подобно тому как в обычной алгебре используются переменные и математические операции, алгебра логики позволяет формулировать логические выражения в абстрактной форме, используя логические переменные и выполняемые над этими переменными логические операции.

Исходными объектами в алгебре логики являются **высказывания**. Высказывание представляет собой повествовательное предложение, которое содержит некоторое утверждение о рассматриваемом объекте и может быть истинным или ложным.

Примеры высказываний:

- 1) «Число 2 чётное» — истинное высказывание;
- 2) «Марс — спутник Земли» — ложное высказывание.

Вопросительные и восклицательные предложения, а также фразы, в которых содержится только название объекта без каких-либо утверждений о нём (например: «белая ворона»), *не являются высказываниями*.

Высказывание можно обозначить **логической переменной**. В зависимости от того, является ли высказывание истинным или ложным, соответствующая ему логическая переменная тоже является истинной или ложной. Истинность в алгебре логики обычно обозначают словом **ИСТИНА**, словом *true* или числовым значением 1. Ложность обозначают словом **ЛОЖЬ**, словом *false* или числовым значением 0.

Само высказывание также может содержать в себе одну или несколько логических переменных, значения которых определяют истинность или ложность высказывания. Такое высказывание,

содержащее логические переменные, называют **высказывательной формой**.

Высказывание может быть простым (содержит одно утверждение об одном объекте) или сложным, составленным из нескольких простых высказываний (в том числе обозначенных логическими переменными). В простых высказываниях чаще всего используются операции сравнения: «равно», «не равно», «меньше», «больше», «меньше или равно», «больше или равно». Если операции сравнения содержат математические переменные, то истинность или ложность такого простого высказывания зависит от значений этих переменных. При поиске записей в табличной базе данных простые фильтры и простые запросы формулируются в виде простых высказываний.

Сложное высказывание формулируется из простых при помощи **логических операций**, которые позволяют по истинности или ложности связываемых ими простых высказываний (**операндов**) однозначно определить истинность или ложность результата. Каждая логическая операция описывается **таблицей истинности**, содержащей все возможные комбинации значений операндов и соответствующие им значения результата выполнения логической операции.

Пример:

$A = \langle X > 1 \rangle$ и $B = \langle X < 3 \rangle$ — простые высказывания, обозначенные переменными A и B .

Сложное высказывание « A И B » ($\langle X > 1 \rangle$ И $\langle X < 3 \rangle$) образовано соответствующими операндами и связано логической операцией И.

Рассмотрим основные логические операции.

НЕ (отрицание, инверсия). Обозначается символом $\neg$ или чертой над логической переменной (надчёркиванием).

Логическая операция НЕ имеет только один операнд и меняет его значение с истинного на ложное и наоборот.

Таблица истинности операции НЕ:

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

И (логическое умножение, конъюнкция). Обозначается символом $\wedge$ или &.

Результат логической операции И является истинным только в том случае, когда оба операнда являются истинными.

Таблица истинности операции И:

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

ИЛИ (логическое сложение, дизъюнкция). Обозначается символом $\vee$, $+$ или $|$.

Результат логической операции ИЛИ является истинным, если хотя бы один операнд является истинным.

Таблица истинности операции ИЛИ:

A	B	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

Исключающее ИЛИ (неравнозначность). Обозначается символом $\oplus$.

Результат логической операции исключающее ИЛИ является истинным, только если её операнды имеют различные значения истинности.

Таблица истинности операции исключающее ИЛИ:

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

Операция исключающее ИЛИ может быть заменена сочетанием операций отрицания, конъюнкции и дизъюнкции: $A \oplus B = (A \wedge \bar{B}) \vee (\bar{A} \wedge B)$.

ЕСЛИ...ТО (следование, импликация). Обозначается символом $\rightarrow$.

Результат импликации является ложным только тогда, когда из истины следует ложь. Во всех остальных случаях результат импликации истинен.

Таблица истинности операции импликации:

A	B	$A \rightarrow B$
0	0	1
0	1	1
1	0	0
1	1	1

Операция импликации может быть заменена сочетанием операций отрицания и дизъюнкции: $A \rightarrow B = \bar{A} \vee B$.

ТОГДА И ТОЛЬКО ТОГДА (эквиваленция, тождество). Обозначается символом $\leftrightarrow$ или $\sim$.

Результат эквиваленции является истинным, только если оба её операнда имеют равные значения (оба равны 0 или оба равны 1).

Таблица истинности операции эквиваленции:

A	B	$A \sim B$
0	0	1
0	1	0
1	0	0
1	1	1

Операция эквиваленции может быть заменена сочетанием операций отрицания, конъюнкции и дизъюнкции: $A \sim B = (\bar{A} \vee B) \wedge (A \vee \bar{B})$ или $A \sim B = (\bar{A} \wedge \bar{B}) \vee (A \wedge B)$.

Порядок выполнения логических операций в сложном высказывании определяется их приоритетом. Наивысший приоритет имеет операция НЕ. Второй по порядку выполняется операция И. После операций И выполняется операция ИЛИ. После дизъюнкции выполняются операции исключающее ИЛИ и импликация. Самый низкий приоритет имеет операция эквиваленции. Порядок выполнения логических операций может быть изменён при помощи скобок.

Сложные логические выражения можно преобразовать так, чтобы упростить запись (сократить количество выполняемых

логических операций) или заменить одни логические операции (исключающее ИЛИ, импликацию, эквиваленцию) на другие (операции НЕ, И, ИЛИ). Такие преобразования выполняются согласно следующим законам алгебры логики.

1. Законы ассоциативности и коммутативности:

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

2. Законы дистрибутивности:

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

3. Законы идемпотентности:

$$A \wedge A = A, A \vee A = A.$$

4. Закон двойного отрицания:

$$\overline{\overline{A}} = A.$$

5. Закон непротиворечия:

$$A \wedge \overline{A} = 0.$$

6. Закон исключённого третьего:

$$A \vee \overline{A} = 1.$$

7. Свойства логических констант:

$$\overline{1} = 0, A \wedge 1 = A, A \vee 1 = 1, \overline{0} = 1, A \wedge 0 = 0, A \vee 0 = A.$$

8. Законы де Моргана:

$$\overline{A \wedge B} = \overline{A} \vee \overline{B}, \overline{A \vee B} = \overline{A} \wedge \overline{B}.$$

Для сложного высказывания тоже можно составить таблицу истинности. Такая таблица должна содержать все возможные комбинации исходных значений используемых в выражении переменных и для каждой такой комбинации значений переменных — все результаты выполнения имеющихся логических операций и результат всего выражения.

Пример: логическое выражение $(A \wedge \overline{B}) \vee (\overline{A} \wedge B)$.

Его таблица истинности:

A	B	$\overline{A}$	$\overline{B}$	$A \wedge \overline{B}$	$\overline{A} \wedge B$	$(A \wedge \overline{B}) \vee (\overline{A} \wedge B)$
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

При поиске записей в табличной базе данных сложные фильтры и сложные запросы формулируются в виде сложных высказываний с использованием рассмотренных выше логических операций.

Кроме логических операций в алгебре логики также существуют **кванторы** — особые операции, определяющие степень общности высказывания.

1. **Квантор общности** («для всех»). Обозначается символом $\forall$.

Пример записи выражения: « $\forall x \ x \cdot 0 = 0$ » означает «Для всех x умножение на нуль дает в результате нуль».

2. **Квантор существования** («существует»). Обозначается символом $\exists$.

Пример записи выражения: « $\exists x \ x$ — чётное» означает «Существуют значения x , которые являются чётными».

3. **Квантор единственности** («существует одно и только одно»).

Обозначается символами $\exists!$.

Пример записи выражения: « $\exists! x \ x \cdot x = 1$ » означает «Существует только одно значение x , произведение которого само на себя равно единице».



Вопросы и задания

В чём заключается разница между поиском записей в табличной базе данных с помощью фильтров и запросов?

Практическая работа 3.3

Поиск записей в базе данных с помощью фильтров и запросов

Задание 1. В табличной базе данных «Процессоры» осуществить поиск записей с использованием простого фильтра (например, будем искать записи, удовлетворяющие условию: технологический процесс больше 32 нм).

Задание 2. В базе данных «Процессоры» осуществить поиск записей с использованием сложного запроса.

Варианты выполнения работы:

- задать различные условия поиска записей в фильтрах и запросах.



Поиск записей с использованием простого фильтра в базе данных «Процессоры»



Электронное приложение к главе 3.

1. Открыть *Таблицу 1* базы данных «Процессоры», дважды щёлкнув по соответствующей ссылке в окне базы данных.

Появится таблица базы данных (рис. 3.12) с кнопками на *Панели инструментов*, позволяющими создавать и отменять фильтры (рис. 3.13).

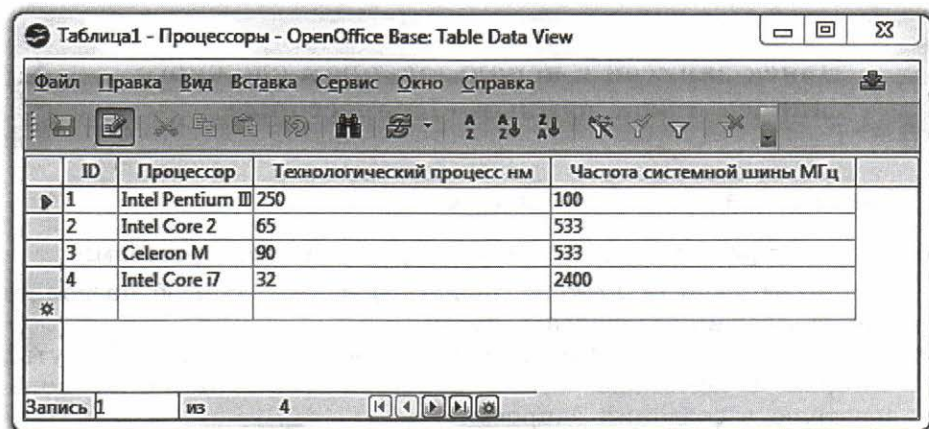


Таблица1 - Процессоры - OpenOffice Base: Table Data View

ID	Процессор	Технологический процесс нм	Частота системной шины МГц
1	Intel Pentium III	250	100
2	Intel Core 2	65	533
3	Celeron M	90	533
4	Intel Core i7	32	2400

Запись 1 из 4

Рис. 3.12

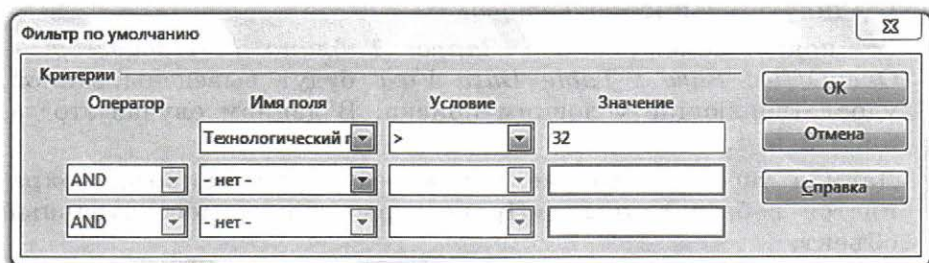


Рис. 3.13

- Щёлкнуть по кнопке , позволяющей выбрать параметры фильтра. В появившемся диалоговом окне *Фильтр по умолчанию* ввести условия поиска записей (*Имя поля*, *Условие* и *Значение*). Фильтр будет создан (рис. 3.14).

Щёлкнуть по кнопке *ОК*.

В появившемся окне *Процессоры: Таблица 1* будут выведены записи, удовлетворяющие условиям поиска. В данном случае это записи 2, 3 и 4.



Фильтр по умолчанию

Критерии	Оператор	Имя поля	Условие	Значение
		Технологический процесс	>	32
AND		- нет -		
AND		- нет -		

ОК, Отмена, Справка

Рис. 3.14

3. Для того чтобы отменить фильтр и вернуть все записи базы данных, щёлкнуть по кнопке .



Поиск записей с использованием сложного запроса в базе данных «Процессоры»



Электронное приложение к главе 3.

Создание запроса можно проводить различными способами:

- самостоятельно (в *Режиме дизайна*) — этот способ позволяет начать создание запроса с нуля;
- с использованием *Мастера запросов* (более простой путь), который с помощью серии диалоговых окон помогает пользователю в создании запроса.

Создадим сложный запрос для табличной базы данных «Процессоры» с помощью *Мастера запросов*.

1. В левой части диалогового окна *Процессоры* в разделе *База данных* выбрать пункт *Запросы*, в средней части окна в разделе *Задачи* выбрать пункт *Использовать мастер для создания запроса ...*
2. В появившемся диалоговом окне *Мастер запросов* (шаг 1: *Выбор полей*) выбрать поля базы данных из *Таблицы 1*, которые будут отображаться в запросе (рис. 3.15).
Щёлкнуть по кнопке *Дальше>*.
3. В диалоговом окне *Мастер запросов* (шаг 3: *Условие поиска*) выбрать поля базы данных из *Таблицы 1*, *Условия* и *Значения*, которые будут в запросе (рис. 3.16).

Щёлкнуть по кнопке *Готово*.

В появившемся окне *Запрос_Таблица 1—Процессоры—OpenOffice Base : Table Data View* будут выведены записи, удовлетворяющие условиям поиска. В данном случае это записи 2 и 4.

Данная таблица, полученная в результате запроса, сохраняется в базе данных «Процессоры» как самостоятельный объект.

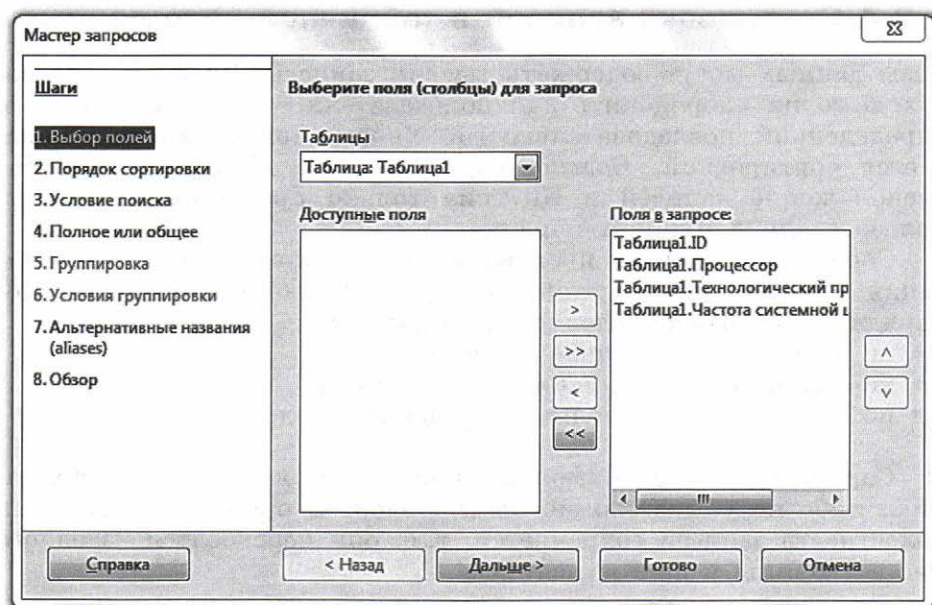


Рис. 3.15

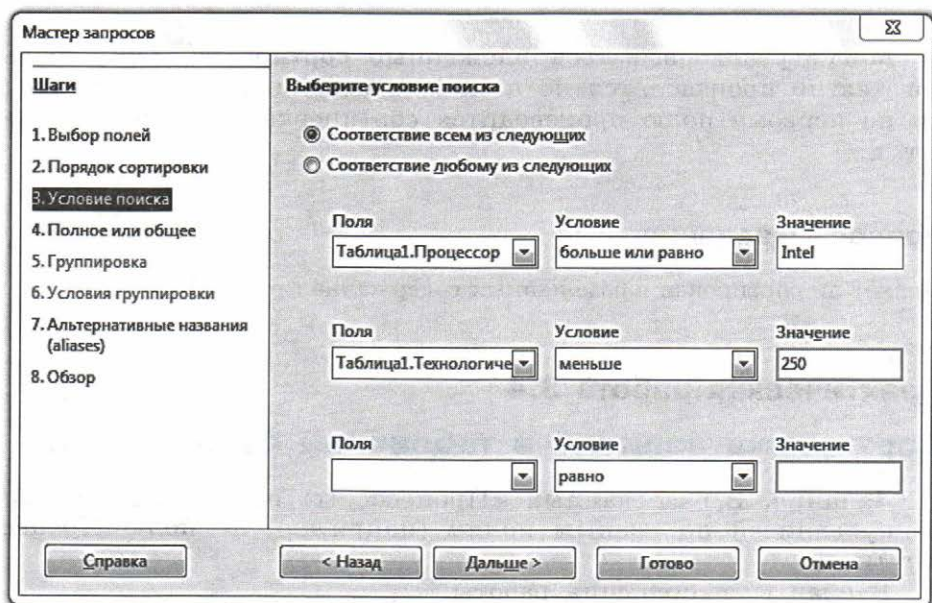


Рис. 3.16

3.2.4. Сортировка записей в табличной базе данных

Базы данных могут содержать тысячи записей. Часто бывает необходимо их упорядочить для пользователя, т. е. расположить в определённой последовательности. Упорядочение записей называется сортировкой. *Важно:* сортировка не меняет физическое расположение записей в БД, она только представляет их для пользователя в нужном порядке.

Сортировка записей производится по какому-либо полю. Значения, содержащиеся в этом поле, располагаются в определённом порядке, который определяется типом поля:

- по алфавиту, если поле текстовое;
- по величине числа, если поле числовое;
- по дате, если тип поля Дата/Время и т. д.

Сортировка записей может производиться либо по возрастанию, либо по убыванию значений полей. В процессе сортировки целостность записей сохраняется, т. е. они переносятся из одного места таблицы в другое целиком.



Сортировка записей базы данных — это их упорядочение по значениям одного из полей.

Могут реализовываться **вложенные сортировки**. Они последовательно производятся по нескольким полям. После сортировки по первому полю производится сортировка по второму полю и т. д.



Вопросы и задания

Меняет ли сортировка в базе данных содержание записей?



Практическая работа 3.4

Сортировка записей в табличной базе данных

Задание. В базе данных «Процессоры» осуществить сортировку записей по разным полям (например, по полю *Процессоры*).

Варианты выполнения работы:

- выбрать различные поля для сортировки записей в базе данных.



Задание. Сортировка записей в табличной базе данных «Процессоры»



Электронное приложение к главе 3.



1. Открыть *Таблицу 1* базы данных «Процессоры», дважды щёлкнув по соответствующей ссылке в окне базы данных. Появится таблица базы данных с кнопками на *Панели инструментов* (рис. 3.17), позволяющими сортировать записи по возрастанию и убыванию.



Рис. 3.17

2. Выделить поле базы данных, по которому будет вестись сортировка (например, *Технологический процесс*).

Осуществить сортировку записей по возрастанию, щёлкнув по кнопке  на *Панели инструментов*.

3. Результатом будет отсортированная база данных (рис. 3.18).

ID	Процессор	Технологический процесс, нм	Частота системной шины, МГц
4	Intel Core i7	32	2400
3	Celeron M	90	533
2	Intel Core 2	65	533
1	Intel Pentium III	250	100

Рис. 3.18

3.2.5. Печать данных с помощью отчётов

Для вывода документов на печать используются отчёты. Отчёты являются производными объектами базы данных и создаются на основе таблиц, форм и запросов.

Вопросы и задания

Для чего нужны отчёты в табличной базе данных?



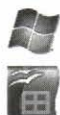
Практическая работа 3.5

Создание отчёта в базе данных

Задание. В табличной базе данных «Процессоры» создать отчёт.

Варианты выполнения работы:

- задать различные дизайны отчётов.



Создание отчёта в базе данных «Процессоры»



Электронное приложение к главе 3.

Создадим отчёт для базы данных «Процессоры» с помощью *Мастера отчётов*.

1. В левой части диалогового окна *Процессоры* в разделе *База данных* выбрать пункт *Отчёты*, в средней части окна в разделе *Задачи* выбрать пункт *Использовать мастер для создания отчёта ...*
 2. В диалоговых окнах *Мастер отчётов* выполнить шаги 1–6 самостоятельно.
- Будет получен один из вариантов отчёта (рис. 3.19).

ID	Процессор	Технологический процесс, нм	Частота системной шины, МГц
1	Intel Pentium III	250	100
2	Intel Core 2	65	533
3	Celeron M	90	533
4	Intel Core i7	32	2400

Рис. 3.19

3.3. Иерархическая модель данных

Иерархическая модель данных графически может быть представлена как перевёрнутое дерево, состоящее из объектов различных уровней. Верхний уровень (**корень**) занимает один объект, второй — объекты второго уровня и т. д.

Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении **предка** (объект, более близкий к корню) к **потомку** (объект более низкого уровня), при этом объект-предок может не иметь потомков или иметь их несколько, тогда как объект-потомок обязательно имеет только одного предка. Объекты, имеющие общего предка, называются **близнецами**.

Базой данных, основанной на иерархической модели, является Реестр Windows, в котором хранится вся информация, необходимая для нормального функционирования компьютерной системы (данные о конфигурации компьютера и установленных драйверах, сведения об установленных программах, настройки графического интерфейса и др.). Содержание реестра автоматически обновляется при установке нового оборудования, инсталляции программ и т. д.

Вопросы и задания



1. В чём заключаются характерные особенности иерархических моделей данных?
2. Что такое распределённая база данных?

3.4. Сетевая модель данных

Сетевая модель данных является обобщением иерархической за счёт того, что на связи между объектами в сетевых моделях не накладывается никаких ограничений.

Сетевой моделью данных является, например, генеалогическое древо семьи, так как потомки (объекты нижележащего уровня) имеют всегда более одного предка (объекта вышестоящего уровня).

Программа GenoPro (Живая Родословная) предназначена для создания, редактирования, распечатывания и представления в Интернете больших генеалогических деревьев, которые являются сетевыми моделями. Особенностью программы является лёгкость, с которой можно создавать очень сложные родословные, включающие иллюстративные материалы и различные комментарии.



Вопросы и задания

В чём заключаются характерные особенности сетевой модели данных?



Практическая работа 3.6

Создание генеалогического древа семьи

Задание. Создать базу данных «Генеалогическое древо семьи». Варианты выполнения работы:

- построить генеалогическое древо разных семей.



Создание базы данных «Генеалогическое древо семьи»



Электронное приложение к главе 3.

Создадим основанную на сетевой модели данных базу данных «Генеалогическое древо семьи» в программе GenoPro (Живая Родословная).

1. Запустить программу GenoPro (Живая Родословная) командой [Все программы-GenoPro].
2. В появившемся диалоговом окне программы на *Панели инструментов* выбрать себя, т. е. воспользоваться кнопками  (*Новый мужчина*) или  (*Новая женщина*).
3. С помощью кнопки  на *Панели инструментов* вставить в генеалогическое древо своих родителей.
4. Поочерёдно выделить родителей и с помощью той же кнопки  вставить в генеалогическое древо родителей своих родителей, т. е. дедушек и бабушек.
5. Поочерёдно выделить всех родственников в генеалогическом древе семьи и в контекстном меню объектов выбрать пункт *Свойства*. В появившемся диалоговом окне ввести сведения о родственнике (рис. 3.20).

В результате мы получим генеалогическое древо семьи, которое является примером сетевой модели данных (рис. 3.21).

Персона - свойства

Гиперссылки		Отображение		Семья		Вклады пользователя	
Общие данные	Рождение	Смерть	Образование	Занятость	Контакты	Изображения	Источники
Титул	Имя	Отчество	Изображение				
Фамилия/девичья фамилия	Последняя фамилия	Оуффико					
Прозвище/псевдоним	Альтернативные имена						
Полное имя							
Отображаемое имя							
Сын 3 строк							
Дата рождения	Пол						
Дата смерти	☒ Мужской ☐ Женский ☐ Животное ☐ Неизвестно						
Возраст							
Безопасность							
Род занятий/профессия		Организация/работодатель					
Домашний телефон	Мобильный телефон	Рабочий телефон					
Электронная почта		Послать e-mail					
OK Отмена Помощь							

Рис. 3.20

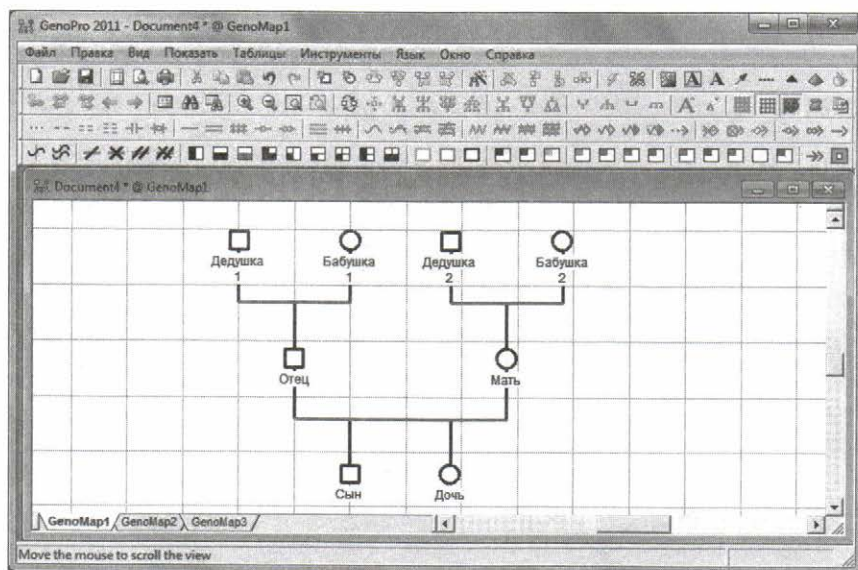


Рис. 3.21

ЭОР к главе 3 на сайте ФЦИОР (<http://fcior.edu.ru>)

- Ввод данных в БД
- Запросы на выборку данных
- Методы и средства разработки приложений. Системы управления базами данных
- Назначение и классификация баз данных
- Поиск информации в БД. Формирование условий
- Понятие СУБД. Классификация СУБД
- Проектирование баз данных
- Проектирование объектов данных
- Проектирование отчётов
- Проектирование экранных форм
- Табличное и графическое представление данных

Глава 4

СОЦИАЛЬНАЯ ИНФОРМАТИКА

При изучении данной главы рекомендуется установить следующее программное обеспечение:

для операционной системы Windows:

- программу разработки презентаций Microsoft PowerPoint;
- текстовый редактор Microsoft Word;
- браузеры Mozilla FireFox и Opera;

для операционных систем Windows и Linux:

- текстовый редактор OpenOffice Writer;
- программу разработки презентаций OpenOffice Impress.



4.1. Информационное общество

В современном человеческом обществе информация представляет собой наиболее важный, ключевой ресурс. В деятельности всё большего числа людей начинает преобладать именно производство, хранение, переработка и реализация (продажа) информации как единственного продукта труда. Эта информация может быть представлена в различных видах — как новостная, справочная, инновационная («ноу-хау»), организационно-управленческая, экономическая, научная и т. д. Высшей формой информации являются знания. В качестве примера производства информации для последующей её продажи можно привести создание художественного информационного контента (книги, музыка, фильмы и пр.), а также разработку программного обеспечения. Общество, достигшее настолько высокой степени развития, при которой основным ресурсом и результатом труда становится именно информация, а не материальные и энергетические ресурсы, принято называть **информационным обществом**.

Основные признаки информационного общества:

- возрастание роли информации, знаний и информационных технологий в жизни общества;
- увеличение числа людей, занятых информационными технологиями, коммуникациями и производством информационных продуктов и услуг;
- рост информатизации общества за счёт использования телефонии, радио, телевидения, сети Интернет, а также традиционных и электронных средств массовой информации (СМИ);
- создание глобального информационного пространства, обеспечивающего эффективное информационное взаимодействие граждан различных стран, их доступ к мировым информационным ресурсам и удовлетворение их потребностей в информационных продуктах и услугах;
- всё более глубокое проникновение информационных технологий в социальную, политическую и экономическую сферы, в процессы управления государством и взаимодействия органов власти с населением (электронная демократия, электронное государство, электронное правительство).

Примерами реализации концепции «электронного правительства» в России являются сайты Президента РФ (рис. 4.1), Государственной Думы (рис. 4.2), правительства Москвы (рис. 4.3), министерств и ведомств (рис. 4.4, 4.5), где любой гражданин России или житель другой страны может не только получить всю необходимую нормативную и правовую информацию, но и обратиться с просьбой, предложением или жалобой в соответствующее министерство или к должностному лицу.

Существуют также специальные сайты для обслуживания населения. Примеры:

- Портал городских услуг Москвы (<https://pgu.mos.ru>, рис. 4.6) — для дистанционного заказа, а в некоторых случаях — и получения в электронной форме тех или иных услуг (например, возможность заказа необходимой справки, получения загранпаспорта или записи в образовательное учреждение), для записи на приём к врачу или в образовательное учреждение и т. д.;
- сайты «Москва наш город» (<http://gorod.mos.ru>, рис. 4.7), «Наш Санкт-Петербург» (<http://gorod.gov.spb.ru>, рис. 4.8) и аналогичные, предназначенные для обращений жителей соответствующих городов с жалобами на те или иные городские проблемы и предложения по их решению;
- сайты для заказа билетов (на самолёт, поезд, в кино или театр) и т. д.



Рис. 4.1

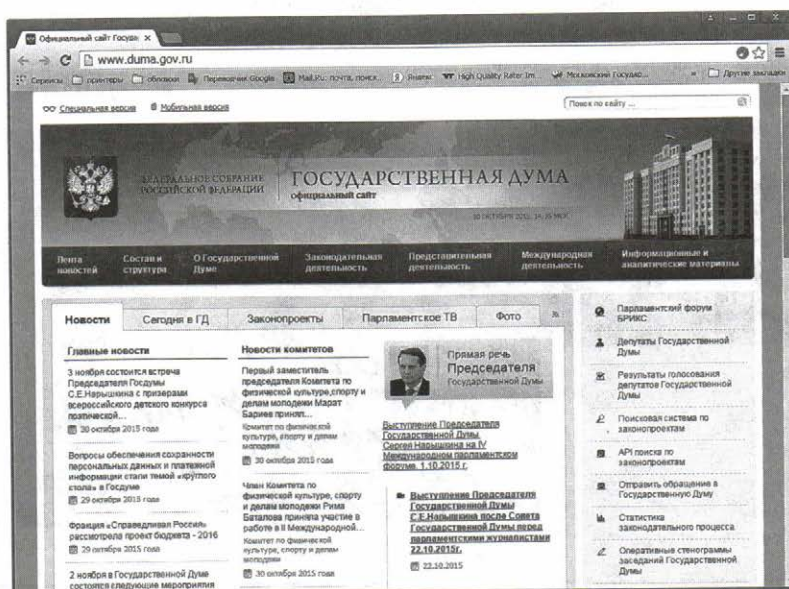


Рис. 4.2

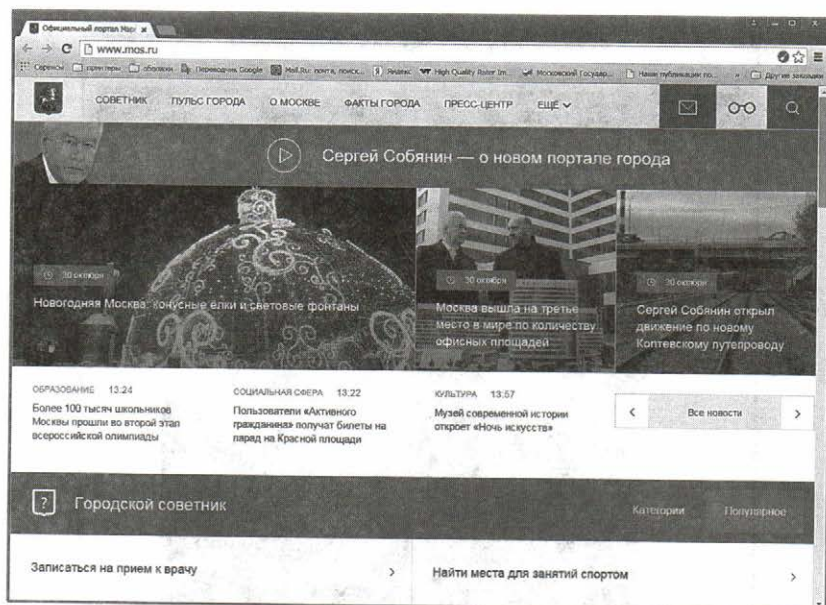


Рис. 4.3

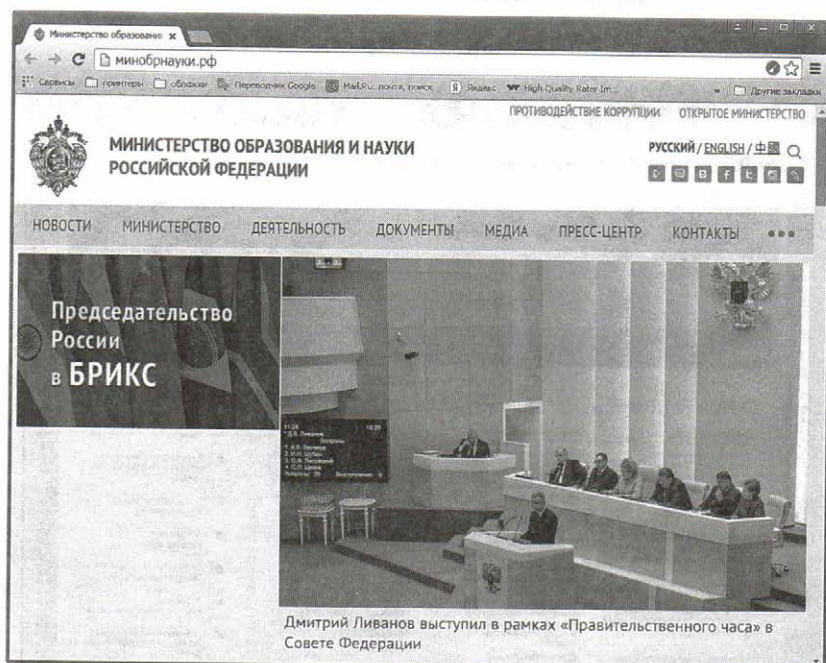


Рис. 4.4

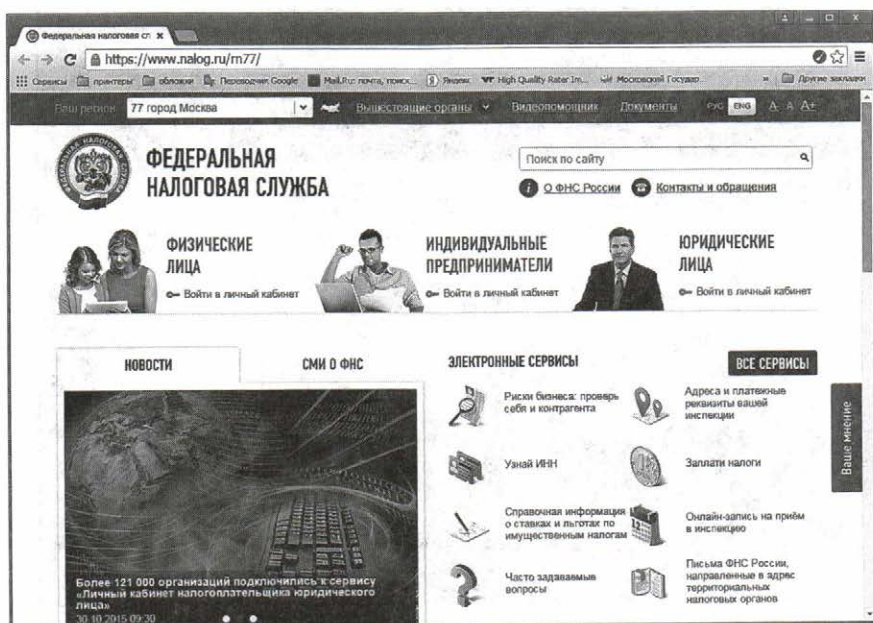


Рис. 4.5

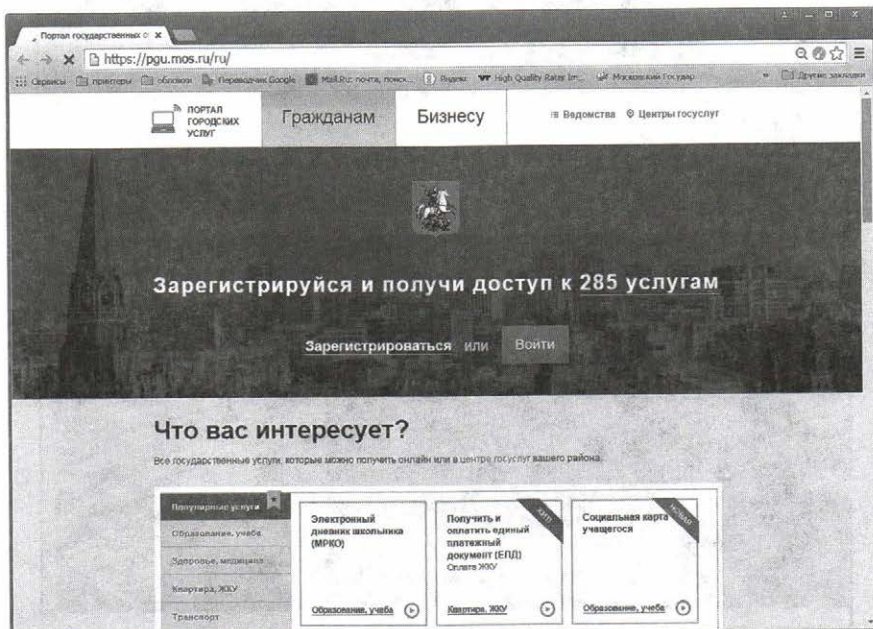


Рис. 4.6



Рис. 4.7

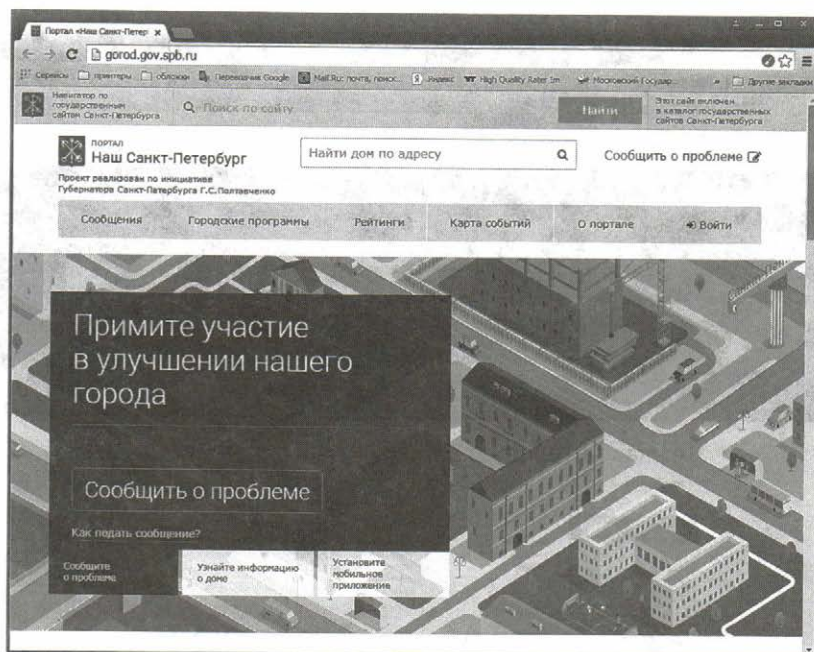


Рис. 4.8

Кроме уже упомянутых существуют сайты интернет-магазинов, музеев (в том числе с виртуальными экспозициями), электронных СМИ, социальных сетей и пр. Возрастает и количество точек доступа в компьютерную сеть Интернет (в том числе бесплатного) по беспроводным каналам Wi-Fi (в кафе и гостиницах, в транспорте — например, в метро, в парках и даже на автобусных остановках), развиваются сети высокоскоростного беспроводного мобильного доступа в Интернет (3G, 4G, Yota и др.). В итоге вся совокупность информационных ресурсов, доступных грамотному пользователю компьютера с доступом в Интернет, формирует информационную среду, предоставляющую все необходимые для жизни, профессиональной деятельности, досуга возможности и услуги. При этом в будущем предполагается слияние подобных информационных сред отдельных городов и стран в единую глобальную общемировую информационную среду, доступную любому жителю нашей планеты независимо от его реального местоположения, — среду, которая представляет собой единую систему из отдельных информационных элементов.

4.1.1. Информационная культура

Понятие «информационная культура» характеризует грань жизни людей, связанную с информационным аспектом жизни в условиях растущего влияния научно-технического прогресса на общую культуру как общества в целом, так и отдельной личности.

Под **информационной культурой** понимается:

- осознание природы информации, её видов и свойств, определяющих качество получаемой информации;
- знание основных технологий и технических средств получения, хранения, обработки, передачи и представления информации различных видов и форм представления;
- знание надёжных источников достоверной и объективной информации;
- умение применять различные технические средства (прежде всего аппаратные и программные средства компьютерной техники) для работы с информацией в профессиональной и повседневной бытовой деятельности, а также умение выбирать наиболее оптимальные средства и возможности для решения конкретных задач по работе с информацией;
- умение критически оценивать получаемую информацию и выбирать достоверные, объективные сведения, отделяя их от непроверенной и недостоверной информации;
- навыки самообучения (самостоятельного получения знаний путём целенаправленного поиска, сопоставления информации и практического экспериментирования);

- способность к самостоятельному критическому мышлению, способность осознавать, где и как полученные знания могут быть применены в профессиональной и бытовой деятельности, способность к творческому мышлению и генерированию новых идей;
- морально-нравственные и этические аспекты, связанные с получением, хранением, обработкой, представлением информации и с общением (соблюдение правовых норм, связанных с информационной сферой и коммуникациями, осознание необходимости соблюдать общепринятые правила общения и поведения в обществе, понимание общечеловеческих моральных и культурных ценностей, осознание своей сопричастности к социуму);
- формирование и развитие личностных качеств, обеспечивающих коммуникабельность, способность к совместной работе в различных профессиональных сообществах и социальных группах, отсутствие расовой, религиозной, этнокультурной нетерпимости.

Более коротко можно сформулировать информационную культуру как такую совокупность качеств личности, благодаря которым другим людям удобно и приятно взаимодействовать (общаться, иметь общие дела) с данной личностью.



Вопросы и задания

1. Что такое информационное общество? Насколько, по вашему мнению, современное общество соответствует этому определению?
2. Как вы понимаете смысл концепции электронного правительства? Приведите примеры её реализации.
3. Что такое информационная культура? Приведите примеры, свидетельствующие, по вашему мнению, о наличии или отсутствии у конкретного человека информационной культуры.

4.2. Правовые основы информационной среды

Вся совокупность различных технических средств для получения, хранения, обработки, передачи и представления информации, доступных информационных и коммуникационных ресурсов и услуг, а также политические, экономические, социальные и культурные условия их реализации представляет собой **информационную среду**, в которую оказывается погружён каждый член современного человеческого общества практически с самого рождения.

Высокий уровень развития информационной среды обеспечивает гражданам более удобную и комфортную среду обитания,

однако вместе с тем накладывает на них определённые обязанности, связанные с корректным использованием предоставляемых информационных возможностей с правовой и морально-этической точки зрения.

Авторское право — это один из наиболее существенных правовых аспектов, касающихся получения и использования информации. Оно регулирует правоотношения, связанные с созданием и распространением (изданием, исполнением, копированием и т. д.) различных произведений науки, литературы и искусства (в том числе музыки и кино/видео) как результатов творческой деятельности, а также программ для компьютеров и баз данных.

Основным субъектом авторского права является автор (или коллектив соавторов), создавший то или иное произведение. Автору(ам) изначально принадлежат все права на это произведение — личные неимущественные права и исключительное имущественное право на использование произведения в любой форме.

К личным неимущественным авторским правам относятся:

- право авторства — право быть признанным общественностью в качестве автора произведения;
- право на имя — право самому использовать своё произведение или разрешать использовать его другим с указанием подлинного имени автора, под псевдонимом или анонимно (без указания имени);
- право на обнародование — право опубликовать или разрешить кому-либо публикацию произведения в любой форме, либо отозвать произведение из публикации;
- право на защиту репутации автора — право на защиту самого произведения и его названия от любых искажений или иных действий, наносящих ущерб чести и достоинству автора.

Указанные личные неимущественные права возникают автоматически уже по факту создания автором того или иного произведения и являются бессрочными. В России эти права могут принадлежать только автору как физическому лицу и не могут быть переданы кому-либо другому (например, нельзя сделать так, чтобы автором произведения стал считаться другой человек вместо настоящего автора). При этом действует «презумпция авторства»: автором считается тот, кто указан в качестве автора на оригинале или на экземпляре копии произведения, если не будет доказано иное.

Таким образом, от автора не требуется никаких действий для официальной регистрации своего личного авторского права — достаточно лишь создать произведение и указать на нем себя в качестве автора.



Имущественное право, в отличие от рассмотренной выше категории неимущественных прав, определяет коммерческую принадлежность произведения. Изначально исключительное имущественное право принадлежит автору произведения — именно автор может решать, как, где и когда воспроизводить (копировать), распространять, публиковать (исполнять) или иным способом доводить своё произведение до всеобщего сведения, а также разрешать перевод произведения на другие языки и/или другую его переработку. Но своё имущественное право автор может передать (продать) другим лицам или организациям, например издательствам, радио- или телестудиям и т. д. Такие лица или организации, получившие от автора имущественные права на произведение, называются **правообладателями**.

Принадлежащее ему имущественное право автор может продать или передать правообладателю:

- как исключительное имущественное право — в этом случае автор это имущественное право полностью утрачивает и более не может ни сам распоряжаться воспроизведением, распространением и т. д. своего произведения, ни передать имущественное право другим правообладателям;
- как неисключительное имущественное право — в этом случае правообладатель получает права только на разовую публикацию, воспроизведение и т. д., тогда как исключительное имущественное право остаётся у автора (который в том числе может передать его и другим правообладателям).

Обычно передача исключительного имущественного права оформляется официально (заключением договора с правообладателем). Если же такой договор не заключается, то считается, что передано неисключительное право.

Если личное неимущественное авторское право действует бессрочно (т. е. вечно), то имущественное право имеет ограниченный срок действия (который зависит от конкретной страны). Так, в России исключительное право действует в течение всей жизни автора плюс ещё 70 лет после его смерти. После этого произведение переходит в разряд общественного достояния (к которому, в частности, относится и большая часть классических произведений литературы).

Особо рассматривается ситуация, когда произведение создаётся в рамках профессиональной деятельности по заданию руководства организации, в которой работает автор. В этом случае считается, что автор получает за создание произведения заработную плату, поэтому исключительные имущественные права на такое произведение изначально принадлежат организации-работодателю.

Признаком авторского права является **копирайт** — указание фамилии и инициалов автора либо названия организации-правообладателя, предваряемое символом копирайта — буквой «С» в кружочке: © (от английского *copyright* — право на копирование).

Плагиа́т и цитирование. Заимствование части чужого авторского произведения (текста, аудио- или видеозаписи) без указания имени настоящего автора и произведения-источника называют **плагиа́том**. Это деяние также является нарушением авторского права и, согласно закону об охране авторских прав, в случае причинения крупного ущерба автору или правообладателю карается штрафом или лишением свободы. Использовать в своей работе можно только небольшие фрагменты (**цитаты**) из чужих авторских произведений, при этом цитирование должно быть действительно необходимо по смыслу текста работы, объём цитируемых фрагментов должен быть небольшим по сравнению со всем остальным произведением, каждая цитата в тексте обязательно должна заключаться в кавычки и снабжаться информацией о первоисточнике (об авторе и названии исходного произведения).

В настоящее время существует специальное программное обеспечение (например, реализованный в Интернете онлайн-сервис «Антиплагиа́т»), позволяющее проверить свою работу на наличие заимствований из чужих авторских произведений. Этим программным обеспечением широко пользуются преподаватели при проверке ученических работ.

Небольшое исключение делается только для использования произведений и их фрагментов в учебных целях — для подготовки и проведения уроков, без распространения используемого материала. Однако и в этом случае необходимо указание произведения-первоисточника.

4.2.1. Лицензирование программного обеспечения

Программы для компьютеров и базы данных также относятся к произведениям, охраняемым авторским правом. Это означает, что создатель (разработчик) программы автоматически получает на неё все личные неимущественные права. Исключительные имущественные права при этом приобретает автор (если он разрабатывал программу сам, как частное лицо) либо фирма, в которой он работает. (Последнее бывает чаще всего, так как современные программные пакеты в силу их большой сложности разрабатываются коллективами программистов.)

Автор или фирма-правообладатель далее распространяет созданную программу или базу данных в соответствии со своей

маркетинговой политикой — за определённую плату или бесплатно.

Условия распространения программы или базы данных, права и обязанности её пользователя при этом оговариваются в особом документе — **пользовательском соглашении**, которое обычно предлагается прочитать перед началом установки программы и выразить своё согласие, нажав соответствующую кнопку или установив соответствующий переключатель. (Отказ от пользовательского соглашения приводит к прерыванию установки программы.)

Ещё один документ, который пользователь получает при приобретении программного продукта, — это **пользовательская лицензия**, наличие которой является признаком законного приобретения, владения и пользования программным продуктом.

Возможны следующие разновидности программ по условиям их распространения.

Коммерческое (проприетарное) ПО. Распространяется на коммерческой основе (за плату). Пользователю предоставляется только право работать с данной программой. Копирование её другим лицам, внесение в программу любых изменений и т. д. запрещено. Для подтверждения легальности приобретения программы служит специальный код — серийный номер, предоставляемый каждому пользователю индивидуально вместе с лицензией. В некоторых случаях требуется выполнить отдельную операцию регистрации или активации программы (например, через специальный веб-сайт).

Демонстрационные и триал-версии ПО. Как правило, для рекламы коммерческих версий ПО фирмы выпускают демонстрационные версии (демо-версии), распространяемые бесплатно, но с ограниченным функционалом (например, нет возможности сохранять результаты работы либо сохраненный файл — скажем, рисунок, — содержит «водяные знаки» — строки текста с указанием, что он был создан в демонстрационной версии. Разновидностью демо-версий являются триал-версии, работающие только в течение некоторого «испытательного» периода (от нескольких дней до нескольких недель). По истечении этого срока триал-версия перестаёт запускаться либо переходит в режим демо-версии.

Условно-бесплатное ПО (shareware). Распространяется бесплатно, но содержит призыв к пользователям оплатить некоторую сумму, если программа им понравилась и они хотят продолжать с ней работать. Этот призыв может появляться в отдельном окне при запуске программы или при ее завершении; иногда такое окно может несколько мешать работе пользователя. Обычно после оплаты автор программы высылает пользователю специальный код, после ввода которого напоминание об оплате исчезает. Без

выполнения этого требования условно-бесплатная программа все равно остается функциональной, так что оплата за неё остаётся на совести пользователей.

Бесплатное ПО (freeware). Распространяется полностью бесплатно, без каких-либо требований оплаты.

Следует учитывать, что для всех перечисленных выше категорий ПО (в том числе shareware и freeware) условия лицензии запрещают копирование программ другим лицам, внесение в эти программы каких-либо изменений и т. д. Распространять shareware- и freeware-программы может только их автор или правообладатель.

В отличие от этого **свободное программное обеспечение** не имеет такого ограничения. Особая лицензия, предоставляемая пользователям свободного ПО, обычно разрешает не только бесплатно пользоваться программой, но также свободно копировать её, изменять, дорабатывать и т. д. Примером свободного ПО является операционная система Linux и многие прикладные программы для нее. К свободному ПО также относится пакет офисных приложений OpenOffice, являющийся аналогом коммерческого пакета Microsoft Office.

Особой разновидностью также является **открытое программное обеспечение** (ПО с открытым кодом), для которого любой желающий может получить не только саму программу, но и её исходный текст на соответствующем языке программирования (для ознакомления и возможных доработок). Обычно открытое ПО также бывает и свободным, но свободное ПО не обязательно является открытым.

Нелицензионный контент. Произведения — тексты, аудиозаписи, видеозаписи, программы и базы данных, полученные незаконным путём (скопированные у других лиц), являются нелицензионными («пиратскими»). За пользование нелицензионным контентом, а особенно за его незаконное распространение предусмотрена уголовная ответственность. Сайты, содержащие нелицензионный контент, могут быть заблокированы (на время, пока владелец сайта не устранил нарушение, или навсегда); оборудование, используемое для незаконного тиражирования и распространения нелицензионного контента, может быть конфисковано, а виновным может грозить штраф или лишение свободы.

Вопросы и задания

1. Что вы понимаете под информационной средой? Приведите примеры.
2. Что такое авторское право? Какие бывают авторские права?



3. Какие авторские права действуют бессрочно? Какие авторские права ограничены по времени действия и каков срок их действия в России?
4. Что такое плагиат и чем он отличается от цитирования?
5. Что является подтверждением законного приобретения и пользования программным продуктом? Где записаны права и обязанности пользователя?
6. Какие бывают виды программного обеспечения по принципам его распространения? Чем они различаются?
7. Почему крайне не рекомендуется использовать, а тем более распространять нелегальный контент?

Практическая работа 4.1

Законы об охране авторских прав

Задание. Ознакомиться с законодательной базой охраны авторских прав.

Варианты выполнения работы:

- в операционной системе Windows или в операционной системе Linux;
- просмотр веб-страниц в различных браузерах (Mozilla Firefox, Opera, Chrome или др.).



Ознакомление с текстом Гражданского кодекса Российской Федерации (ГК РФ)



Ознакомимся с основными понятиями авторского права и правовыми нормами по охране авторского права.

1. Запустить браузер в операционной системе Windows или в операционной системе Linux и в поле *Адрес:* ввести <http://www.gk-rf.ru/glava70> (доменное имя и путь к сайту с официальными текстами законов и кодексов Российской Федерации — главе Гражданского кодекса РФ «Авторское право»).
2. Самостоятельно ознакомиться со статьями главы 70 ГК РФ, описывающими основные понятия и нормы защиты авторского права.
3. Под руководством учителя обсудить вопросы, связанные с охраной авторских прав на различные объекты авторского права, с цитированием и плагиатом.

4.3. Социальные сервисы и сети

Одним из ярких примеров реализации в жизни пользователей компьютеров информационной среды являются социальные сервисы.

В реальной жизни каждый из нас ежедневно общается с другими людьми, оказывает им или получает от них какие-то услуги и т. д. Такие взаимоотношения людей в обществе (социуме) называют социальными взаимоотношениями.

С переходом социума к стадии информационного общества социальные взаимоотношения также изменяются, получают своё отражение в компьютерной сети Интернет. Сегодня при помощи Интернета можно не только общаться любому количеству людей почти одновременно, независимо от реального местоположения собеседников, но и получать различные услуги дистанционно через компьютерную сеть. Различные сервисы сети Интернет, нацеленные на реализацию социальных взаимоотношений, называют **социальными сервисами**.



Приведём примеры социальных сервисов.

Портал городских услуг (см. рис. 4.6) позволяет (после прохождения регистрации) получить доступ к большинству основных услуг, предоставляемых различными государственными организациями города Москвы. В частности, через этот портал выполняется запись учащихся в школу, на посещение дополнительных занятий в Дворцах детского творчества и т. д.

Сайт Федеральной налоговой службы (см. рис. 4.5) предоставляет возможности получения налоговой информации. Например, на этом сайте можно узнать свой идентификационный номер налогоплательщика (ИНН) и заказать соответствующий официальный документ (свидетельство) о присвоении ИНН.

Сайт Единой медицинской информационно-аналитической системы (ЕМИАС, <http://emias.info>, рис. 4.9) позволяет, указав номер своего медицинского полиса и дату рождения, посмотреть расписания работы врачей своей поликлиники и записаться на приём на выбранную дату и время. При этом предлагается распечатать на принтере талон на приём к выбранному врачу.

Ещё более популярными являются социальные сервисы Интернета, нацеленные на общение пользователей: электронная почта, телеконференции, чаты, IP-телефония, форумы, блоги, сетевые сообщества.

Электронная почта — сервис, предоставляющий возможность текстовой переписки с другими пользователями, пересылки вместе с письмами произвольных вложенных файлов («аттачей»), а также рассылки писем группам пользователей.

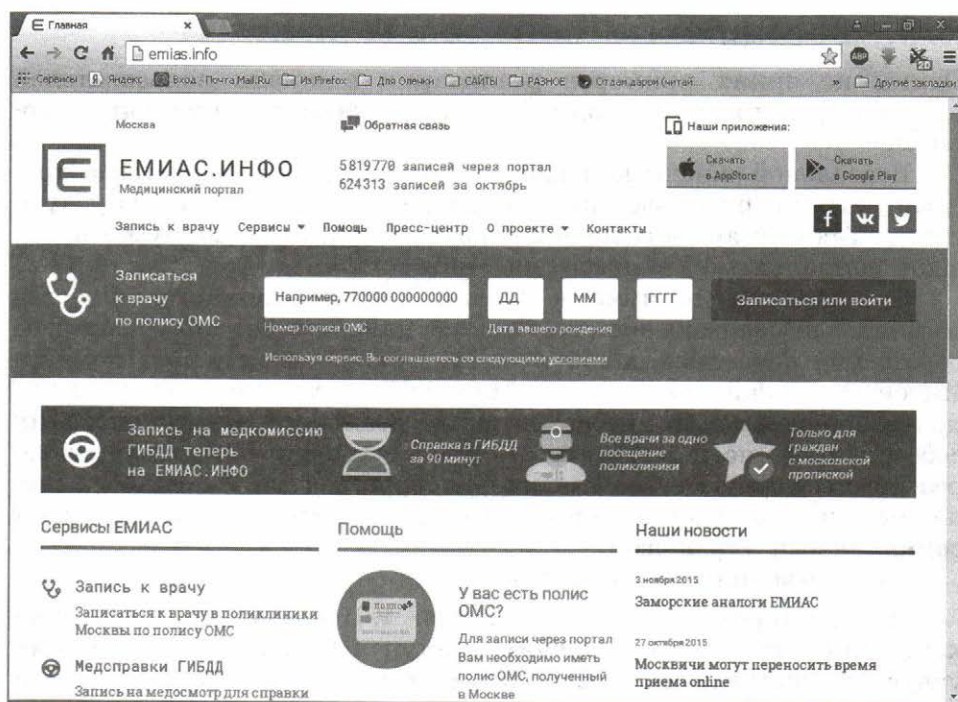


Рис. 4.9

Телеконференции и форумы — сайты, реализующие общение в форме текстовой переписки по определённым темам. После публикации в выбранной теме первого сообщения остальные пользователи могут добавлять к нему свои ответы или дополнения, а также ответы и дополнения на ранее размещённые ответы и дополнения других пользователей. Тем самым в составе темы форума или телеконференции формируется древовидная структура сообщений, в которой можно раскрывать и прочитывать желаемые ветви. Форумы и телеконференции могут быть свободными (немодерируемыми) или контролируруемыми модератором. В качестве модератора может выступать пользователь, создавший тему (написавший в ней первое сообщение) либо, для очень популярных форумов или тем, могут назначаться модераторы из числа наиболее активных пользователей, участвующих в обсуждении.

Современные интернет-телеконференции и форумы часто используются для проведения различных коллективных научных и учебных мероприятий (конференций, семинаров), собраний, совещаний и т. д. При этом возможна реализация видеоконфе-

ренций, предоставляющих участникам возможность при помощи подключённых к компьютерам микрофонов, аудиокколонок и веб-камер общаться голосом (в дополнение к текстовой переписке и демонстрации каких-либо информационных материалов — текстов, иллюстраций, презентаций) и видеть друг друга.

Чаты — сервисы, позволяющие общаться в письменном виде в реальном времени. Как правило, чат предполагает общение сразу множества пользователей. Каждый из зарегистрировавшихся в чате под некоторым именем (**ником**) пользователей может написать и отправить в чат желаемую фразу. Эта фраза, предваряемая ником её автора, появляется на экранах компьютеров всех участников чата в общем списке, и тот, кому была адресована фраза, отыскивает её среди остальных. Однако возможен и так называемый приват-режим, когда для общения выделяется отдельный чат для двух или нескольких пользователей по их желанию.

IP-телефония — сервис, позволяющий реализовать голосовое общение (аналогичное телефонному) через сеть Интернет. Такая услуга по стоимости гораздо дешевле, чем обычная междугородняя телефонная связь, а при звонках с одного компьютера на другой обходится практически бесплатно (за исключением стоимости доступа в Интернет с учётом объёма трафика — передаваемой информации). Современные системы IP-телефонии, такие как Skype, позволяют также реализовать видеосвязь между абонентами при помощи подключённых к их компьютерам веб-камер.

Блоги (сетевые журналы) — это сайты, позволяющие зарегистрированным на них пользователям публиковать свои сообщения — заметки, фотографии и т. д., формируя тем самым свой сетевой дневник, доступный всем другим пользователям либо только выбранным пользователям (сетевым друзьям). Публикации в таких блогах обычно носят субъективный характер и отражают воззрения и эмоции их авторов, но зато в блогах можно найти уникальную информацию, отсутствующую в других источниках.

Википедия и другие вики-ресурсы — сетевые энциклопедии, авторами которых могут быть любые пользователи. Опубликованные ими статьи затем просматриваются другими пользователями, а также специалистами (экспертами), которые исправляют допущенные ошибки.

Интернет-магазины, сайты заказа билетов, туров, гостиниц и пр. — сайты, позволяющие посмотреть представленные товары или услуги (билеты на поезд или самолёт, в театр или кино,

туристические поездки, гостиничные номера и пр.), прочитать информацию о них, включая отзывы пользователей, которые ранее приобрели такие товары или услуги, сравнить цены на один и тот же товар или услугу в разных интернет-магазинах и оформить заказ. При этом возможна дистанционная оплата товара или услуги при помощи имеющейся банковской карты.

Мобильный банк — сервис, обычно реализованный на сайтах крупных банков для их клиентов. Каждый зарегистрированный клиент банка получает на сайте доступ в свой **личный кабинет**, в котором может просматривать состояние своего счёта (оставшуюся сумму, поступления и затраты денег), а также выполнять те или иные денежные операции. Например, можно выполнить оплату коммунальных услуг (за квартиру, электроэнергию, и т. д.), перечислить оплату на свой мобильный телефон и пр.

Дистанционное обучение — возможность получения образовательных услуг заочно, через сеть Интернет, пользуясь своим компьютером. Как правило, обучающийся (клиент дистанционного обучения) получает доступ на сайт образовательного учреждения (например, вуза) — в свой личный кабинет и к коллекции различных учебных материалов — **электронных учебников**, пособий, видеоматериалов, электронных моделей и пр. Обучающийся самостоятельно изучает эти материалы, а также может консультироваться с назначенным для него преподавателем (**тьютором**, **куратором**). Преподаватель также пересылает обучающемуся задания и предлагает для выполнения компьютерные тесты, а затем проверяет результаты работы обучающегося. По завершении курса обучающийся получает соответствующий документ (сертификат, свидетельство или диплом). Это может быть как электронный сертификат, который предлагается скачать с сайта (из своего личного кабинета), так и документ об образовании государственного образца, для получения которого необходимо лично явиться в образовательное учреждение и в некоторых случаях очно сдать экзамен.

Дистанционное обучение позволяет обучаться и повышать свою квалификацию в ведущих образовательных учреждениях России и даже других стран без необходимости ехать туда на проживание на время обучения и без необходимости отрыва от основной работы. В будущем предполагается, что практически всё обучение может стать дистанционным, поэтому можно будет говорить о дистанционном образовании.

Сегодня в школах уже внедряются элементы дистанционного обучения — электронный журнал (дневник), в котором учителя выставляют оценки и сообщают домашнее задание, коллекции цифровых образовательных ресурсов (дополнительных материалов,

иллюстраций, анимаций, аудио- и видеофрагментов, интерактивных моделей и пр.), а также электронные учебники, доступные с любого компьютера как в классе, так и дома. С 2015 года все бумажные школьные учебники, используемые в школах, сопровождаются электронными учебниками.

Сетевые сообщества (социальные сети) — это наиболее крупные комплексные социальные ресурсы, сочетающие в себе возможности блога, телеконференции (форума), а также в отдельных случаях — коллективного игрового сервиса, чата и интернет-магазина. Зарегистрировавшийся в социальной сети пользователь может разместить на своей личной странице некоторый желаемый объем личной информации, публиковать свои текстовые заметки, иллюстрации, аудио- и видеозаписи, размещать свои комментарии к публикациям других пользователей социальной сети и т. д. Публикуемые материалы можно сделать открытыми для всех пользователей социальной сети либо закрытыми, предназначенными только для выбранных пользователей (**друзей, френдов** — от англ. *friend*). Чтобы добавить кого-либо в список своих друзей либо, наоборот, попроситься к кому-нибудь в список друзей, нужно отправить выбранному пользователю запрос на добавление в друзья. При подтверждении пользователем этого запроса он автоматически будет добавлен в ваш список сетевых друзей.

Социальные сети могут быть общими (без выделения какой-то социальной группы пользователей и определённой тематики), тематическими либо ориентированными на определённую группу пользователей. Примерами общих социальных сетей являются Facebook, ВКонтакте и Twitter, а социальная сеть Одноклассники. Ру построена по групповому принципу — для пользователей, которые учатся или ранее учились в одном образовательном учреждении (школе, вузе и т. д.).

Преимуществом социальных сетей является реализация практически всех аспектов социального взаимодействия между людьми. Кроме того, крупные социальные сети, объединяющие множество пользователей, дают возможность достаточно быстро найти нужного человека или максимально быстро распространить требуемую срочную информацию при помощи «репостов» (повторных публикаций ранее опубликованных сообщений другими пользователями).

Недостатки социальных сетей прямо следуют из их преимуществ.

Лёгкость публикации и распространения информации зачастую делает социальные сети инструментом преступников, социальные сети могут использоваться экстремистами для планирова-

ния террористических актов, вербовки сторонников, а также для ведения информационной войны путём публикации заведомо ложной информации, порочащей честь и достоинство государств или государственных лидеров. Примерами таких публикаций являются многочисленные «информационные вбросы», публикуемые специально нанятыми провокаторами («сетевыми троллями»). Особая опасность таких «информационных вбросов», распространяемых через социальные сети, заключается в массовости их воздействия на сознание доверчивых пользователей (прежде всего молодежи, чьё мировоззрение и жизненные позиции ещё окончательно не сформировались). Кроме того, многие средства массовой информации (СМИ) очень часто ошибочно (не проверяя достоверность сведений) либо намеренно публикуют сведения, полученные из социальных сетей, в качестве официальных сообщений.

Возможность «заочного» общения в социальной сети нередко приводит к проявлениям грубости среди общающихся, к оскорблениям, проявлениям агрессии, унижению чести и достоинства. В случае подобных нарушений можно исключить оппонента из своей сферы общения (**забанить** его), предписав сайту социальной сети более не показывать вам сообщения и публикации, сделанные этим пользователем. Однако все остальные пользователи сети продолжают видеть эти сообщения. Другая возможность состоит в подаче жалобы владельцам данного сетевого сервиса: если модераторы социальной сети сочтут жалобу правильной, то могут забанить провинившегося пользователя — запретить ему возможность публиковать что-либо в данной социальной сети под выбранным им ником на некоторое время или навсегда. Наконец, во многих странах приняты законы, согласно которым за подобные публикации их авторов могут привлечь к реальной административной или уголовной ответственности. Например, в России такая ответственность (штраф или лишение свободы) предусмотрена за публикацию и распространение информации экстремистского характера, за призывы к национальной розни, к вражде, к свержению существующего государственного строя. А в некоторых зарубежных странах уголовная ответственность предусмотрена и за личные оскорбления, допускаемые в социальной сети.

Публикация в социальной сети личной конфиденциальной информации может принести вред её владельцу: такая информация может быть использована мошенниками или ворами. Например, известны случаи, когда квартиры пользователей были ограблены потому, что ранее они публиковали в социальной сети фотографии своей «роскошной жизни» и сведения о том, когда они уезжают на отдых в турпоездку, желая похвастаться этим перед другими пользователями.

Ещё один недостаток социальных сетей характерен для практически любых сервисов удалённого общения — это полная или частичная анонимность: в большинстве социальных сетей можно в качестве личной информации указать вымышленные имя и фамилию. Это предоставляет удобство для различных мошенников.

Наконец, основной недостаток социальных сетей заключается в возможном проявлении психологической зависимости от них. Социальная сеть предоставляет искусственную информационную среду, которая начинает подменять собой реальные социальные взаимоотношения между людьми. Пользователи социальной сети зачастую предпочитают общение друг с другом только через компьютер и проводят целые дни, читая и публикуя сообщения, забывая при этом о работе или учёбе, переставая общаться с реальными друзьями и даже с родными людьми. Социальная сеть в подобном случае превращается в «информационный наркотик», существование без которого становится для пользователя почти невозможным. Лишение же возможности общения с социальной сетью зачастую вызывает у «компьютерного наркомана» приступы агрессии, опасные для окружающих.

Таким образом, современные социальные сети представляют собой удобный инструмент для общения и самовыражения, но вместе с тем неконтролируемое использование этого инструмента может приводить к значительным негативным последствиям.

Вопросы и задания



1. Что понимается под социальными взаимоотношениями? Как эти социальные взаимоотношения реализуются в информационном обществе?
2. Что такое социальные сервисы? Приведите примеры социальных сервисов, которыми вам приходилось пользоваться.
3. Что такое дистанционное обучение? Какие элементы дистанционного обучения реализованы в вашей школе?
4. Что представляет собой социальная сеть? Какие сервисы она предоставляет?
5. Каковы, по вашему мнению, преимущества социальных сетей?
6. Опишите недостатки социальных сетей.
7. Пользуетесь ли вы социальными сетями? Какими? Для каких целей?
8. Приходилось ли вам в социальной сети сталкиваться с грубостью, оскорблениями или с заведомо ложными «информационными вбросами»? Каковы были ваши действия?
9. Почему не рекомендуется открыто публиковать в социальной сети личную информацию? Чем это может грозить?

4.4. Информационная безопасность

Риски, связанные с использованием социальными сетями, а также другими интернет-сервисами и компьютерным программным обеспечением, делают необходимыми различные методы и средства обеспечения информационной безопасности.

Информационные угрозы. Различные опасности, которые возникают при работе с информацией (в том числе при работе с компьютером) и могут принести вред её владельцу, называются информационными угрозами. В их число входят:

- непреднамеренное уничтожение информации из-за ошибок пользователя или поломок аппаратуры;
- непреднамеренное искажение информации из-за помех;
- несанкционированный доступ посторонних лиц к конфиденциальной (секретной) информации личного, производственного, коммерческого, финансового характера или к информации, составляющей государственную тайну, кража информации;
- намеренное искажение или уничтожение информации;
- создание вредоносных программ (компьютерных вирусов и других), нацеленных на искажение или уничтожение информации и вывод из строя технических средств для работы с информацией (прежде всего компьютеров);
- намеренное проникновение на чужие компьютеры через компьютерную сеть (хакерские атаки) для использования их вычислительных ресурсов в своих целях;
- нарушение работоспособности или попытка вывода из строя сетевых сервисов;
- распространение искажённой или ложной информации с мошенническими целями;
- нарушения правил этики, морали и нравственности при общении или распространении противоречащей этим правилам информации;
- распространение искаженной или заведомо ложной информации, порочащей честь и достоинство отдельных граждан, организаций или государства в целом.

Важно понимать, что информационные угрозы несут в себе реальную опасность. Например:

- нарушение работоспособности информационных систем и сервисов наносит ущерб их владельцам: отсутствие своевременного доступа к важной информации; невозможность вызвать аварийные, пожарные, медицинские, правоохранительные службы; нарушение работоспособности электронной аппара-

туры на опасных производствах или энергостанциях создают прямую опасность жизни и здоровью граждан;

- похищение личной конфиденциальной информации (паролей доступа, банковской информации, например данных банковской карты), создание и распространение вредоносных программ, активирующих без ведома пользователя те или иные платные услуги или операции с банковскими счетами — это первый шаг преступников к краже принадлежащих владельцу этой информации денежных средств;
- доступ к конфиденциальной информации, связанной с адресом проживания, местом и временем работы и т. д., позволяет преступникам планировать и осуществлять различные преступления против личности владельца информации и его имущества;
- распространение заведомо ложной информации, порочащей честь и достоинство государства, как правило, нацелено на формирование у граждан этого государства или у жителей других стран искажённого представления о происходящих в стране событиях, на формирование экстремистских настроений, национальной вражды, религиозной ненависти и т. д.

Информационная безопасность — это совокупность мер по защите информации и технических средств для работы с информацией.

Основные цели информационной безопасности:

- защита государственных, национальных интересов;
- обеспечение каждого члена общества достоверной и максимально полной информацией;
- защита конфиденциальной информации от доступа посторонних лиц;
- правовая защита отдельных лиц и общества в целом при получении, распространении и использовании информации.

Таким образом, информационная безопасность подразумевает действия по предотвращению возможного повреждения или уничтожения информации, а также несанкционированного доступа к ней, и обеспечение беспрепятственного доступа к информации со стороны легитимных пользователей.

Отдельно может рассматриваться информационная безопасность какой-либо организации, например школы, как состояние защищённости её информационной среды. Может также рассматриваться информационная безопасность государства как состояние сохранности его информационных ресурсов и защищённости законных прав личности и общества в информационной сфере.

Меры по обеспечению информационной безопасности можно разделить на следующие категории:

- технические — аппаратные и программные средства и технологии защиты от вредоносных программ, внешних сетевых атак и пр. (в том числе различные антивирусные программы);
- правовые — совокупность нормативных и правовых актов, регулирующих вопросы защиты информации;
- административные — разрешения и запреты, регламентирующие работу пользователей информационных ресурсов.

Основным законом, касающимся вопросов использования и защиты информации, в нашей стране является **Федеральный закон № 149-ФЗ «Об информации, информационных технологиях и о защите информации»**¹⁾, принятый 27 июля 2006 г. Согласно ему, любые действия, приводящие к повреждению или уничтожению информации, к её намеренному искажению, а также стремление получить неправомерный доступ к чужой конфиденциальной информации, являются преступлениями. В России соответствующие меры наказания за них определены в Уголовном кодексе Российской Федерации (УК РФ) № 63-ФЗ, вступившем в действие 13.06.1996 г. Преступления, связанные с компьютерной информацией, рассмотрены в нём в главе 28 (статья 272 «Неправомерный доступ к компьютерной информации», статья 273 «Создание, использование и распространение вредоносных программ для ЭВМ», статья 274 «Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети»).

Так, неправомерный доступ к охраняемой законом компьютерной информации (на электронном носителе, в электронно-вычислительной машине или компьютерной сети), результатом которого стало уничтожение, блокирование, модификация или копирование информации, нарушение работы ЭВМ или компьютерной сети, наказывается штрафом до 200 тысяч рублей или в размере заработной платы за период до 18 месяцев, исправительными работами на срок от шести месяцев до одного года, либо лишением свободы на срок до двух лет. А то же действие, совершенное группой лиц по предварительному сговору или организованной группой, наказывается штрафом от 100 до 300 тысяч рублей или в размере заработной платы за период от одного года до двух лет, исправительными работами на срок от одного года до двух лет, либо арестом на срок от трёх до шести месяцев, либо лишением свободы на срок до пяти лет.

¹⁾ Ред. от 13.07.2015; с изменениями и дополнениями вступил в силу с 10.01.2016.

Создание программ для ЭВМ (или внесение изменений в существующие программы), заведомо приводящих к несанкционированному уничтожению, блокированию, модификации либо копированию информации, нарушению работы ЭВМ или компьютерной сети, а также использование или распространение таких программ наказывается лишением свободы на срок до трёх лет со штрафом в размере до 200 тысяч рублей или в размере заработной платы за период до 18 месяцев. А те же действия, повлёкшие тяжкие последствия, наказываются лишением свободы на срок от трёх до семи лет.

Нарушение правил эксплуатации ЭВМ или компьютерной сети, приведшее к уничтожению, блокированию или модификации охраняемой законом информации ЭВМ, если это причинило существенный вред, наказывается лишением права занимать определённые должности на срок до пяти лет, либо обязательными работами на срок от 180 до 240 часов, либо ограничением свободы на срок до двух лет. А то же действие, повлёкшее тяжкие последствия, наказывается лишением свободы на срок до четырёх лет.

Таким образом, создание вредоносных программ карается штрафами (чаще всего с конфискацией соответствующего компьютерного оборудования), а в отдельных случаях — реальными тюремными сроками.

Ещё один важный законодательный акт — **Федеральный закон № 152-ФЗ «О персональных данных»**, вступивший в действие 27 июля 2006 г. Он нацелен на защиту персональной, т. е. личной, информации каждого человека. Персональные данные — это любая информация, относящаяся к определённому физическому лицу, в том числе его фамилия, имя, отчество, дата и место рождения, адрес, семейное, социальное, имущественное положение, образование, профессия, доходы и другая информация. Обработка таких персональных данных может осуществляться только с согласия владельца этих персональных данных, за исключением особо оговорённых в законе случаев.

Технические меры информационной безопасности заключаются в использовании специальных аппаратных и программных средств для защиты информации, компьютерных систем и сетей, а также для обнаружения и уничтожения вредоносных программ.

Среди таких средств чаще всего используются:

- **антивирусные программы (антивирусы)** — в настоящее время являются комплексными и нацелены на борьбу не только с компьютерными вирусами, но и со всеми разновидностями вредоносных программ;

- средства защиты компьютера от внешних информационных атак через компьютерные сети — **брандмауэры**, или **файрволы**.

Административные меры информационной безопасности заключаются в реализации информационного обмена в конкретной организации (предприятии, фирме, учреждении). Фактически речь идёт о дисциплине работы сотрудников с информацией и о регламентировании прав доступа к ней. В число таких мер могут входить:

- запрет для сотрудников или учащихся использовать свои информационные носители (например, флешки) на рабочих компьютерах — для предотвращения заражения вредоносными программами и кражи информации;
- распределение прав доступа к тем или иным компьютерам в локальной сети, а также к конкретным папкам или файлам в сети и на отдельном компьютере;
- авторизация (возможность входа в компьютер, входа в сеть, доступа к тем или иным программам или документам по назначенному для каждого пользователя логину и паролю) с предоставлением конкретного перечня прав и возможностей работы с информационными объектами и т. д.

Электронная подпись — это особое свойство некоторого электронного документа (например, пересылаемого по компьютерной сети Интернет), которое подтверждает отсутствие искажения информации в документе и принадлежность этого документа его владельцу. Электронная подпись — это разновидность криптографического преобразования информации.

Для формирования электронной подписи используются два ключа шифрования. Первый, закрытый (секретный), ключ позволяет создавать электронную подпись и применять её к электронным документам. Другой, открытый (общедоступный), ключ однозначно взаимосвязан с закрытым ключом и позволяет проверить подлинность электронной подписи.

Оба этих ключа владелец электронной подписи получает вместе с сертификатом электронной подписи — особым документом, который подтверждает принадлежность открытого ключа электронной подписи его владельцу. Такие сертификаты выдают *удовверяющие центры* или их представители.

Использование электронных подписей регламентировано **федеральным законом № 63-ФЗ «Об электронной подписи»**. Согласно ему, возможно три разновидности электронной подписи: простая, усиленная неквалифицированная и усиленная квалифицированная. Электронный документ, подписанный простой или усилен-

ной неквалифицированной подписью, считается равнозначным бумажному документу с собственноручной подписью его автора (но при этом между участниками обмена такими документами должно быть заключено соответствующее соглашение). Усиленная же квалифицированная подпись на электронном документе — это аналог подписи и печати на бумажном документе.

В настоящее время электронная подпись постепенно начинает внедряться в электронный документооборот предприятий и фирм (как внутренний, так и внешний), применяться для отчётных документов (например, передаваемых в налоговые органы), при получении электронных государственных услуг на соответствующем портале, для подтверждения документов, предоставляемых в качестве доказательства в суде и т. п.

Вопросы и задания



1. Какими могут быть информационные угрозы? К каким реальным последствиям они могут приводить?
2. В чём заключается информационная война?
3. Что понимается под хакерской атакой? Для чего она предпринимается?
4. Что такое вредоносная программа?
5. Что понимается под информационной безопасностью? Какие меры составляют информационную безопасность?
6. Что понимается под правовыми и административными мерами информационной безопасности?
7. Что такое электронная подпись? На чем она основана и в каких случаях может быть использована?

Практическая работа 4.2



Законы об информационной безопасности и электронной подписи

Задание. Ознакомиться с законодательной базой использования и защиты информации. Ознакомиться с законодательной базой использования электронной подписи.

Варианты выполнения работы:

- в операционной системе Windows или в операционной системе Linux;
- просмотр веб-страниц в различных браузерах (Mozilla Firefox, Opera, Chrome или др.).



Ознакомление с текстом Федерального закона № 149-ФЗ «Об информации, информационных технологиях и о защите информации»



Ознакомимся с правовыми нормами защиты информации.

1. Запустить браузер в операционной системе Windows или в операционной системе Linux и в поле *Адрес:* ввести http://www.consultant.ru/document/cons_doc_LAW_61798 (доменное имя и путь к сайту «КонсультантПлюс»).
2. Самостоятельно ознакомиться со статьями 2–10.2, описывающими основные понятия, связанные с информацией, информационными технологиями и защитой информации.
3. Самостоятельно ознакомиться со статьёй 16 «Защита информации».
4. Самостоятельно ознакомиться со статьёй 17 «Ответственность за правонарушения в сфере информации, информационных технологий и защиты информации».



Ознакомление с текстом Уголовного кодекса Российской Федерации (УК РФ)



Ознакомимся с наказаниями за преступления, связанные с нарушением информационной безопасности.

1. Запустить браузер в операционной системе Windows или в операционной системе Linux и в поле *Адрес:* ввести http://www.consultant.ru/document/cons_doc_LAW_10699 (доменное имя и путь к сайту «КонсультантПлюс»).
2. Самостоятельно ознакомиться со статьями главы 28 «Преступления в сфере компьютерной информации».



Ознакомление с текстом Федерального закона № 152-ФЗ «О персональных данных»



Ознакомимся с основными понятиями и нормативным регулированием использования персональной информации.

1. Запустить браузер в операционной системе Windows или в операционной системе Linux и в поле *Адрес:* ввести http://www.consultant.ru/document/cons_doc_LAW_61801 (доменное имя и путь к сайту «КонсультантПлюс»).
2. Самостоятельно ознакомиться с текстом глав 1 и 2 Федерального закона о персональных данных.



Ознакомление с текстом Федерального закона № 63-ФЗ «Об электронной подписи»



Ознакомимся с основными понятиями и нормативным регулированием использования электронной подписи.

1. Запустить браузер в операционной системе Windows или в операционной системе Linux и в поле *Адрес:* ввести **http://www.consultant.ru/document/cons_doc_LAW_112701** (доменное имя и путь к сайту «КонсультантПлюс»).
2. Самостоятельно ознакомиться с текстом статей 2–6 и статьи 9 Федерального закона об электронной подписи.

ОГЛАВЛЕНИЕ

Рекомендации по использованию учебника	3
--	---

Глава 1. Компьютер как средство автоматизации информационных процессов.....	5
--	----------

1.1. История развития вычислительной техники.....	5
1.1.1. Развитие электронной вычислительной техники.....	7
Практическая работа 1.1. Виртуальные компьютерные музеи	11
1.2. Архитектура персонального компьютера.....	13
Практическая работа 1.2. Сведения об архитектуре компьютера.....	24
1.3. Операционные системы.....	26
1.3.1. Основные характеристики операционных систем ...	26
Практическая работа 1.3. Сведения о логических разделах дисков.....	32
1.3.2. Операционная система Windows.....	34
Практическая работа 1.4. Значки и ярлыки на Рабочем столе.....	38
1.3.3. Операционная система Linux	40
Практическая работа 1.5. Настройка графического интерфейса операционной системы Linux.....	44
Практическая работа 1.6. Установка пакетов в операционной системе Linux	45
1.3.4. Мобильные операционные системы.....	47
1.3.5. Особенности программного обеспечения мобильных систем	49
1.4. Защита от несанкционированного доступа к информации	54
1.4.1. Защита с использованием паролей	54
1.4.2. Биометрические системы защиты	57
Практическая работа 1.7. Биометрическая защита: идентификация по характеристикам речи	59
1.5. Физическая защита данных на дисках.....	60

1.6. Защита от вредоносных программ	62
1.6.1. Вредоносные и антивирусные программы	62
1.6.2. Компьютерные вирусы и защита от них	66
Практическая работа 1.8. Защита от компьютерных вирусов	70
1.6.3. Сетевые черви и защита от них	81
Практическая работа 1.9. Защита от сетевых червей.....	84
1.6.4. Троянские программы и защита от них	90
Практическая работа 1.10. Самозащита компьютера с помощью программы Kaspersky Antivirus.....	93
1.6.5. Блокираторы и другие программы-вымогатели.....	95
1.6.6. Хакерские утилиты и защита от них.....	97
Практическая работа 1.11. Защита от хакерских атак ...	99
ЭОР к главе 1 на сайте ФЦИОР (http://fcior.edu.ru)....	102
Глава 2. Моделирование и формализация	103
2.1. Моделирование как метод познания	103
2.2. Системный подход в моделировании	105
2.3. Формы представления моделей.....	107
2.4. Формализация	109
2.5. Основные этапы разработки и исследования моделей на компьютере	110
2.6. Инструменты программирования для разработки и исследования моделей.....	112
2.6.1. Понятие массивов.....	112
Практическая работа 2.1. Исследование процесса изменения температуры средствами программирования с использованием заполнения массива.....	113
2.6.2. Другие составные типы данных	117
2.6.3. Использование массивов данных в разработке моделей.....	128
Практическая работа 2.2. Проведение исследования на упорядочение и поиск экстремальных значений потока информации о температуре воздуха	134
2.6.4. Использование элементов графики в разработке моделей.....	142
Практическая работа 2.3. Проектирование простого графического редактора	150

2.6.5. Исследование математических моделей	166
Практическая работа 2.4. Графическое решение уравнения	168
Домашний проект. Графическое решение уравнения	175
2.6.6. Оптимизационное моделирование в экономике	176
Практическая работа 2.5. Построение и исследование оптимизационной модели	178
Домашний проект. Оптимизация раскроя	182
2.7. Исследование интерактивных компьютерных моделей ...	182
2.7.1. Исследование физических и астрономических моделей.	182
Практическая работа 2.6. Построение и исследование модели «Бросание мячика в стенку»	185
Домашний проект. Построение и исследование модели «Бросание мячика в стенку»	197
2.7.2. Исследование химических моделей	197
Практическая работа 2.7. Построение и исследование модели «Распознавание волокон»	199
2.7.3. Исследование биологических моделей	203
Практическая работа 2.8. Построение и исследование модели «Популяция»	204
ЭОР к главе 2 на сайте ФЦИОР (http://fcior.edu.ru)	207

Глава 3. Базы данных. Системы управления базами данных (СУБД)

3.1. Базы данных	208
3.2. Система управления базами данных	211
3.2.1. Основные объекты СУБД: таблицы, формы, запросы, отчёты	211
Практическая работа 3.1. Создание базы данных	212
3.2.2. Использование формы для просмотра и редактирования записей в базе данных	216
Практическая работа 3.2. Создание формы в базе данных	217
3.2.3. Поиск записей в базе данных с помощью фильтров и запросов	222
Практическая работа 3.3. Поиск записей в базе данных с помощью фильтров и запросов	228
3.2.4. Сортировка записей в табличной базе данных	232

Практическая работа 3.4. Сортировка записей в табличной базе данных.....	232
3.2.5. Печать данных с помощью отчётов.....	233
Практическая работа 3.5. Создание отчёта в базе данных.....	234
3.3. Иерархическая модель данных.....	235
3.4. Сетевая модель данных.....	235
Практическая работа 3.6. Создание генеалогического древа семьи.....	236
ЭОР к главе 3 на сайте ФЦИОР (http://fcior.edu.ru)....	238

Глава 4. Социальная информатика..... 239

4.1. Информационное общество.....	239
4.1.1. Информационная культура.....	245
4.2. Правовые основы информационной среды.....	246
4.2.1. Лицензирование программного обеспечения.....	249
Практическая работа 4.1. Законы об охране авторских прав.....	252
4.3. Социальные сервисы и сети.....	253
4.4. Информационная безопасность.....	260
Практическая работа 4.2. Законы об информационной безопасности и электронной подписи.....	265

Учебное издание

Угринович Николай Дмитриевич

ИНФОРМАТИКА

11 класс

Базовый уровень

Ведущий редактор *О. А. Полежаева*

Ведущие методисты *И. Ю. Хлобыстова, И. Б. Животова*

Художник *Н. А. Новак*

Технический редактор *Е. В. Денюкова*

Корректор *Е. Н. Клитина*

Компьютерная верстка: *Л. В. Катуркина*

Подписано в печать 01.03.17. Формат 70х100/16. Усл. печ. л. 22,1.

Тираж 3000 экз. Заказ № м4357.

ООО «ВИНОМ. 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>