

Г. А. Заборовский А. Е. Пупцев

ИНФОРМАТИКА

Учебное пособие для 11 класса
общеобразовательных учреждений
с русским языком обучения

Допущено
Министерством образования
Республики Беларусь

Минск
«Народная асвета»
2010

Образовательный портал www.adu.by/ Национальный институт образования

УДК 004(075.3=161.1)

ББК 32.81я721

3-41

Авторы:

Г. А. Заборовский (главы 1, 2), А. Е. Пупцев (главы 1, 3, 4)

Рецензенты:

кафедра информатики и компьютерного моделирования
Гродненского государственного университета имени Янки Купалы
(канд. пед. наук, доцент *Н. П. Макарова*);
старший преподаватель кафедры естественнонаучных дисциплин
и информационных технологий Минского областного института
развития образования *А. А. Буславский*

Заборовский, Г. А.

3-41 Информатика : учеб. пособие для 11-го кл. общеобразоват. учреждений с рус. яз. обучения / Г. А. Заборовский, А. Е. Пупцев. — Минск : Нар. асвета, 2010. — 150 с. : ил.

ISBN 978-985-03-1370-6.

УДК 004(075.3=161.1)

ББК 32.81я721

ISBN 978-985-03-1370-6

© Заборовский Г. А., Пупцев А. Е., 2010

© Оформление. УП «Народная асвета»,
2010

Образовательный портал www.adu.by/ Национальный институт образования

От авторов

Уважаемые школьники!

В одиннадцатом классе Вы продолжите углублять свои знания в области информационных технологий, основ алгоритмизации и программирования.

В первой главе пособия Вы познакомитесь с инструментами и методами веб-конструирования, изучите основы языка разметки гипертекстовых документов HTML, познакомитесь с особенностями подготовки графики для сети Интернет, научитесь создавать веб-сайты.

Во второй главе будет продолжено изучение основ алгоритмизации и программирования. Полученные ранее знания и умения Вы научитесь применять для решения практических задач из различных предметных областей.

Третья глава посвящена обработке информации в системах управления базами данных.

Четвертая, заключительная, глава завершает знакомство с информационными системами и технологиями.

Вопросы, отмеченные знаком  , предназначены для закрепления изученного материала. Дополнительный материал для любознательных отмечен знаком .

После параграфов предлагаются упражнения, которые позволяют Вам закрепить свои знания и практические умения работы на компьютере.

Желаем успехов в изучении информатики и информационных технологий. Хорошие знания помогут Вам выбрать профессию и приносить пользу Родине.

ОСНОВЫ ВЕБ-КОНСТРУИРОВАНИЯ

§ 1. Представление о веб-конструировании

1.1. Инструменты и методы разработки веб-сайтов

Большинство информационных ресурсов сети Интернет представлено в виде веб-страниц, которые объединяются в веб-сайты. Благодаря развитию телекоммуникационных технологий размещенная на веб-страницах информация доступна множеству людей. В отличие от информации на бумажных носителях, например книг, она может оперативно изменяться и практически мгновенно доставляться в любое место по запросу пользователя.

Для создания веб-страниц используется язык разметки гипертекстовых документов HTML (*HyperText Markup Language*).

В качестве инструментов веб-конструирования могут использоваться офисные программы, например входящие в состав Microsoft Office (Word, PowerPoint), которые не являются специальными средствами разработки веб-страниц. Лучших результатов можно достичь, используя специальные программы — веб-редакторы, например Microsoft FrontPage.

Выделяют две основные группы методов и соответствующих инструментов разработки веб-сайтов: *визуальные* и *ручные* (программные).

Визуальные методы позволяют производить все работы по созданию веб-страниц с высокой степенью автоматизации и не требуют знания языка разметки HTML. Они уменьшают трудоемкость и сроки разработки сайта. Суть визуальных методов отражена в принципе WYSIWYG (от англ. *What you see is what you get* — Что видишь, то и получаешь).

Разработано немало специальных инструментов — редакторов визуального конструирования. Наиболее известными являются Microsoft FrontPage, Adobe (Macromedia) Dreamweaver, NamoWebEditor и др. С помощью таких редакторов веб-страницы создаются (рисуются) в интерактивном режиме, при этом автоматически генерируется соответствующий **HTML-код**, который представляет собой набор команд языка разметки HTML. Отметим, что упомянутые редакторы позволяют выполнять все работы по созданию сайта без непосредственного подключения к сети Интернет, или, как говорят, в режиме *оффлайн* (*offline*). Затем созданный веб-сайт публикуется в сети Интернет, т. е. размещается на веб-сервере.



В последнее время для конструирования и сопровождения веб-сайтов используются системы управления их содержимым (контентом) — CMS (*Content Management System*), которые предоставляются специальными платными или бесплатными службами. Системы CMS представляют собой своего рода конструкторы. Они позволяют создавать сайт в режиме непосредственного подключения к сети, или *онлайн* (*online*), и сопровождать его в дальнейшем. При создании структуры сайта и разработке навигации по нему в системе CMS также не требуется знаний языка HTML. Из бесплатных систем CMS наиболее популярны Joomla! (<http://joomla.ru/>) и Drupal (<http://www.drupal.ru>).

Конечно, трудно создать хороший сайт, не зная основ языка разметки HTML. Для работы с HTML-кодом могут использоваться специальные инструменты разработки, позволяющие набирать команды HTML **вручную**, например редакторы HotDog, Adobe HomeSite и др. Эти инструменты облегчают ввод и редактирование кода. Однако, вводить основные команды (**теги**) языка HTML можно даже в простейшем текстовом редакторе **Блокнот**, а просматривать результаты работы можно с помощью браузера.

Важную роль в выборе инструментов и методов веб-конструирования играет статичность или динамичность создаваемых страниц, а также наличие интерактивных элементов. *Статические* страницы отображаются браузером пользователя в том виде, в каком были созданы и размещены на веб-сервере. *Динамические* страницы генерируются по запросу пользователя — информация на них загружается серверными программами из баз данных. Такие базы данных обычно содержат информацию, которая требует постоянного обновления. Например, интернет-магазин пополняется сведениями о новых товарах и изменяющихся ценах.

Интерактивные (т. е. управляемые пользователем) элементы веб-страниц используются для ввода пароля, выбора товара, оценки его качества, ввода ответа при тестировании или голосовании, для формирования запроса на поиск информации в удаленной базе данных и т. п.



Кроме языка разметки гипертекстовых документов HTML, при создании веб-сайтов используют и специальные языки веб-программирования. Широкое применение получил язык сценариев Java Script. Написанные на нем конструкции, или *скрипты*, вставляются непосредственно на веб-страницы и интерпретируются браузером. Они используются для создания отдельных, как правило, интерактивных элементов веб-страниц, например динамических меню, часов, календарей, форм запросов, счетчиков посещений страниц, систем голосования и т. п. Для серверного программирования наиболее

часто используют языки PHP (от англ. *Hypertext Preprocessor* — препроцессор гипертекста), PERL (от англ. *Practical Extraction and Report Language* — практический язык для извлечения данных и составления отчетов).



1. В чем разница между визуальными и ручными методами веб-конструирования?
2. Какие инструменты могут использоваться при создании веб-сайтов?
3. Какие страницы называют статическими? Динамическими?

1.2. Проектирование сайта

Выделяют следующие основные этапы разработки веб-сайтов:

- определение тематики сайта, его целей и задач;
- проектирование структуры сайта, определение разделов и связей между страницами;
- разработка дизайна сайта, т. е. стиля оформления страниц;
- подготовка материалов (текстов и графики) для размещения на веб-страницах;
- конструирование страниц сайта (создание HTML-кода);
- размещение в сети (публикация) и тестирование сайта.

Рассмотрим на примере, как спроектировать веб-сайт кинотеатра.

Определим основную цель сайта: привлечение зрителей, и задачи: информирование о репертуаре кинотеатра, реклама фильмов.

Разработку проекта начнем с построения информационной модели сайта. Пусть для простоты наш первый сайт будет состоять из четырех веб-страниц. Структуру этого сайта для наглядности изобразим в виде двухуровневой схемы (рис. 1.1).

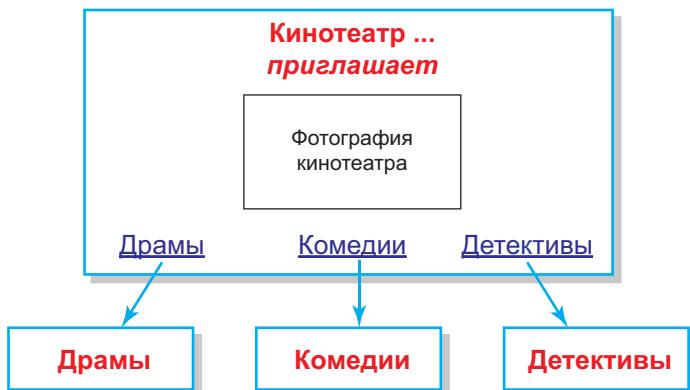


Рис. 1.1

На первом (верхнем) уровне схемы изобразим первую (главную) веб-страницу. На ней будут размещены: общая информация о кинотеатре (например, фотография, адрес) и гиперссылки.

На втором уровне схемы изобразим веб-страницы, посвященные фильмам определенных жанров, например «Драмы», «Комедии», «Детективы». Они имеют схожую структуру и будут отличаться лишь содержанием (рис. 1.2).

Структура сайта в дальнейшем может быть усложнена добавлением страниц последующих уровней, например страниц, посвященных актерам, режиссерам, сценаристам, композиторам и т. п.

Теперь спроектируем дизайн сайта. Основными структурными элементами оформления веб-страниц являются текстовые блоки (основной текст, заголовки, списки, текстовые гиперссылки) и графические объекты: изображения (рисунки, фотографии, анимации), «обои» (фоновые рисунки), изображения-гиперссылки, разделительные линии. Для размещения элементов в заданных местах страниц будем использовать таблицы.

Сочетание свойств элементов веб-страницы определяет стиль ее оформления. **Стиль текста** задается совокупностью параметров формата: шрифт, размер, начертание; отступы, выравнивание, межсимвольный и межстрочный интервалы и др. **Стиль графических элементов** задается совокупностью параметров формы: размер, цвет, фактура материала, а также разнообразных эффектов: тень, блеск, прозрачность и т. п.

Для стилистически грамотного оформления документов удобно использовать готовые шаблоны — темы. Напомним, что **темой** называют специально разработанный набор элементов оформления и цветовых схем документа. С помощью темы можно задать стиль оформления как отдельной страницы, так и всего сайта. Темы позволяют задавать стили основного текста, заголовков, гиперссылок, списков; цвет фона; «обои»; цвет и толщину границ таблицы и т. п. Темы предлагаются во всех офисных программах, однако наибольшую пользу приносит их использование при создании презентаций и веб-сайтов.

Все страницы одного уровня будем оформлять в едином стиле.

Название фильма	
Фрагмент фильма (фотография)	Жанр Режиссер Актеры Студия, год
Краткое содержание	

Рис. 1.2

Непосредственное конструирование страниц сайта начнем с подготовки всех его элементов (текстов, рисунков). Затем эти элементы и (или) ссылки на них будем вставлять в готовые шаблоны.

Файлы всех страниц веб-сайта будем сохранять в одной папке, например KINO. Это упростит размещение сайта на веб-сервере. В случае большого количества страниц или изображений их следует сохранять во вложенных папках, например KINO\FOTO\. Файлу главной веб-страницы обычно дают имя index или main с расширением .htm.

Только после размещения файлов в папках можно создавать гиперссылки, с помощью которых выполняется навигация по сайту — переходы со страницы на страницу. В нашем примере удобно сначала разработать страницы нижнего уровня, а затем оформить главную страницу и создать гиперссылки для вызова страниц нижнего уровня. Затем следует проверить работу ссылок в режиме оффлайн. Лишь после тщательной проверки и исправления ошибок созданный веб-сайт можно опубликовать, т. е. разместить на веб-сервере.



1. Какие этапы можно выделить при разработке веб-сайта?
2. Что понимают под стилем оформления веб-страницы?

Упражнение

Разработайте проект веб-сайта по одной из тем: Моя семья. Моя школа. Мои друзья. Мои любимые занятия. Моя Родина — Беларусь. Природа родного края. Интересные профессии.

§ 2. Использование офисных приложений для создания веб-страниц

2.1. Создание веб-страниц в редакторе MS Word

Любой документ MS Office можно сохранить в виде веб-страницы. Для этого достаточно выполнить команду **Файл → Сохранить как веб-страницу**. Однако далеко не всякий подготовленный для печати на бумаге документ будет хорошо выглядеть в браузере. Наилучшего результата можно достичь, если с самого начала готовить документ для размещения в сети Интернет.

Использование текстового редактора MS Word является одним из самых простых способов разработки веб-страниц. Рассмотрим на примере, как это делается. Начнем со страниц нижнего уровня, а затем оформим главную страницу и создадим гиперссылки.

The screenshot shows a web browser window with the title bar "Анастасия Слуцкая". The address bar displays "file:///D:/!Work/KINO/drama.htm". The main content area features a painting of a woman in a crown and dark robes, identified as Anastasia Slutskaya. To the right of the image are four text boxes containing film details:

- Жанр:** историческая драма
- Режиссер:** Юрий Елхов
- Актеры:** Светлана Зеленковская, Геннадий Давыдько, Анатолий Кот, Фуркат Файзиев, Сергей Глушко...
- "Беларусьфильм", 2003 г.**

Below the image and text boxes, there is a descriptive paragraph in Russian:

Начало XVI века.
Богатые земли в центре Европы на пересечении торговых путей становятся желанной добычей для соседей. Полчища татар нападают с юга. Быстрыми набегами крымчаки достигают самых отдаленных белорусских городов. Падают крепости, опустошаются земли. Бесстрашная княгиня Анастасия Слуцкая встает на защиту белорусской земли от бесчисленных полчищ, несущих опустошение и смерть...

Рис. 1.3

Пример 1. Создать веб-страницу фильма «Анастасия Слуцкая» (рис. 1.3).

Прежде всего с помощью команды **Файл** → **Создать** выберем пункт **Веб-страница** (этот пункт можно выбрать и в области задач **Создание документа**). Затем, выполнив команду **Формат** → **Тема**, выберем для оформления страницы тему **Перетекание** (рис. 1.4).

Для размещения на странице фотографии и текстов используем таблицу из двух строк и двух столбцов. Объединим ячейки нижней строки. Вставим подготовленные заранее тексты и изображение из файлов. Оформим их в соответствии с рисунком 1.3.

Созданный документ сохраним как веб-страницу в папке KINO под именем drama.htm. Выберем тип сохраняемого файла **Веб-страница (*.htm; *.html)**. При этом все используемые в документе изображения будут помещены в отдельную папку с именем веб-страницы и расширением .files (в нашем примере — drama.files).

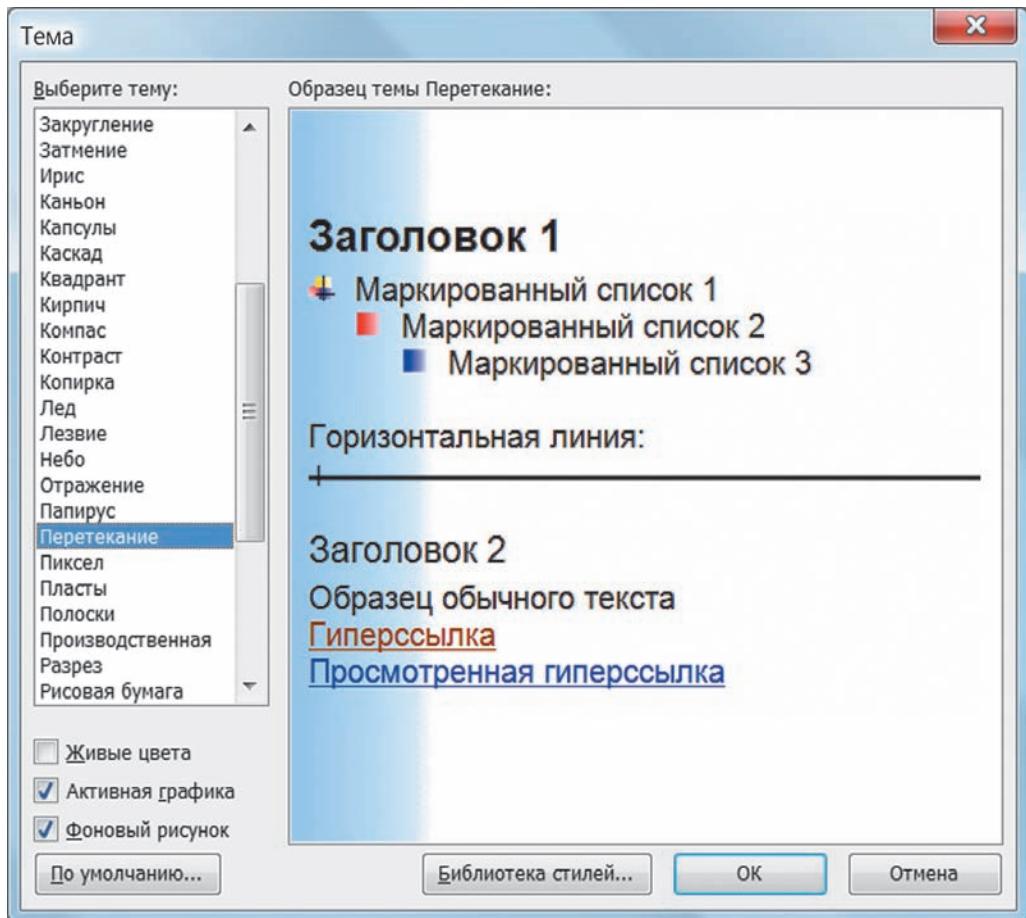


Рис. 1.4

Аналогичным способом создаются страницы других фильмов и жанров, например «Комедии» и «Детективы». Поскольку они имеют сходную структуру и отличаются лишь содержанием, то новые страницы можно получать редактированием уже созданных, заменяя изображения и тексты и сохраняя их под новыми именами.

Создадим веб-страницу фильма «В августе 44-го...» (рис. 1.5) на основе уже созданной нами страницы «Анастасия Слуцкая». Сохраним ее под именем avgust44.htm.

Пример 2. Создать главную страницу сайта кинотеатра «Беларусь» (рис. 1.6), содержащую гиперссылки на страницы фильмов.



В августе 44-го...

Жанр: военный детектив

Режиссер: Михаил Пташук

Актеры: Владислав Галкин, Евгений Миронов, Beata Тышкевич, Александр Феклистов, Ярослав Бойко, Юрий Колокольников, Алексей Петренко...

"Беларусьфильм", 2000 г.

События происходят в августе 1944 года.
На территории уже освобожденной Белоруссии действуют агенты противника.
Шпионская группа, регулярно передающая с территории республики шифрованные радиограммы, остается неуловимой.
Контрразведчикам из СМЕРШа во главе с капитаном Алехиным поручено разыскать диверсионную группу...

Рис. 1.5



Кинотеатр «Беларусь»

Ретроспектива белорусских фильмов

Драмы Комедии Детективы

Рис. 1.6

Для главной страницы сайта используем тему **Перетекание**.

Важный этап разработки веб-страницы кинотеатра — создание гиперссылок, открывающих страницы фильмов. Для размещения гиперссылок создадим таблицу. Для создания текстовых гиперссылок выполним следующие действия:

- Выделим текст гиперссылки, например слово «Драмы».
- С помощью команды **Вставка** → **Гиперссылка** или кнопки  на панели инструментов выберем пункт **Связать с файлом, веб-страницей**, выделим имя файла drama.htm и подтвердим выбор нажатием кнопки ОК.

Гиперссылки в виде изображений создаются аналогично.

Завершив создание главной страницы, не забудем сохранить ее под именем main.htm в той же папке, в которой ранее сохраняли страницы фильмов.

По окончании работы просмотрим созданные страницы в браузере. Проверим работу ссылок, переходя по ним на страницы фильмов и возвращаясь на главную страницу с помощью кнопки **Назад** браузера.



Повысить привлекательность веб-страницы можно, разместив на ней динамические и интерактивные элементы (анимации, аудио- и видеофрагменты, формы опросов). При этом следует помнить, что анимации часто отвлекают пользователей от восприятия текстовой информации, а большие размеры аудиовideoфайлов существенно замедляют загрузку страниц.



Пример 3. Разместить на главной странице сайта кинотеатра «Беларусь» бегущую строку.

С помощью команды **Вид** → **Панели инструментов** активизируем панель **Веб-компоненты** (рис. 1.7). Нажмем на этой панели кнопку .



Рис. 1.7

В открывшемся окне **Бегущая строка** (рис. 1.8) наберем требуемый текст, например **«Предлагает!»**. Сделаем нужные установки: выберем направление движения **Справа налево** и нажмем кнопку ОК.

Сохраним измененную страницу.

Редактор MS Word позволяет подключать к веб-странице видео- и звуковые файлы. Для этого необходимо нажатием значка **Звук**  или **Фильм**  на панели **Веб-компоненты** открыть соответствующее диалоговое окно, нажать кнопку **Обзор**, найти требуемый видео- или звуковой файл, установить параметры

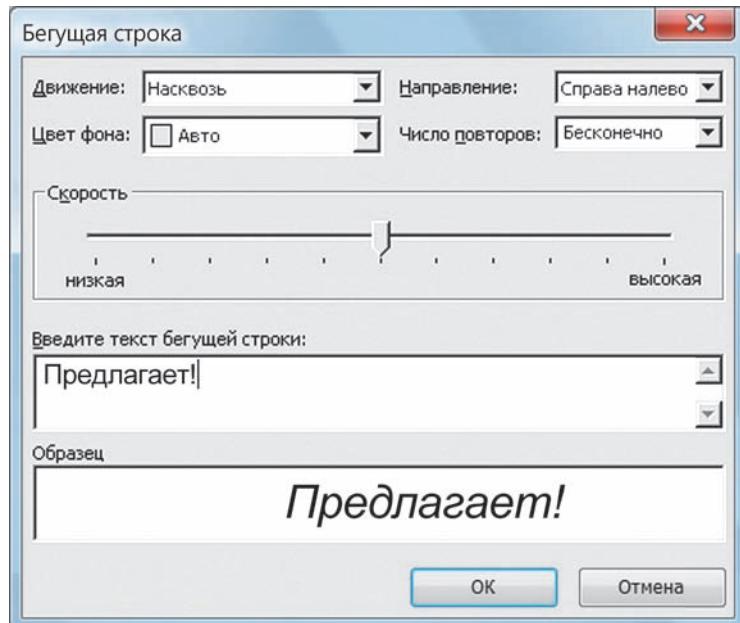


Рис. 1.8

ры проигрывания, например количество повторов, и подтвердить выбор нажатием кнопки ОК.

- ?**
1. Как можно сохранить документ MS Word в виде веб-страницы?
 2. С помощью каких действий можно создать гиперссылку?

Упражнение

Создайте фрагмент сайта по одной из тем: Моя Родина — Беларусь. Моя школа. Мои друзья. Мои любимые занятия. Моя семья (пример главной страницы представлен на рисунке). В качестве гиперссылок используйте картинки из коллекции клипов.



2.2. Сохранение презентаций PowerPoint в виде веб-страниц

Презентация PowerPoint, как правило, содержит несколько слайдов, каждый из которых может быть сохранен как отдельная веб-страница. Всю презентацию можно сохранить как сайт, структура которого будет соответствовать структуре презентации.

Пример 1. Создать фрагмент сайта на основе презентации «Функциональные блоки компьютера».

Откроем в редакторе PowerPoint презентацию «Функциональные блоки компьютера» (файл comp.ppt). Просмотрим ее структуру. Она состоит из 5 слайдов, которые содержат тексты и изображения (рис. 1.9).

Для сохранения презентации в виде веб-страницы в меню **Файл** выберем пункт **Сохранить как веб-страницу**. В появившемся окне **Сохранение документа** выберем тип сохраняемого файла **Веб-страница (*.htm, *.html)**, введем имя

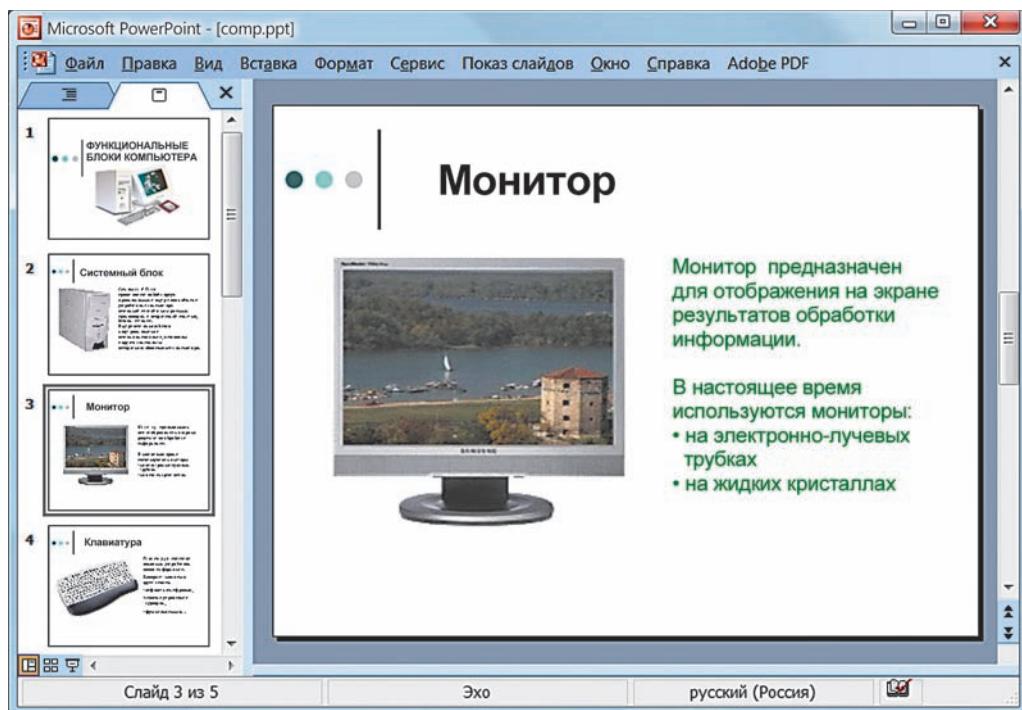


Рис. 1.9

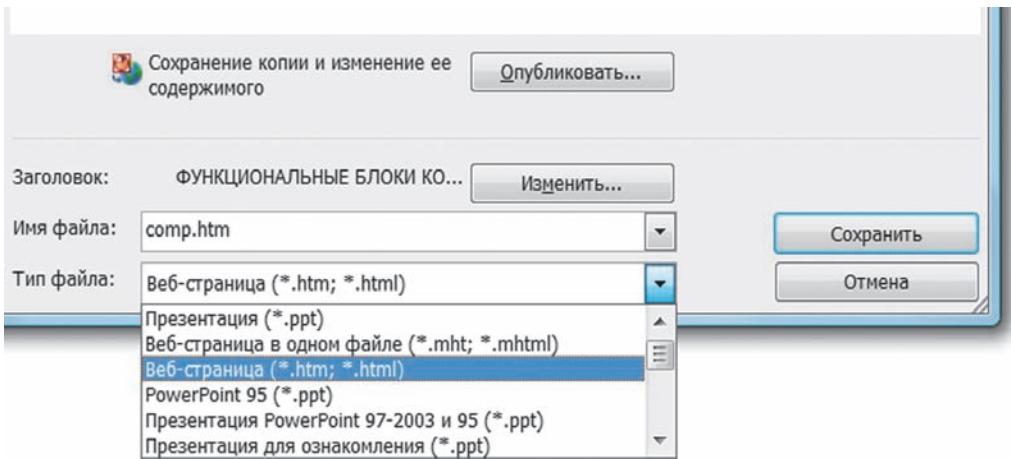


Рис. 1.10

файла или оставим прежнее (в нашем примере — comp.htm) и нажмем кнопку **Сохранить** (рис. 1.10).

При использовании предлагаемых по умолчанию настроек все слайды презентации сохраняются в папке compr.files в виде отдельных страниц, которые можно будет пролистывать в правой части окна браузера с помощью гиперссылок. Эти гиперссылки располагаются в левой части сохраненной страницы comp.htm (по умолчанию белого цвета на черном фоне).

Изменим эти настройки. С этой целью нажатием кнопки **Опубликовать** вызовем окно **Публикация веб-страницы** (рис. 1.11) и настроим параметры веб-документа. В этом окне можно выбрать публикуемые объекты (**Полная презентация** или **Слайды**), поддержку браузеров, а также изменить заголовок веб-страницы. По умолчанию как объект для публикации предлагается **Полная презентация**. В этом случае в виде веб-страниц будут сохранены все слайды.

Цветовые схемы, эффекты анимации и другие параметры настраиваются в окне **Параметры веб-документа** (рис. 1.12), которое вызывается кнопкой **Веб-параметры**.

На вкладке **Общие** выберем цветовую схему **Цвета презентации (выделение)**. В этом случае фон веб-страниц будет соответствовать фону слайдов презентации.

Установленный по умолчанию флажок **Добавить панель смены слайдов** следует снимать лишь в том случае, когда на слайдах предварительно установлены гиперссылки, указывающие переходы между слайдами.

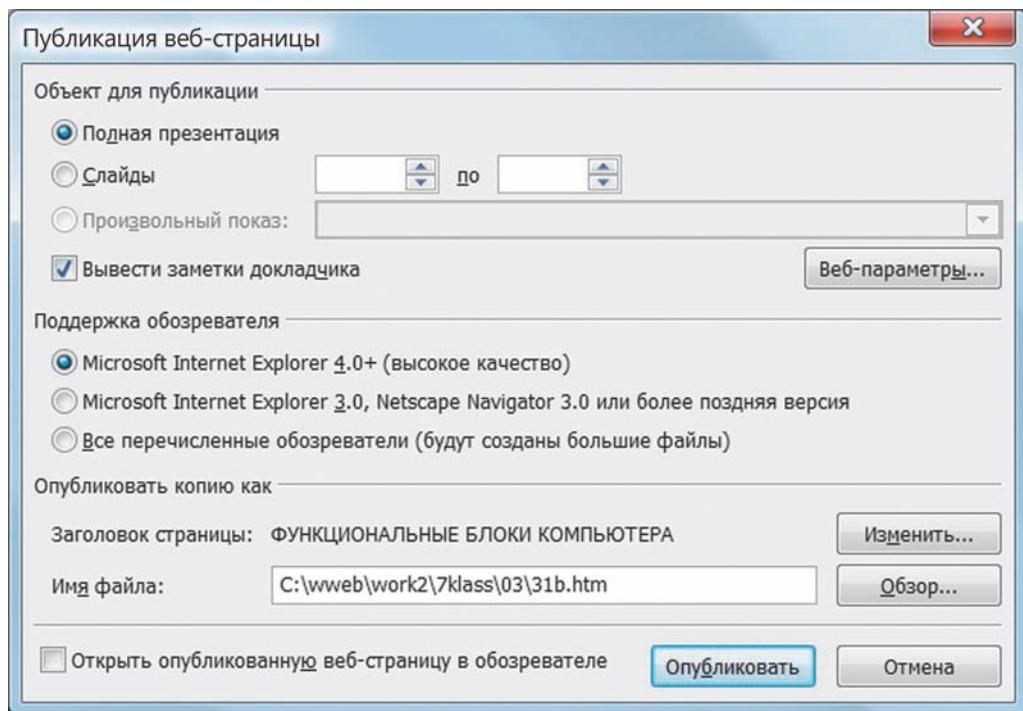


Рис. 1.11

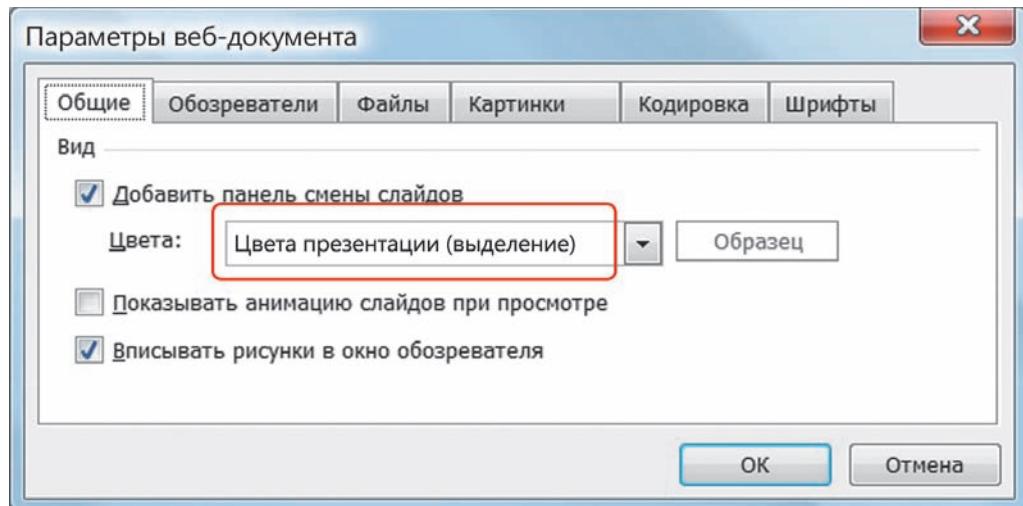


Рис. 1.12

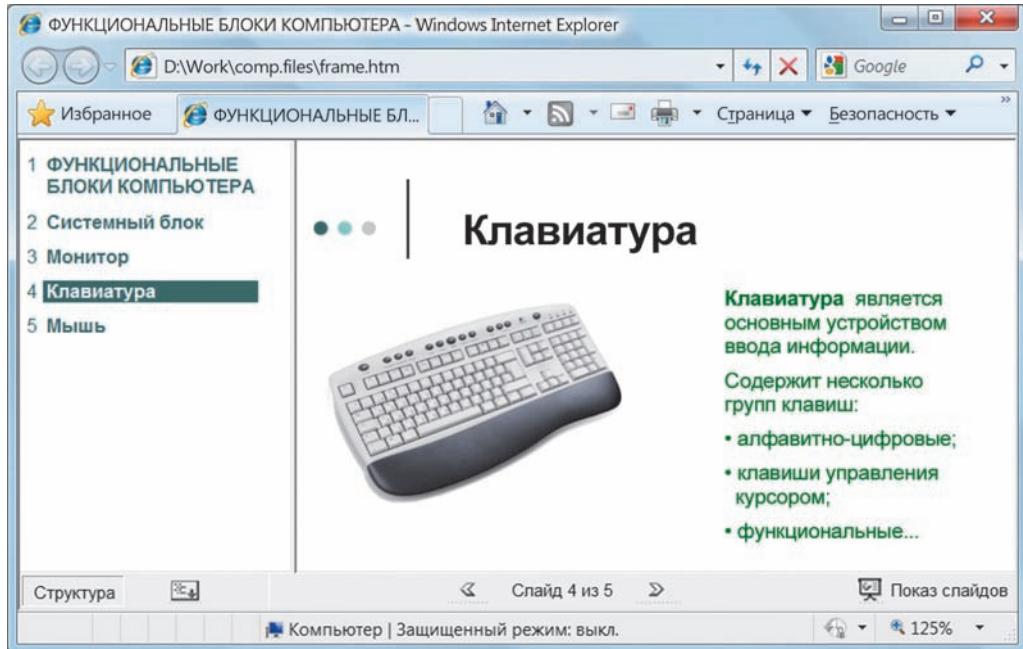


Рис. 1.13

Сохраним документ. Просмотрим сохраненные веб-страницы. Вид одной из них показан на рисунке 1.13.

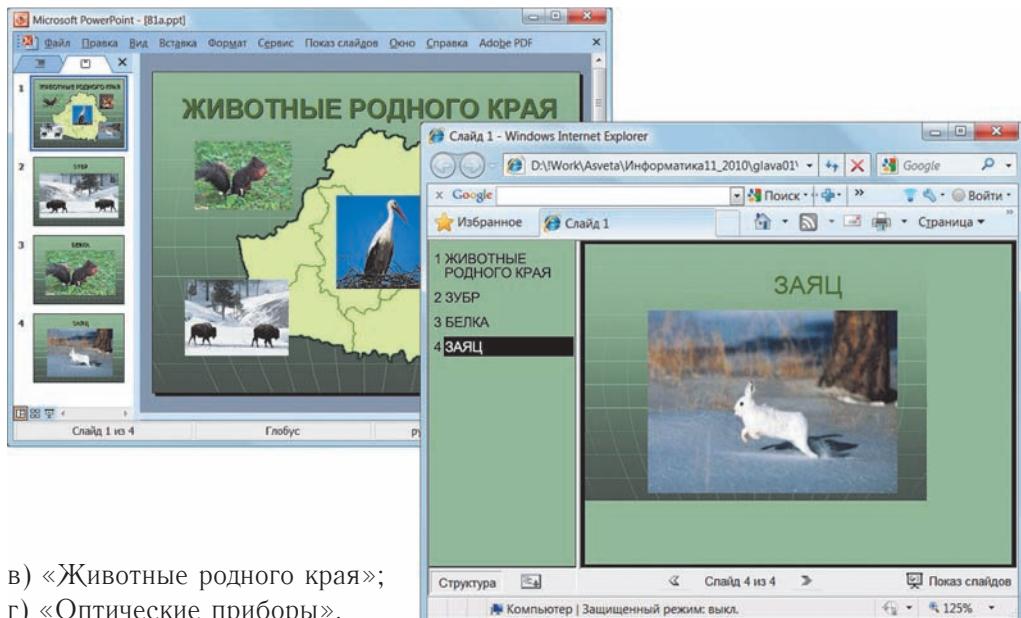
В заключение отметим, что сохранение документов MS Office в виде веб-страниц является самым простым способом создания веб-сайтов и широко применяется в сфере образования для размещения в сети разнообразных материалов учебного назначения: сочинений, рефератов, докладов и презентаций. В профессиональном веб-конструировании эти инструменты и методы практически не используются из-за неоптимальности HTML-кода получаемых страниц, что затрудняет их редактирование и приводит к очень большим размерам файлов и медленной загрузке страниц.

- ?**
1. С помощью каких действий можно сохранить презентацию PowerPoint в виде веб-документа?
 2. Какие параметры веб-страниц можно настраивать?

Упражнение

Создайте фрагмент сайта на основе готовой презентации:

- «Электрический ток»;
- «Пространственные фигуры»;



- в) «Животные родного края»;
г) «Оптические приборы».

§ 3. Основы языка разметки HTML

3.1. Создание HTML-документа в редакторе Блокнот

Веб-страница представляет собой текстовый документ, в котором расставлены команды языка HTML. Они интерпретируются браузером. Например, эти команды могут указывать, как должно отображаться содержание страницы на экране.

HTML-документ можно создавать в простейших текстовых редакторах, например в редакторе **Блокнот**, а затем сохранять в файле с расширением .htm или .html.

Разметка HTML-документа состоит в расстановке **тегов** — заключенных в угловые скобки команд языка HTML. Их можно набирать прописными или строчными латинскими буквами.

Краткое описание основных тегов приведено в Приложении 1.

Большинство тегов парные. Открывающим тегом описывается команда и начинается ее действие. Закрывающим тегом, который повторяет открывающий, но предваряется косой чертой / (слешем), это действие прекращается. Например, тег **** задает жирное начертание. Размеченный текст «Второе **слово** выделено жирным шрифтом» браузером будет отображаться так:

Второе **слово** выделено жирным шрифтом.

Рассмотрим структуру простейшего HTML-документа.

```
<html>
  <head>
    <title>Заголовок окна</title>
  </head>
  <body>
    Содержание документа
  </body>
</html>
```

HTML-документ начинается отывающим тегом `<html>`, а заканчивается — закрывающим `</html>`. Внутри, как в контейнере, расположены два блока.

В блоке `<head>...</head>` (голова) размещается неотображаемая на странице служебная информация. Например, текст, заключенный между тегами `<title>...</title>`, отображается не на странице, а в заголовке окна браузера.

Вся отображаемая браузером информация: тексты, рисунки, видеофрагменты, анимации — размещается в блоке `<body>...</body>` (тело).

В HTML-документе можно также размещать неотображаемые браузером комментарии в угловых скобках с восклицательным знаком `<!Комментарии>`.

Пример 1. В редакторе **Блокнот** создать HTML-документ, содержащий текст:

Авторская страница

Это мой первый проект

Фамилия Имя Отчество

Сохранить созданный документ под именем `primer1.htm`. Просмотреть его в браузере.

Откроем редактор **Блокнот**. Наберем или скопируем готовый шаблон HTML-документа из файла `шаблон.htm`. Наполним его требуемым содержанием, т. е. введем заданный текст, как показано на рисунке 1.14.

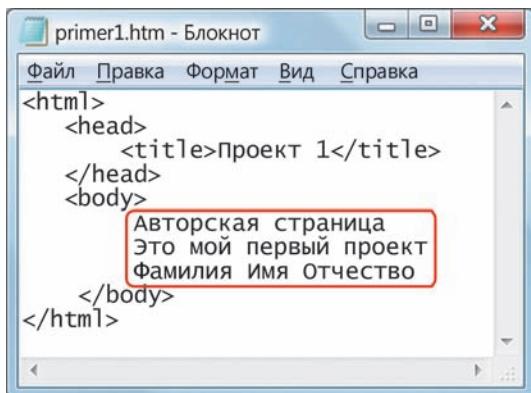


Рис. 1.14

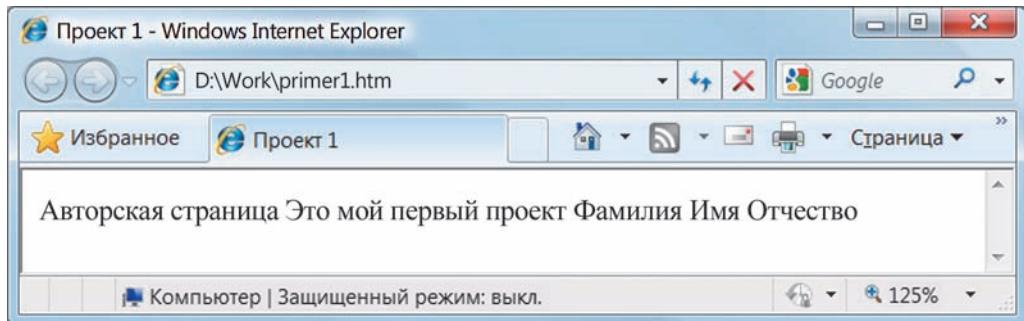


Рис. 1.15

Сохраним документ в формате HTML. Поскольку созданные в редакторе **Блокнот** документы по умолчанию сохраняются с расширением .txt, воспользуемся меню **Файл** → **Сохранить как**, в появившемся диалоговом окне выберем вариант **Все файлы**, введем имя и расширение: primer1.htm.

Откроем созданный документ в браузере (рис. 1.15). Обратим внимание, что текст отображается в одну строку, хотя набран в три строки с отступами. Переходы на новые строки, а также все пробелы более одного браузером игнорируются.

Итак, вид веб-страницы в браузере задается тегами HTML, но также зависит от типа браузера. В приведенных примерах используется браузер MS Explorer.

Для отображения текста в виде отдельных абзацев используют теги **<p>**. При просмотре в браузере абзацы отделяются друг от друга пустой строкой. Для принудительного перехода на новую строку без создания абзаца используют непарный тег **
. Нередко между абзацами помещают разделительную линию, которая задается непарным тегом **<hr>.

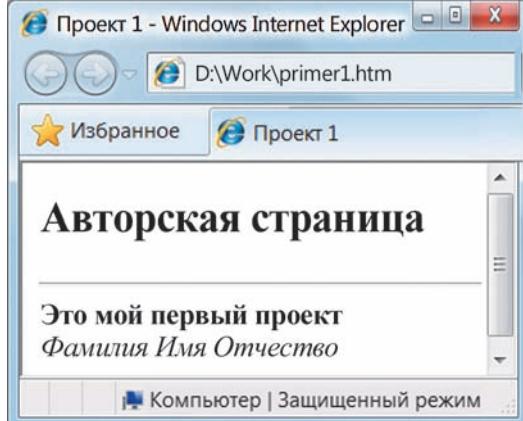


Рис. 1.16

За оформление текста отвечают теги форматирования (см. Приложение 1). Для отображения заголовков используются теги **<h1>...<h6>**. Заголовок уровня 1 — самый крупный, а уровня 6 — самый мелкий.

Начертание символов задается следующими тегами: **** — **полужирное**, **<i>** — **курсивное**, **<u>** — **подчеркнутое**.

Пример 2. Оформить созданную в примере 1 страницу в соответствии с рисунком 1.16.

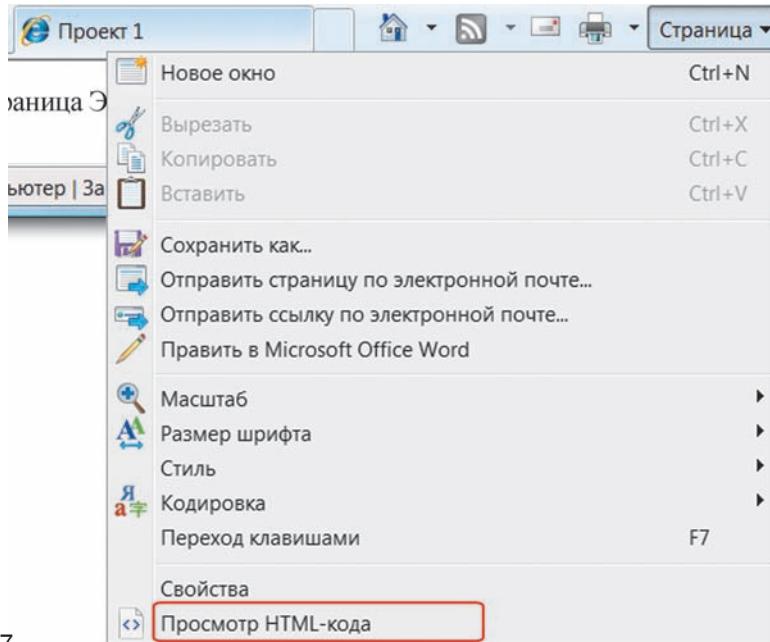


Рис. 1.17

Отредактируем HTML-документ в редакторе **Блокнот**. Его можно открыть непосредственно из браузера с помощью команды **Страница** (или **Вид**) → **Просмотр HTML-кода** (рис. 1.17).

Расставим теги:

```
<h2>Авторская страница</h2><hr>           <!Заголовок уровня 2>
<b>Это мой первый проект</b> <br>        <!Жирный шрифт>
<i>Фамилия Имя Отчество</i>             <!Курсив>
```

Сохраним документ в файле с именем `avtor.htm`. Для просмотра обновленной страницы нажмем кнопку **Обновить** или клавишу F5.

Открывающие теги языка HTML могут содержать **атрибуты**, которыми задаются параметры разметки документа. Каждый атрибут имеет название (имя) и определенное значение, которое записывается в кавычках после знака «равно». Тег может иметь несколько атрибутов, которые перечисляют через пробелы, например:

```
<тег атрибут1="значение" атрибут2="значение" ...>.
```

Порядок следования атрибутов в теге неважен. Если значение атрибута содержит только буквы английского алфавита, цифры или дефис и не содержит пробелов, кавычки можно опустить.

Вид всей веб-страницы задается атрибутами тега `<body>`, в то время как отдельные ее элементы, например заголовки, разделы, абзацы, таблицы, рисунки, могут иметь свое особенное оформление.

Цвет фона страницы задается атрибутом `bgcolor`, а цвет текста — атрибутом `text`. Значением этих атрибутов является цвет, который задается своим названием на английском языке, например `red` (красный), или его шестнадцатеричным кодом, например `#FF0000` (Приложение 2). По умолчанию цвет страницы белый, а цвет текста — черный.

Синий цвет страницы можно задать так: `<body bgcolor="blue">` или так: `<body bgcolor="#0000FF">`.

Заметим, что цвета, отображаемые разными браузерами, а также выводимые на печать, могут несколько отличаться от приведенных в Приложении 2.

Для выравнивания текста всей страницы, отдельного абзаца, раздела или заголовка используется атрибут `align`, который может принимать следующие значения: `center` — выравнивание по центру, `left` — по левому краю, `right` — по правому краю. Выравнивание текста по ширине использовать не рекомендуется, поскольку при уменьшении окна браузера между словами могут образовываться большие промежутки.

Для задания шрифта, цвета и размера символов текста используется тег ``. Шрифт задается атрибутом `face`, значением которого является название шрифта, например `Arial`. Цвет символов задается атрибутом `color`. Если шрифт не задан, то по умолчанию используется шрифт `Times` черного цвета.

Размер символов задается атрибутом `size` и может выражаться в условных единицах, которые могут принимать значения от 1 до 6. По умолчанию принято значение размера 3.

Размер символов может выражаться и в относительных единицах: числах по отношению к базовому размеру, например `size="+n"` или `size="-n"`. Так, для базового размера 3 атрибут `` задает размер 5, а атрибут `` — размер 1.

Пример 3. Создать веб-страницу в соответствии с рисунком 1.18.

Откроем в редакторе **Блокнот** файл `rgimeg3.txt` с текстом объявления и введем HTML-код:

```
<html>
  <head>
    <title>Объявление</title>
  </head>
```

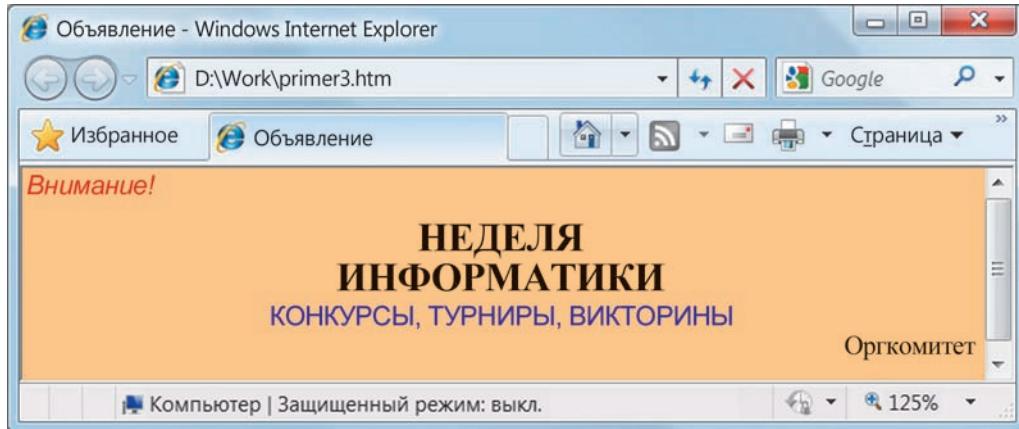


Рис. 1.18

```

<body bgcolor="gold">
    <font face="Arial" size="+2" color="red"><i>Внимание!
    </i></font>
    <h1 align="center">НЕДЕЛЯ<br>ИНФОРМАТИКИ</h1>
    <font face="Arial" size="+2" color="blue">
        КОНКУРСЫ, ТУРНИРЫ, ВИКТОРИНЫ</font>
    <p align="right">Оргкомитет</p>
</body>
</html>

```

Заметим, что допускается комбинирование и вложенность тегов. Так, для отображения слова «Внимание!» шрифтом Arial и курсивным начертанием в тег `` вложен тег `<i>`.

Сохраним документ в файле с именем Primer3.htm. Просмотрим его в браузере.



Фрагмент текста может отображаться в виде бегущей строки с помощью тега `<marquee>`. Направление движения указывается атрибутом `direction`. Например, движение слова «Внимание!» слева направо задается так:

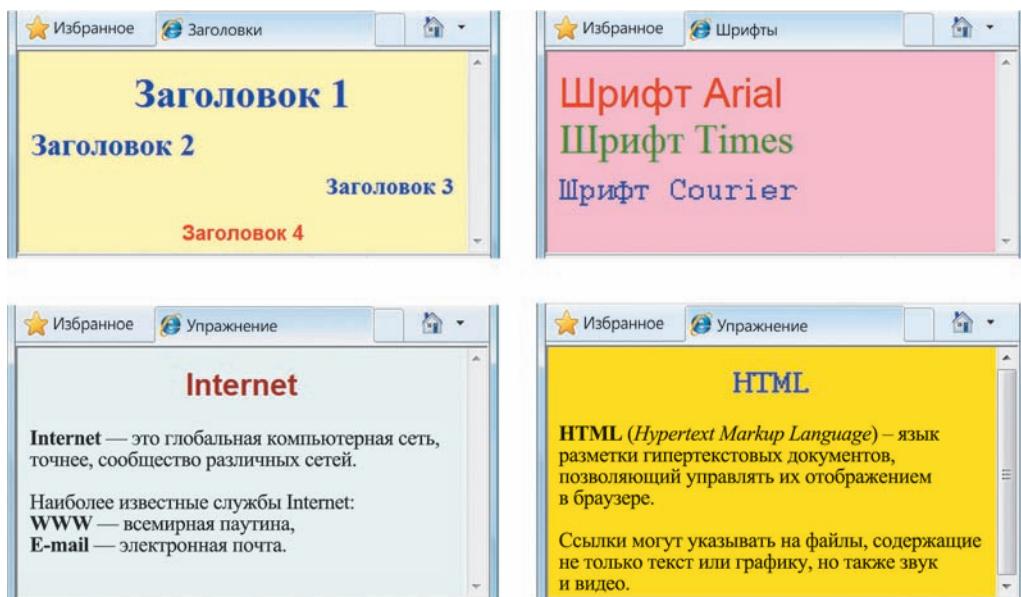
```
<marquee direction="right">Внимание!</marquee>
```



- Каким тегом задается абзац? Переход на новую строку?
- Какими тегами задается полужирное и курсивное начертание символов?
- Как задать фон и цвет текста всей веб-страницы?
- Какими тегами и атрибутами задается шрифт, размер и цвет символов?

Упражнение

Откройте предложенный учителем текстовый документ в редакторе **Блокнот**. Оформите и сохраните его в виде веб-страницы, изображенной на одном из рисунков. Для задания цвета фона воспользуйтесь Приложением 2.



3.2. Изображения на веб-страницах

Важную роль в оформлении веб-страниц играют графические объекты: фотографии, рисунки, фоновые изображения («обои»), разделительные линии. Они делают страницы более привлекательными, а во многих случаях являются и основными источниками информации.

Предназначенные для веб-страниц изображения можно создавать и редактировать в различных графических редакторах. Они могут быть получены с помощью цифрового фотоаппарата, сканера или взяты из сети Интернет. Наиболее предпочтительны графические форматы GIF, JPG и PNG, файлы которых имеют небольшие размеры, что позволяет ускорить процесс загрузки. В формате **JPG** сохраняют полутоночные изображения, например фотографии. Формат **GIF** поддерживает анимацию и прозрачный цвет.

Важно подчеркнуть, что все изображения, которые мы видим на веб-странице, хранятся в отдельных файлах, а в HTML-коде имеются лишь ссылки на них.

Для отображения рисунков предназначен непарный тег ``. Его неотъемлемым атрибутом является имя графического файла ``. Веб-страницы и файлы с изображениями могут храниться в разных папках, тогда в атрибуте `src` тела `` необходимо указывать путь. Указание путей обязательно для ссылок на все объекты, например на другие страницы, аудио- и видеофрагменты.

Тег `` может содержать и другие атрибуты, которые определяют способ отображения рисунка.

Высоту и ширину отображаемого на экране рисунка в пикселях можно задать значениями атрибутов `height` и `width` соответственно, а толщину рамки вокруг изображения — значением атрибута `border`. По умолчанию рамки нет, т. е. `border="0"`. Указание размеров позволяет увеличить или уменьшить видимое на экране изображение (при этом исходный рисунок и файл, в котором он хранится, остаются неизменными). Если размеры не заданы, то по мере загрузки рисунка может изменяться расположение текста и других объектов на экране. Чтобы этого не происходило, полезно всегда указывать эти атрибуты, даже если изображение не масштабируется.

Значения атрибута `align` задают выравнивание текста относительно рисунка (`top` — по верхнему краю; `middle` — по середине; `bottom` — по нижнему краю) или способ обтекания рисунка текстом (`left` — рисунок слева от текста; `right` — рисунок справа от текста).

С помощью атрибута `alt` можно задать текст сообщения, которое будет выводиться вместо рисунка, если он не найден или если отключен его показ в браузере. Кроме того, этот текст появляется в виде подсказки при подведении курсора мыши к рисунку.

В качестве фона страницы можно использовать изображение из файла. Фоновый рисунок («обои») задается с помощью атрибута `background` в теге `<body>`. Чтобы повторяющийся фоновый рисунок заполнял страницу без стыков, его верхняя и нижняя, левая и правая стороны не должны отличаться. Задать фоновый рисунок и одновременно закрасить его некоторым цветом нельзя.

Пример 1. Разместить на веб-странице изображения из файлов в соответствии с рисунком 1.19.

В редакторе **Блокнот** откроем файл `graf.txt` с шаблоном страницы.

В теге `<body>` укажем рисунок фона `kletka1.gif`. Расставим теги форматирования текста. Вставим теги для отображения рисунка волка с прозрачным фоном из файла `volk.gif` (без атрибутов); анимации из файла `tv.gif` (граница толщиной 2 пикселя); двух фотографий аиста из файла `aist.jpg` разных размеров.

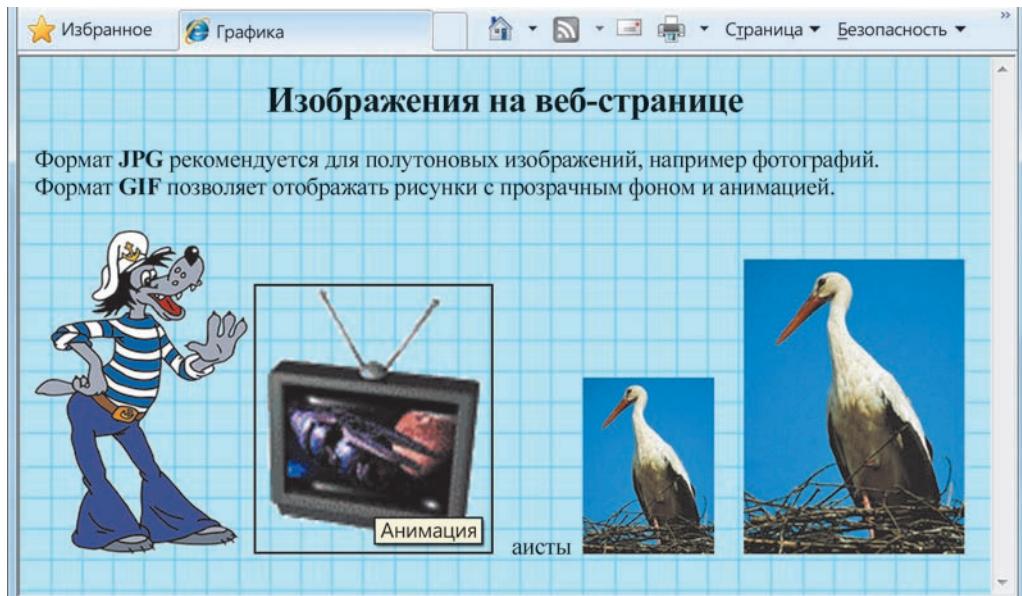


Рис. 1.19

HTML-код веб-страницы может иметь следующий вид:

```
<html>
<head><title>Графика</title></head>
<body background="kletka1.gif">
    <h2 align="center">Изображения на веб-странице</h2>
    Формат <b>JPG</b> рекомендуется для полутоночных изображений, например фотографий.<br>
    Формат <b>GIF</b> позволяет отображать рисунки с прозрачным фоном и анимацией.<p>
        
        
        аисты 
        
    </body>
</html>
```

Сохраним HTML-документ в файле с именем graf.htm. Просмотрим его в браузере. При наведении курсора на область рисунка tv.gif появляется подсказка «Анимация».

Чтобы отображать текст на экране в том же виде, что и в редакторе **Блокнот**, используют тег `<pre>`. Это позволяет упростить форматирование, например не вставлять теги `
` в конце каждой строки.

Пример 2. Создать веб-страницу «Мой родны кут» (рис. 1.20).



Рис. 1.20

В редакторе **Блокнот** откроем файл `kut.txt` с текстом отрывка из поэмы Якуба Коласа «Новая зямля». Добавим теги начала и завершения веб-страницы. Расставим теги форматирования. Заключим текст стихотворения в теги `<pre>...</pre>`.

Ниже текста вставим фотографию музея Якуба Коласа из файла `muzkolas.jpg`. В качестве фона используем изображение листьев из файла `bg02.jpg`.

HTML-код этой веб-страницы может иметь следующий вид:

```
<html>
  <head><title>Мой родны кут</title>
  </head>
  <body background="bg02.jpg">
    <h2 align="center">Мой родны кут</h2>
    <i>Якуб Колас. З паэмы "Новая зямля"</i>
```

```
<pre>Мой родны кут, як ты мне мілы,  
Забыць цябе не маю сілы!  
Не раз, утомлены дарогай,  
Жыццём вясны мае убогай,  
К табе я ў думках залятаю,  
І там душою спачываю.</pre>  
  
</body>  
</html>
```

Сохраним HTML-документ в файле с именем kut.htm.

От удачного выбора цвета фона или фонового рисунка, шрифта и цвета текста, а также других элементов оформления страницы зависит восприятие ее содержания. В Интернете свободно распространяется громадное количество «обоев» и готовых шаблонов веб-страниц разного назначения (например, на сайте «Азбука дизайнера»). Создание собственных веб-страниц на их основе сводится в основном к вводу текста и ссылок на нужные изображения.



Весьма популярны, например, шаблоны поздравительных открыток и рекламных страниц. Такой шаблон несложно разработать самостоятельно.



Пример 3. Создать веб-страницу — шаблон поздравительной открытки (рис. 1.21).



Рис. 1.21

HTML-код веб-страницы может иметь следующий вид:

```
<html>
  <head><title>Открытка</title>
  </head>
  <body background="bg003b.jpg" text=maroon>
    <font face="courier">
      <h1 align="center">Поздравляю<br>с днем рождения!<br>
      </h1>
    </body>
</html>
```

На основе этого шаблона можно создавать различные открытки, изменяя текст поздравления, «обои» и рисунок.



1. Какой тег отображает рисунок на веб-странице?
2. Как задаются размеры изображения?
3. Какой тег задает фоновый рисунок на веб-странице?

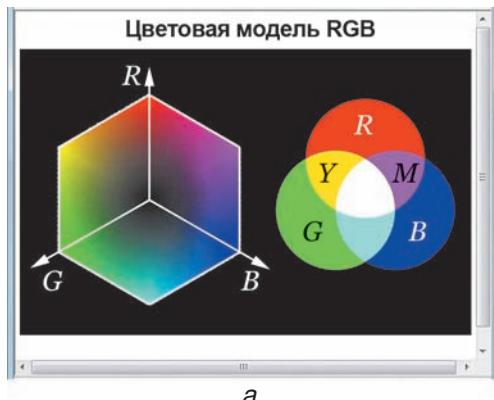
Упражнения

1. Разместите на одной странице несколько изображений:
а) разного размера (например, 150×120, 100×80);



- б) в рамках разной толщины (например, 0, 1, 3, 8).

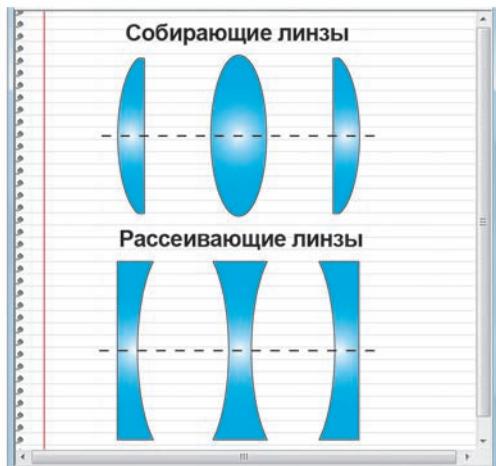


*а**б*

Спадчына
Янка Купала

Ад прадзедаў спакон вякоў
Нам засталася спадчына;
Паміж сваіх і чужакоў
Яна мне ласкай матчынай.

Аб ёй мне баюць казкі-сны
Вясennія праталіны,
І лесу шелест верасны,
І ў полі дуб апалены.

в*г*

2. Откройте предложенный учителем текстовый документ в редакторе **Блокнот**. Оформите и сохраните его в виде веб-страницы, изображенной на одном из рисунков.

3.3. Создание гиперссылок

Переходы со страницы на страницу веб-сайта выполняются с помощью гиперссылок. За организацию ссылок на языке HTML отвечает тег `<a>` с обязательным атрибутом `href`, значением которого является адрес (URL) ресурса. Например, HTML-код текстовой гиперссылки на страницу «Графика», которая сохранена в файле `graf.htm` в той же папке, что и исходная страница, имеет вид:

`Графика`, т. е. адресом ресурса является имя файла.

Обычно близкие по тематике страницы сайта сохраняют в соответствующих папках. Пусть, например, на диске D: в папке kino хранят главную страницу main.htm, во вложенной папке dramy — страницу фильма drama2.htm, а в папке multy — страницы karlson.htm и ну погоди.htm (рис. 1.22). В этом случае при создании гиперссылок необходимо указывать пути к файлам. Возможны два варианта:

- записать *полный путь* от корня диска к искомой странице;
- указать *относительный адрес* (путь от исходной страницы к искомой). Например, ссылка со страницы main.htm на страницу karlson.htm будет такой:

`Карлсон`.

Ссылка со страницы drama2.htm (папка dramy) на страницу karlson.htm (папка multy) будет иметь вид:

`Карлсон`.

Символы `.. /` (две точки и слеш) обозначают возврат в папку kino из вложенной папки dramy.

При переносе папки kino со всеми вложениями в другое место, в том числе на другой диск или компьютер, относительные адреса не изменяются!

Гиперссылкой может служить не только текст, но и рисунок. Для создания такой гиперссылки между парой тегов `` нужно вставить тег, указывающий на файл изображения, например:

``.

Возможно совместное использование текста и графики в одной ссылке, например:

`Графика`.

По умолчанию вызванная гиперссылкой страница открывается в том же окне, что и исходная, замещая ее. Назад можно вернуться с помощью соответствующей кнопки на панели инструментов браузера. Для просмотра вызванной страницы в новом окне необходимо указать атрибут `target="_blank"`, например:

`Графика`.

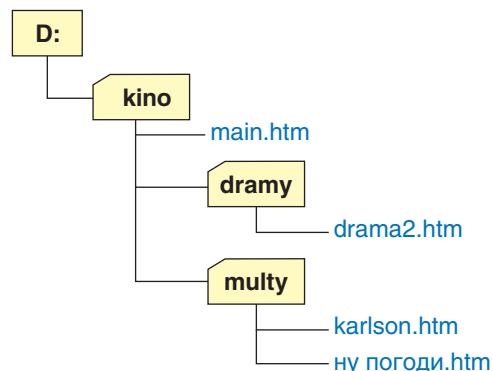


Рис. 1.22

Гиперссылка может указывать не только на веб-страницу, но и на любой размещенный в сети по известному адресу информационный ресурс: рисунок, текст, презентацию, аудиовидеофрагмент. Конечно, на компьютере пользователя должно быть установлено соответствующее программное обеспечение для его отображения.

Пусть, например, реферат по астрономии luna.doc размещен в папке astro на сайте www.referaty.by. Гиперссылка для его просмотра или скачивания может быть задана тегами:

```
<a href="www.referaty.by/astro/luna.doc">Луна</a>.
```

Еще раз подчеркнем, что расставлять гиперссылки следует только после создания ресурсов, на которые они указывают, и размещения их файлов в заданных папках.

Пример 1. Создать главную страницу веб-сайта «Мои работы» (рис. 1.23). Разместить на ней гиперссылки на страницы «Графика», «Об авторе» (открываются в том же окне), а также на рисунок (открывается в новом окне).

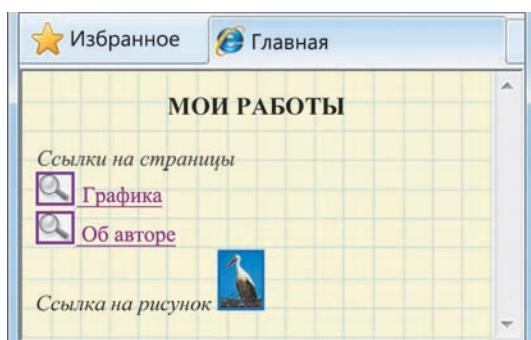


Рис. 1.23

Пусть фрагмент сайта будет размещен в папке PRO. Скопируем в эту папку созданные ранее файлы веб-страниц «Графика» graf.htm и «Об авторе» avtor.htm вместе со всеми используемыми рисунками.

Оформим главную страницу в соответствии с рисунком 1.23. Расставим гиперссылки. HTML-код этой веб-страницы может иметь следующий вид:

```
<html>
  <head><title>Главная</title></head>
  <body background="kletka2.gif">
    <h3 align="center">МОИ РАБОТЫ</h3>
    <i>Ссылки на страницы</i><br>
    <a href="graf.htm">Графика</a><br>
    <a href="avtor.htm">Об авторе</a><br>
    <i>Ссылка на рисунок</i>
    <a href="aist.jpg" target="_blank">
      </a>
  </body>
</html>
```

Сохраним HTML-документ в файле с именем index.htm.

Проверим работу гиперссылок. Страницы «Графика» и «Об авторе» открываются в текущем окне после щелчка мышью по тексту ссылки или картинке с символическим изображением лупы (файл icon.gif). Рисунок (файл aist.jpg) открывается в новом окне после щелчка мышью по его уменьшенному изображению (из того же файла aist.jpg).

Заметим, что для ускорения загрузки страницы с большим количеством графических ссылок следует использовать миниатюрные изображения из отдельных файлов.

В рассмотренном примере все файлы хранятся в одной папке. В случае большого количества страниц или изображений их следует размещать во вложенных папках и указывать пути к ним.



Гиперссылки могут указывать не только на отдельные документы, но и на заданные места в документе. Ссылки внутри длинных страниц повышают удобство их просмотра. Они создаются в два этапа:

1) сначала в месте, куда нужно перейти по ссылке, устанавливается закладка, например:

```
<a name="Закладка1">...</a>;
```

2) затем создается ссылка на эту закладку:

```
<a href="#Закладка1"> текст гиперссылки </a>.
```

Для ссылки на адрес электронной почты его нужно указать в качестве URL, например

```
<a href="mailto:sova@les.by">письмо сове</a>.
```

Как уже отмечалось, в блоке `<head>` могут размещаться теги, которые содержат служебную информацию для браузера и поисковых систем. Они называются **мета-тегами**. Большинство мета-тегов являются необязательными. Однако некоторые из них весьма полезны.

Так, браузер будет отображать текст страницы на национальном языке при указании таблицы кодировки в мета-теге `charset`. Например, для кириллицы мета-тег будет таким:

```
<meta http-equiv="content-type" content="text/html;
charset=Windows-1251">
```

Для хранения информации об авторе используют мета-тег

```
<meta name="author" content="Алесь Палескі">.
```



По словам, перечисленным через запятую или пробел в мета-теге "keywords" (ключевые слова), поисковые системы индексируют веб-страницы, т. е. помещают информацию о них в свою базу данных, чтобы потом оперативно выдавать по запросам пользователей. Длина строки ключевых слов не должна превышать 800 символов. Слова в ней не должны повторяться, например:

```
<meta name="keywords" content="Образование Мультимедиа  
Multimedia Графика Видео Аудио">.
```



1. С помощью какого тега создается гиперссылка?
2. Как в качестве гиперссылки использовать изображение?

Упражнение

Создайте фрагмент веб-сайта из 3—4 веб-страниц по одной из тем: Дом, в котором я живу. Мой класс. Моя школа. Любимые стихи (песни, книги, фильмы). Любимые поэты (писатели, художники, артисты). Любимые занятия (пример главной страницы представлен на рисунке).

ЛЮБИМЫЕ ЗАНЯТИЯ



Спорт



Компьютер



Рыбалка



§ 4. Подготовка изображений для Интернета

При создании веб-сайта приходится заботиться о том, чтобы его страницы загружались как можно быстрее. Это особенно актуально, если на страницах сайта необходимо разместить много изображений. Поэтому так важен грамотный выбор формата их хранения.

Как Вы знаете, нарисованные в графическом редакторе Paint картинки по умолчанию сохраняются в формате BMP (*Bit Map Picture* — битовая карта изображения). В этом формате изображения хранятся в файлах в неупакованном виде. Информационный объем изображения в байтах равен произведению его ширины **W** (width) и высоты **H** (height) в пикселях и глубины цвета **C** в битах на 1 пиксель (**W·H·C**). Так, файл с небольшой картинкой размером 100×100 пикселей при глубине цвета 24 бит/пиксель будет иметь размер около 29 Кбайт. Объем файла с изображением 800×600 пикселей будет иметь размер около 1,4 Мбайт, а время его загрузки при скорости модема 56 кбит/с составит более 200 с, что совершенно неприемлемо.

В большинстве графических форматов изображения хранятся в файлах в упакованном виде. Существуют различные способы уменьшения размера файла изображения с минимальными потерями качества. Этот процесс называют **оптимизацией**.

При большом разнообразии графических форматов для размещения на сайтах в основном используются изображения форматов GIF, JPG и PNG.

Выбор формата хранения изображения во многом зависит от свойств самого изображения. Так, формат **GIF** используют для изображений с четкими линиями, однородными заливками, текстом, например чертежей, карт. Такие изображения называют *штриховыми*. Этот формат поддерживает палитру оттенков, состоящую не более чем из 256 цветов, и позволяет задавать прозрачный фон. Оптимизация заключается в уменьшении количества выбранных цветов. Важным достоинством формата GIF является возможность анимации. Поэтому этот формат широко используется для размещения на веб-страницах простых графических элементов: символов, формул, логотипов, рекламных роликов.

Формат **JPG** позволяет передавать богатую палитру цветов и лучше всего подходит для изображений с плавным переходом тонов без четких линий и контуров, например портретов, пейзажей. Такие изображения называют *полутоновыми*. Оптимизация заключается в замене областей с небольшим градиентом цвета на однотонные. Степень сжатия характеризуют показателем качества от 0 до 100. Часто по умолчанию его принимают равным 65, что соответствует сжатию исходного неупакованного изображения в 6—10 раз. Сохранение изображений в формате JPG предусмотрено практически во всех современных цифровых фотокамерах.

При подготовке изображений для Интернета выработаны определенные рекомендации. Размеры сохраняемых в файлах изображений должны соответствовать размерам изображений, отображаемых на веб-страницах, несмотря на то, что их ширину и высоту можно задавать в теге ``. Например, для мониторов 17' с разрешением 1280×768 пикселей ширина изображения на странице чаще всего не превышает 600—800 пикселей, а высота — 400—600. Если изображение обтекается текстом, то его размер должен находиться в пределах 200—400 пикселей.

Процесс оптимизации изображения сводится к двум основным операциям:

- приведение изображения к требуемому размеру;
- сжатие изображения с сохранением оптимального качества.

Пример 1. Фотография размером 2560×1920 пикселей сохранена в файле объемом 2,4 Мбайт. Ее ширину и высоту уменьшили в 4 раза. Определить объем файла и размеры изображения.

Размер изображения станет равным 640×480 пикселей, при этом объем файла уменьшится в $4 \cdot 4 = 16$ раз и составит 0,15 Мбайт. В 16 раз уменьшится и время загрузки этого изображения!

Заметим, что изменять размеры изображений и сохранять их в форматах GIF или JPEG можно практически в любом растровом графическом редакторе, в том числе в Paint. Для сжатия изображений требуется более сложный редактор.

Рассмотрим оптимизацию изображений в свободно распространяемом графическом редакторе Paint.Net (<http://paintnet.ru>).

Пример 2. Фотография размером 1200×1200 пикселей сохранена в файле goza1200.jpg объемом 745 Кбайт. Оптимизировать изображение так, чтобы на веб-странице его размер составил 300×300 пикселей.

Откроем в редакторе Paint.Net изображение из файла goza1200.jpg (рис. 1.24).

Прежде всего, уменьшим размер изображения. Для этого из меню **Изображение** вызовем окно **Изменение размера** и установим значение **Ширина: 300** пикселей (рис. 1.25). При включенном флагке **Сохранять пропорции** в такой же пропорции изменится и высота.

Сохраним уменьшенное изображение в формате JPG. Для этого с помощью меню **Файл** вызовем диалоговое окно **Сохранить как**, выберем тип файла JPEG и введем новое имя файла, например goza300.jpg. После нажатия кнопки **Сохранить** появится диалоговое окно настройки качества изображения **Параметры сохранения**. По умолчанию показатель **Качество** нашего изображения равен 100. Размер файла составляет примерно 83 Кбайт.

Продолжим оптимизацию путем сжатия изображения с сохранением приемлемого качества. С помощью движка будем уменьшать показатель **Качество** до появления видимых на глаз искажений на оптимизированном изображении. Так, при по-

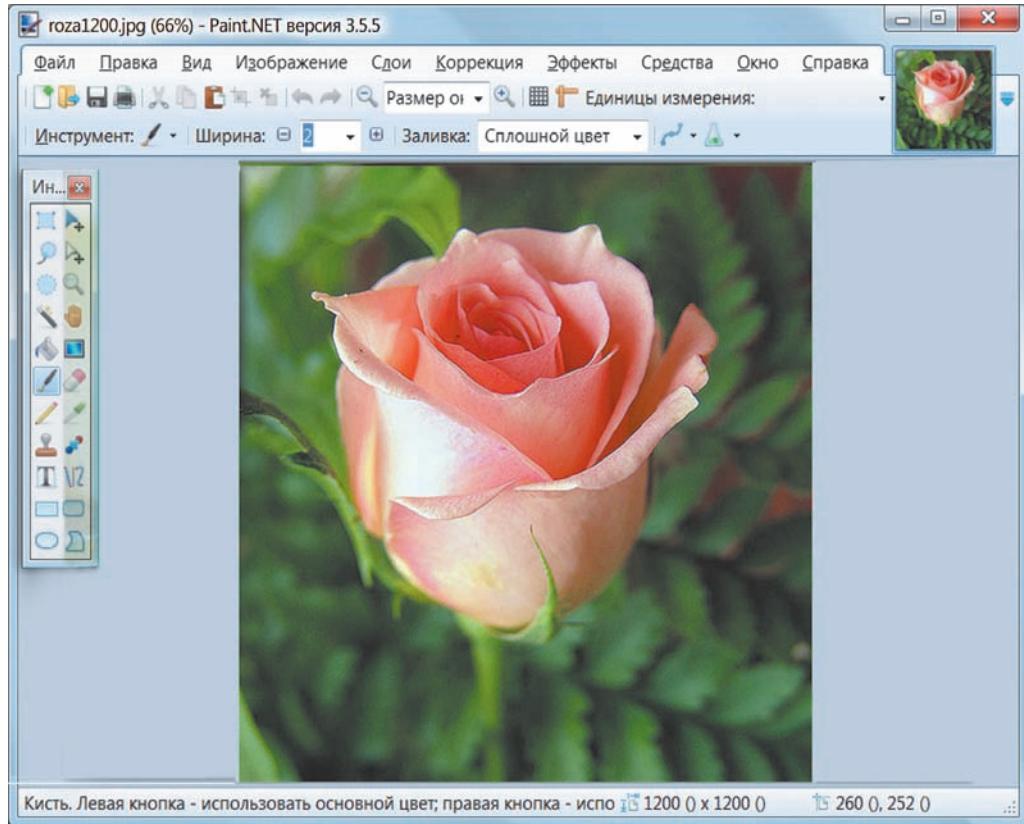


Рис. 1.24

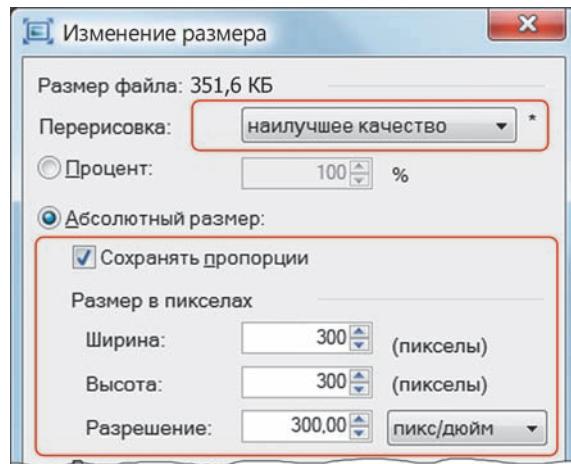


Рис. 1.25

казателе 50 видимых изменений еще нет, а размер файла при этом составляет около 19 Кбайт (рис. 1.26). Остановимся на этом значении, нажав кнопку OK. В итоге качество изображения ухудшилось незначительно, а размер файла с оптимизированным изображением стал почти в 40 раз меньше исходного.

 Гораздо большие возможности оптимизации изображений предоставляет графический редактор Adobe Photoshop.

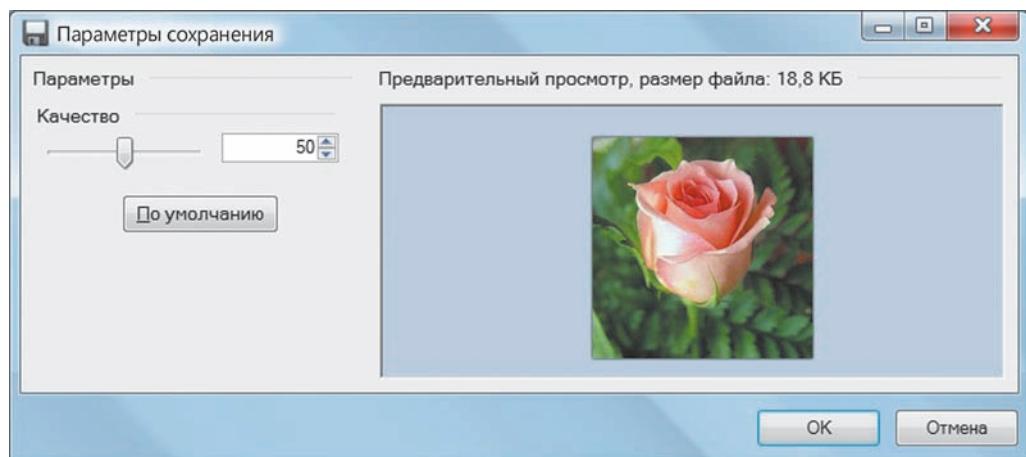


Рис. 1.26

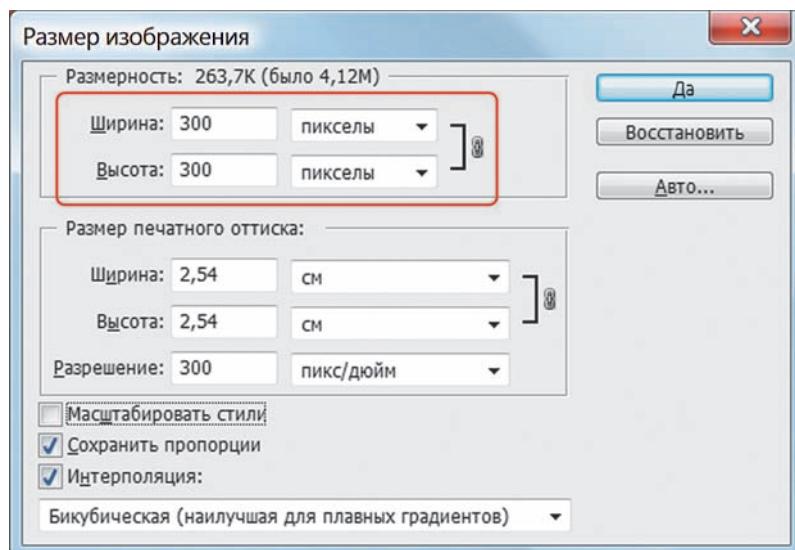


Рис. 1.27



Пример 3. Оптимизировать изображение из файла roza1200.jpg так, чтобы на веб-странице его размер составил 300×300 пикселей.

Откроем в редакторе Adobe Photoshop изображение из файла roza1200.jpg.

Для изменения размера изображения из меню **Изображение (Image)** вызовем окно **Размер изображения (Image Size)** и установим значение **Ширина (Width)** 300 пикселей (рис. 1.27). При включенном флагке **Сохранить пропорции** в такой же пропорции изменится и высота.

Из меню **Файл (File)** вызовем окно **Сохранить для веб и устройств (Save for Web)** и с помощью вкладки **4 варианта** получим четыре изображения: исходное (слева вверху) и оптимизированные с качеством 65 % (справа вверху), 30 % и 10 % (внизу) (рис. 1.28).

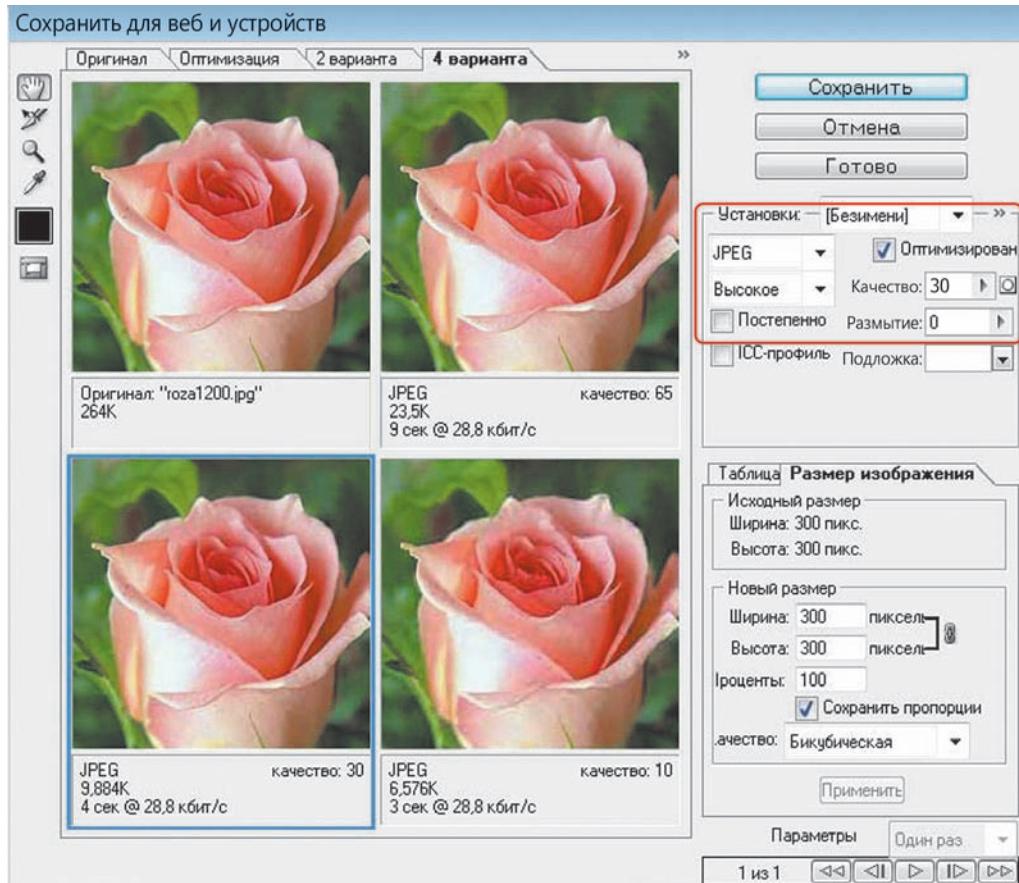


Рис. 1.28

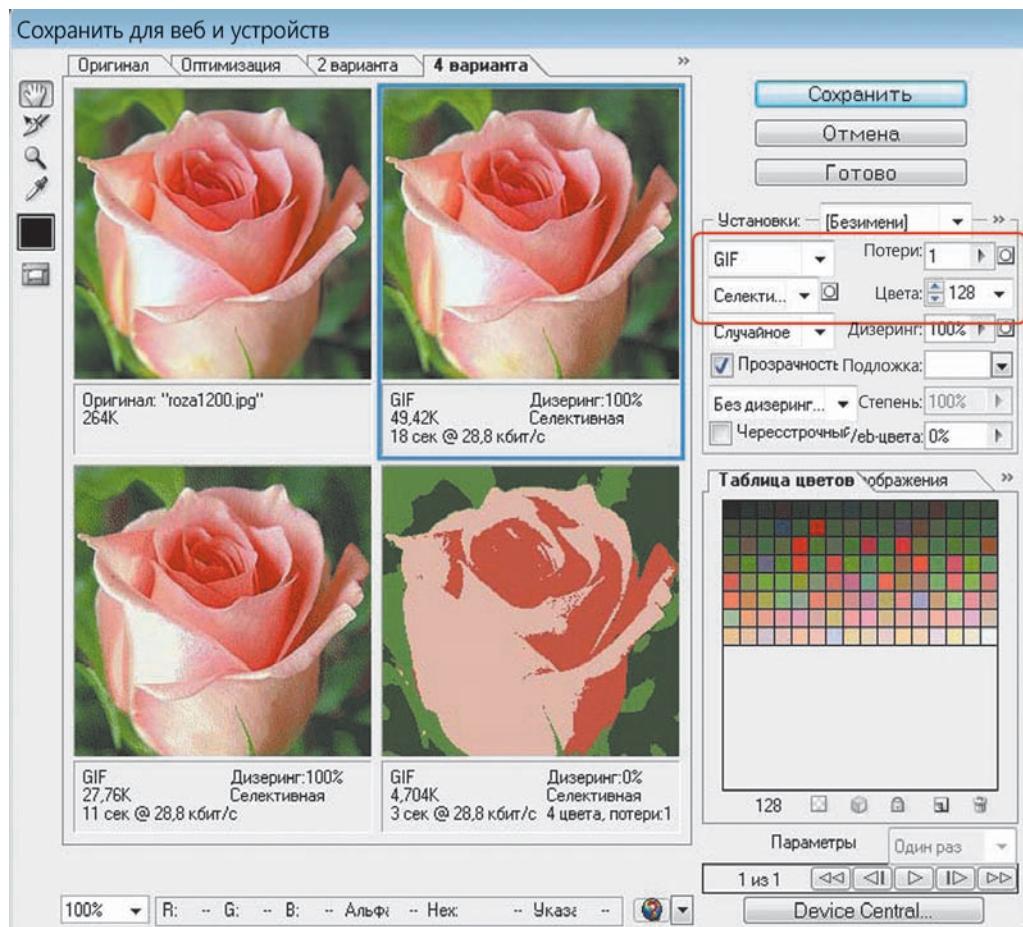


Рис. 1.29

Будем уменьшать показатель **Качество** до появления видимых изменений на оптимизированном изображении. Это можно сделать грубо, выбрав в поле **Установки (Setting)** один из показателей качества (**Максимальное, Высокое, Среднее, Низкое**), и точно, установив подходящее числовое значение. Остановимся, например, на значении 30 и нажмем кнопку **Сохранить (Save)**.

Объем оптимизированного файла не превышает 10 Кбайт, т. е. уменьшился почти в 75 раз. Не забудем ввести новое имя файла, например roza300-30.jpg.

Аналогичным способом оптимизируют изображения в формате GIF. В этом случае уменьшают количество цветов от 256 до значения, при котором появляют-

ся видимые искажения. На рисунке 1.29 показаны четыре изображения: исходное (слева вверху) и оптимизированные с количеством цветов 128 (справа вверху), 32 (слева внизу, потери качества невелики) и 4 (справа внизу, качество неприемлемо).



1. Почему необходимо уменьшать объемы файлов с изображениями для веб-страниц?
2. В каких форматах рекомендуется хранить изображения для веб-страниц?
3. В чем заключается оптимизация изображений для Интернета?

Упражнения

1. Оптимизируйте изображения, сохраненные в предложенных учителем файлах.

2. С помощью клавиши Print Screen поместите в буфер обмена изображение на экране, например Рабочий стол. Откройте редактор Paint, с помощью комбинации клавиш Ctrl + V вставьте изображение из буфера и сохраните под именем ris.bmp. Сохраните это же изображение в форматах JPG и GIF. Сравните объемы файлов.

§ 5. Веб-конструирование в редакторе FrontPage

5.1. Основные элементы интерфейса

Вы уже научились создавать веб-страницы в программах MS Office, а также с использованием языка разметки HTML в простейшем текстовом редакторе **Блокнот**. Рассмотрим теперь возможности специального веб-редактора Microsoft FrontPage, который предназначен для разработки веб-сайтов и относится к программным средствам визуального веб-конструирования. При работе с веб-редактором FrontPage можно обойтись без знания языка разметки гипертекстовых документов HTML. Веб-страница просто конструируется на экране и сохраняется в формате HTML. Выполняемые при этом действия по оформлению веб-документа напоминают работу в текстовом редакторе MS Word.

Интерфейс редактора FrontPage достаточно прост, чтобы пользователь смог быстро освоить основные приемы работы. После запуска FrontPage открывается окно, основные элементы которого представлены на рисунке 1.30.

Меню и панели инструментов **Стандартная**, **Форматирование**, **Рисование** по своим возможностям и приемам использования напоминают аналогичные панели текстового редактора MS Word.

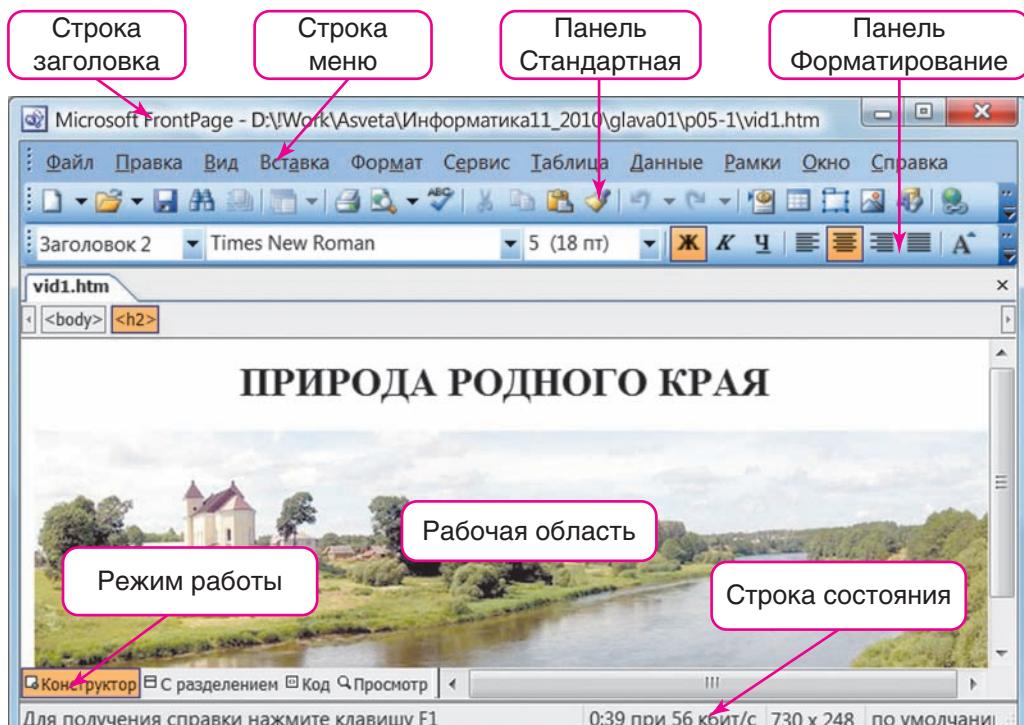


Рис. 1.30

Вид **Рабочей области** зависит от выбранного режима работы.

В режиме **Конструктор** веб-страница строится из текстовых блоков и графических объектов. При этом автоматически генерируется ее HTML-код, который можно просматривать и редактировать в режиме **Код**.

Режим **С разделением** (рис. 1.31) является комбинацией этих двух режимов.

Режим **Просмотр** позволяет просматривать созданные страницы.

С помощью меню **Вид** можно выбрать и другие полезные режимы работы.

Режим **Страница** предназначен для создания и редактирования веб-страниц.

В режиме **Папки** просматривается структура папок сайта, выполняются файловые операции.

Режим **Переходы** отображает структуру связей между страницами и позволяет ее изменять.

Режим **Гиперссылки** наглядно отображает систему ссылок, обеспечивает их проверку и редактирование.

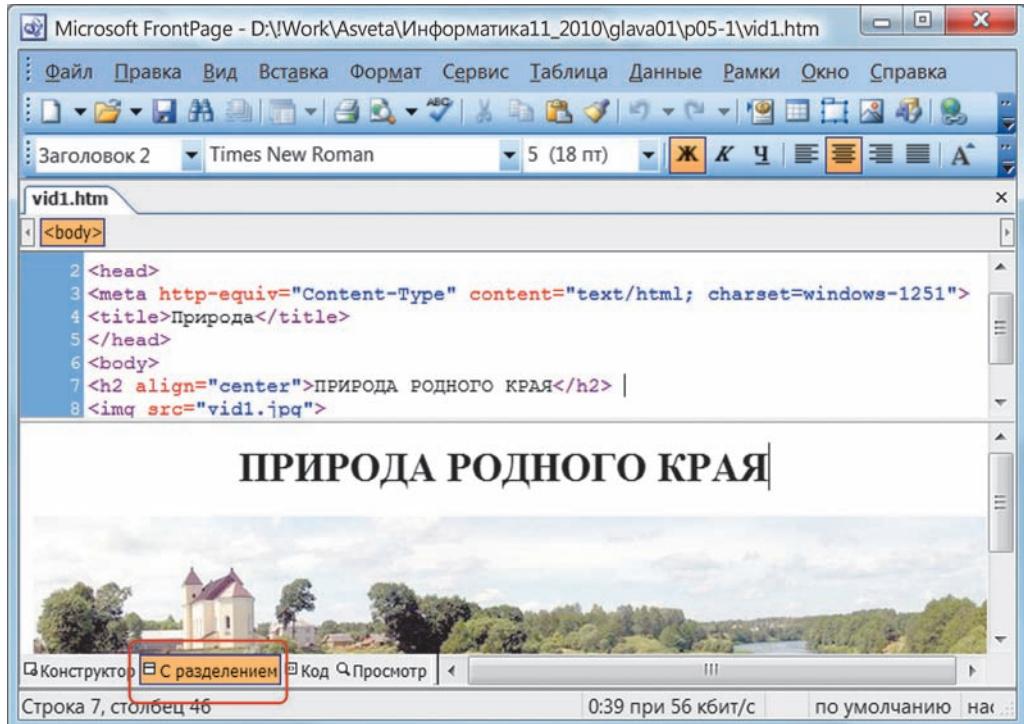


Рис. 1.31

5.2. Работа в редакторе FrontPage

Рассмотрим основные приемы работы в редакторе FrontPage на примере разработки веб-страницы «Озера Беларуси». Она будет содержать текст и фотографии, которые следует подготовить заранее.

Начнем с создания и оформления текстового документа.

Пример 1. Создать веб-страницу «Озера Беларуси» (рис. 1.32).

Запустим редактор FrontPage. Откроется пустая страница с установленными по умолчанию параметрами.

Перед началом работы полезно проверить настройки редактора, и, прежде всего, кодировку. Для этого с помощью команды **Файл → Свойства** откроем окно **Свойства страницы** и на вкладке **Язык** установим **кириллица** (рис. 1.33).

Будем работать в режиме **Конструктор**. Однако при необходимости можем просматривать и редактировать автоматически генерируемый HTML-код в режимах **Код** или **С разделением**.

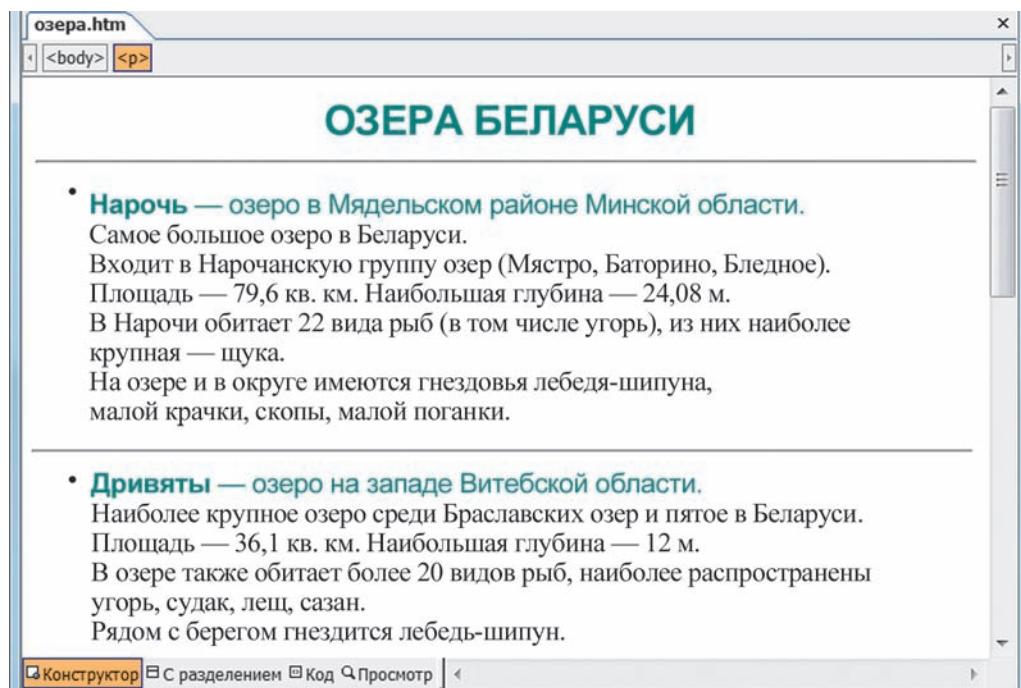


Рис. 1.32

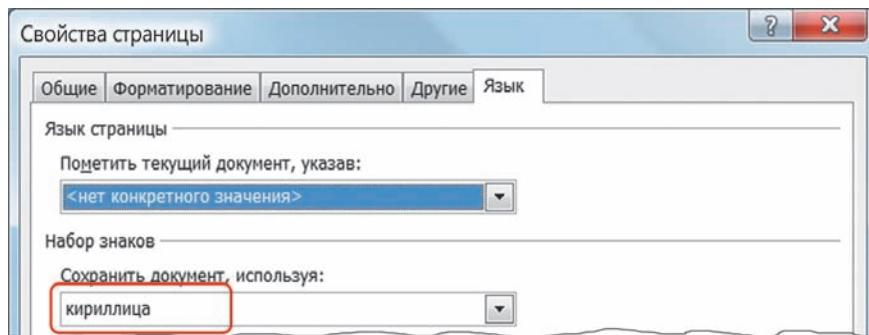


Рис. 1.33

Начнем с размещения и оформления текста, который был предварительно сохранен в файле озера.txt. Откроем его в редакторе **Блокнот**, скопируем нужные фрагменты и с помощью буфера обмена вставим на создаваемую страницу. Текст будет оформлен принятым по умолчанию стилем: шрифт Times, размер 3 (12 пт), выравнивание по левому краю (рис. 1.34).

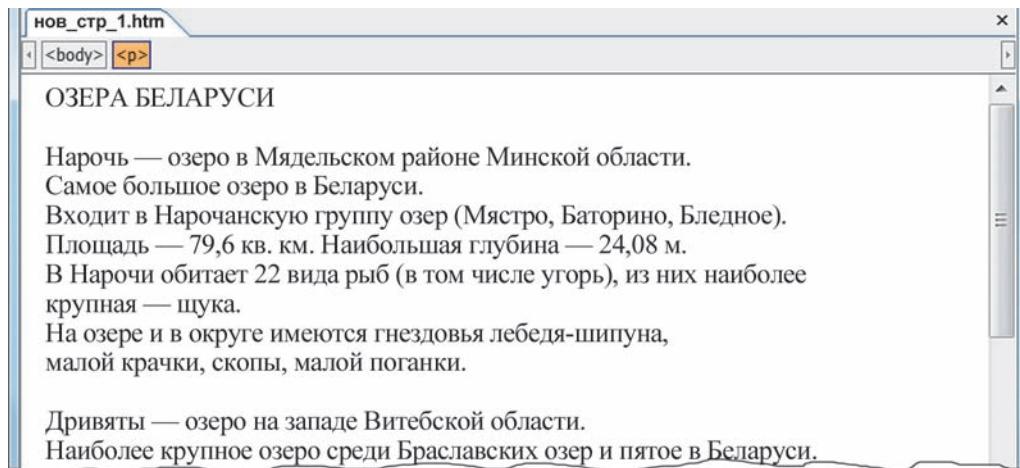


Рис. 1.34

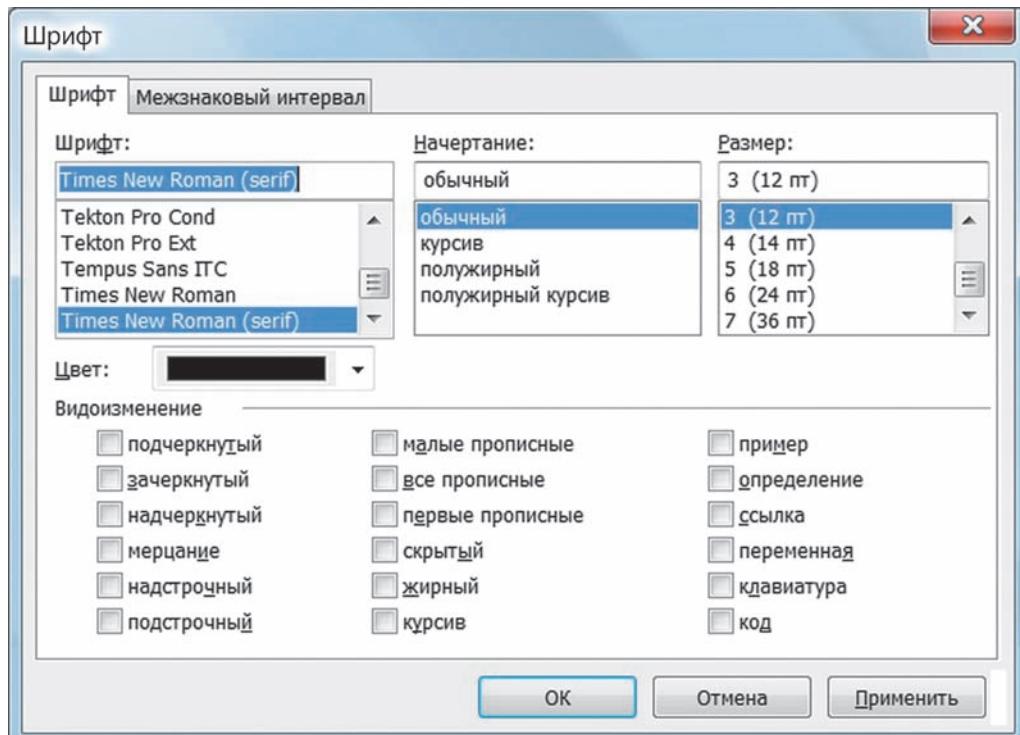


Рис. 1.35

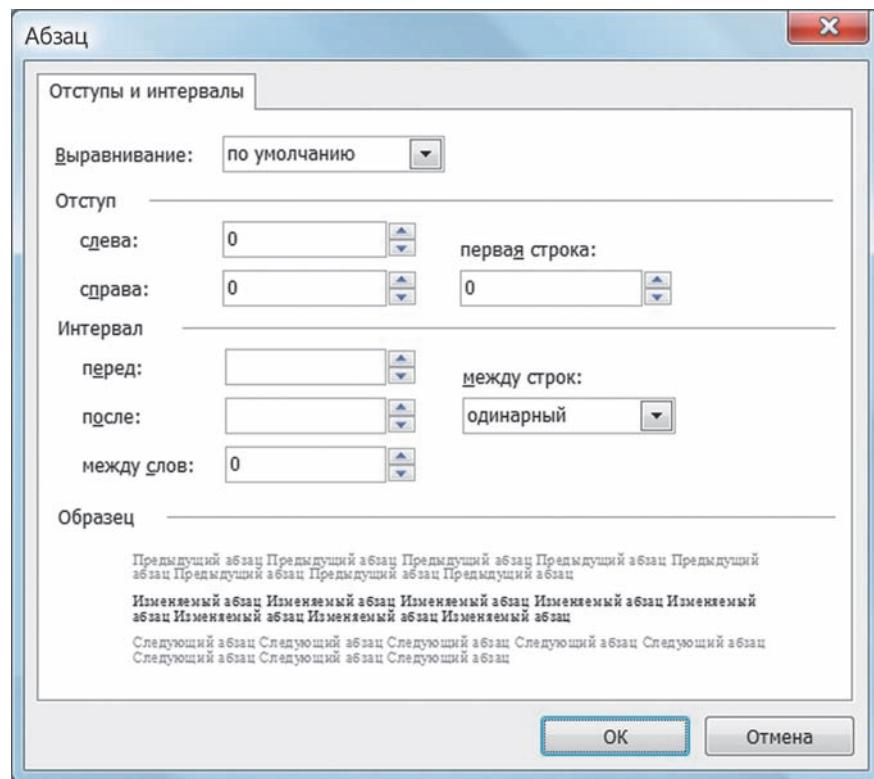


Рис. 1.36

Важно помнить, что отображение веб-страницы на экране зависит от разрешения монитора и настройки браузера. Поэтому размер шрифта принято задавать в условных единицах от 1 до 7. Если размер шрифта не указан, то по умолчанию он принимается равным 3, что при установке в браузере значения **Размер шрифта → средний** соответствует 12 пунктам.

Для форматирования текста используем панель инструментов **Форматирование**, а также окна **Шрифт** (рис. 1.35) и **Абзац** (рис. 1.36) из меню **Формат**.

Установим параметры заголовка: шрифт Arial, размер 5 (18 pt), цвет бирюзовый, выравнивание по центру. Параметры основного текста оставим без изменений, лишь выделим первые строки каждого абзаца (шрифт Arial, цвет бирюзовый). Названия озер выделим полужирным шрифтом.

Повышению удобочитаемости текста и привлекательности страниц способствует использование списков, а также разбиение больших страниц на отдельные блоки с помощью горизонтальных разделительных линий.

Маркированные и нумерованные списки создаются с помощью панели инст-

рументов **Форматирование**, а также диалогового окна **Список**, которое открывается из меню **Формат**. Для оформления веб-страницы «Озера Беларуси» используем графические маркеры.

С помощью команды **Вставка** → **Горизонтальная линия** поместим на страницу горизонтальные разделительные линии. В окне **Свойства горизонтальной линии** зададим параметры линии в соответствии с рисунком 1.37.

Теперь страница будет выглядеть так, как на рисунке 1.32. Сохраним документ, заменив предлагаемое по умолчанию имя `нов_стр_1.htm` на более подходящее `озера.htm`.

Займемся теперь размещением на созданной странице изображений. Подчеркнем, что все изображения, которые мы видим на веб-страницах, хранятся в отдельных файлах, а на самой странице имеются лишь ссылки на соответствующие файлы.

Пример 2. Разместить на веб-странице «Озера Беларуси» изображения в соответствии с рисунком 1.38.

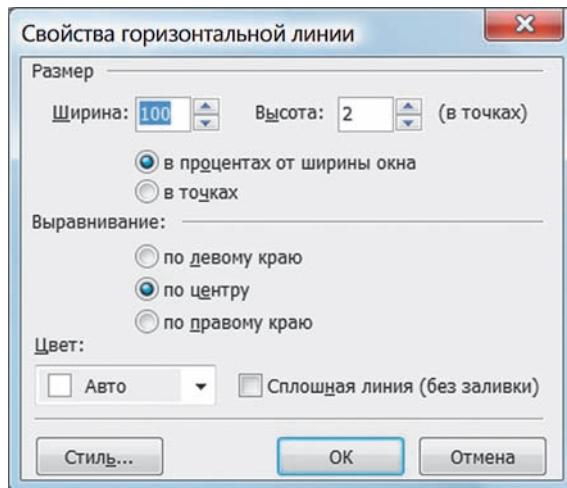


Рис. 1.37

озера.htm

<body> </>

ОЗЕРА БЕЛАРУСИ

- **Нарочь — озеро в Мядельском районе Минской области.**
Самое большое озеро в Беларуси.
Входит в Нарочанскую группу озер (Мястро, Баторино, Бледное).
Площадь — 79,6 кв. км. Наибольшая глубина — 24,08 м.
В Нарочи обитает 22 вида рыб (в том числе угорь), из них наиболее крупная — щука.
На озере и в округе имеются гнездовья лебедя-шипуна, малой крачки, скопы, малой поганки.
- **Дривяты — озеро на западе Витебской области.**
Наиболее крупное озеро среди Браславских озер и пятое в Беларуси.
Площадь — 36,1 кв. км. Наибольшая глубина — 12 м.
В озере также обитает более 20 видов рыб, наиболее распространены угорь, судак, лещ, сазан.
Рядом с берегом гнездится лебедь-шипун.

Конструктор С с разделением Код Просмотр

Рис. 1.38

Установим курсор в то место текста, к которому будет привязан рисунок, например перед началом первой строки. С помощью команды **Вставка** → **Рисунок** или кнопки  на панели инструментов (рис. 1.39) вызовем окно **Рисунок** и выберем требуемый файл с изображением, например narach.jpg.

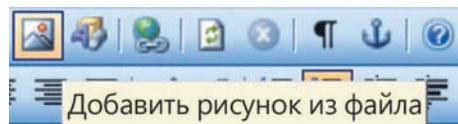


Рис. 1.39

Как и в документах MS Word, изображения могут располагаться по отношению к тексту различными способами. По умолчанию рисунок отображается в выбранном месте строки без обтекания текстом (рис. 1.40, а). При этом можно выбрать способ выравнивания.

На рисунке 1.40, а текст и изображение выровнены по левому краю. Рисунок 1.40, б иллюстрирует обтекание изображения слева. (На это указывает непечатаемый символ стрелка влево  в точке привязки. При необходимости эту точку можно перетаскивать с помощью мыши.)



Рис. 1.40

а

• Нарочь — озеро в Мядельском районе



Самое большое озеро в Беларуси. Входит в Нарочанскую группу озер. Площадь — 79,6 кв. км. В Нарочи обитает 22 вида рыб, крупная — щука.

На озере и в округе имеются гнездовья лебедей, малой крачки, скопы, малой поганки.

б

Параметры изображения настраиваются в диалоговом окне **Свойства рисунка** (рис. 1.41). Вызовем это окно двойным щелчком мыши по выбранному рисунку или с помощью контекстного меню. Зададим **Обтекание: справа, Выравнивание: по правому краю, Толщина границы: 0** и другие параметры в соответствии с рисунком 1.41. Здесь же можно установить размер изображения, но проще подобрать размер рисунка, выделив его и перетащив мышью появившиеся маркеры, как в любом редакторе.

Аналогичным способом вставим второй рисунок из файла braslav.jpg.

В редакторе FrontPage предусмотрена возможность простейшей обработки изображений (коррекция яркости и контрастности, повороты, обрезка и т. д.). С помощью меню **Вид** вызовем панель инструментов **Рисунки**. Увеличим яр-

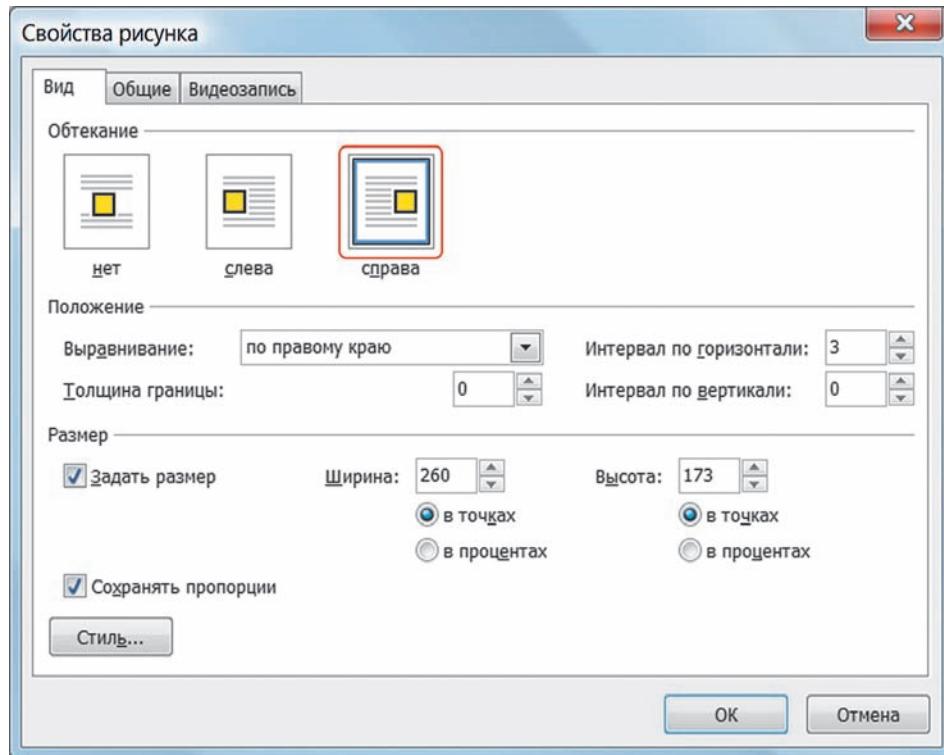


Рис. 1.41

кость и уменьшим контрастность изображений, нажав соответственно кнопки и .

Страница примет требуемый вид (см. рис. 1.38). Сохраним ее под именем озера2.htm и просмотрим в браузере.

Для привлечения внимания к информации можно создать бегущую строку.



Пример 3. На веб-странице «Озера Беларуси» добавить бегущую строку «Приглашаем в лодочный поход».

Установим курсор под нижней горизонтальной линией. Выполним последовательность действий: **Вставка** → **Веб-компонент** → **Бегущая строка**. В окне **Свойства бегущей строки** введем текст «Приглашаем в лодочный поход». Зададим ее параметры в соответствии с рисунком 1.42. Шрифт, размер и начертание текста устанавливается с помощью меню **Шрифт**, которое вызывается кнопкой **Стиль**.

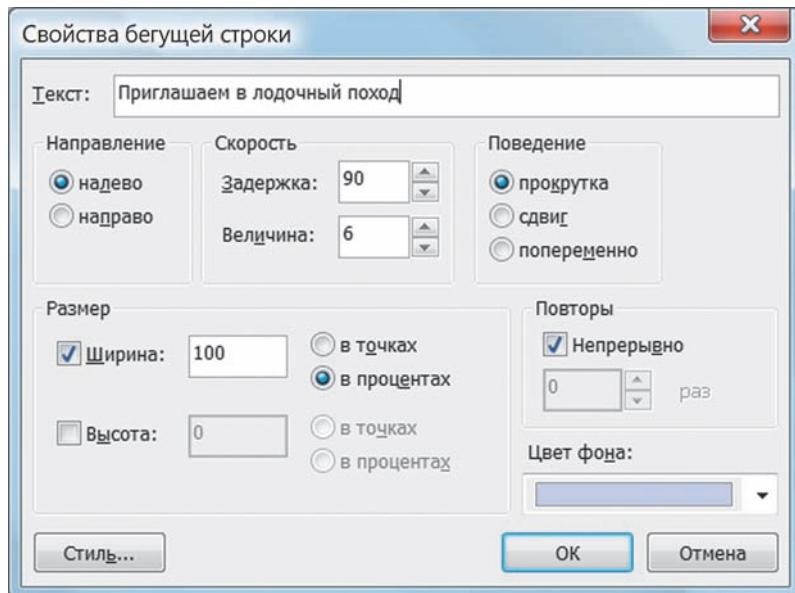


Рис. 1.42

В результате в нижней части страницы появится светло-голубая полоса, по которой справа налево будет перемещаться текст

Приглашаем в лодочный поход



1. Каково назначение редактора FrontPage?
2. Какие возможности форматирования текста имеет FrontPage?
3. Как задается размер шрифта на веб-страницах?

Упражнение

Создайте одну из веб-страниц: Реки Беларуси, Заповедники, Парки.

Используйте фотографии и тексты из указанных учителем файлов. Оформите страницу в соответствии с образцом на рисунке 1.38.

5.3. Использование таблиц

Размещать текст в несколько колонок, а также располагать рисунки и текст в требуемых местах веб-страниц удобно с помощью таблиц.

Для вставки таблицы проще всего использовать стандартную панель инструментов, однако наиболее полно возможности работы с таблицами представлены в меню **Таблица**. Например, в диалоговых окнах **Вставка таблицы** и **Свойства**

таблицы можно указать не только количество строк и столбцов таблицы, но и способ выравнивания, цвет фона ячеек, цвет и толщину границ, интервал между ячейками и другие параметры.

Заметим, что изменить параметры можно и после создания таблицы. В этом случае удобно пользоваться панелью **Таблицы** (рис. 1.43), которая вызывается с помощью команды **Вид → Панели инструментов**.



Рис. 1.43

Рассмотрим на примере, как используются таблицы для размещения объектов на веб-страницах.

Пример 1. Создать веб-страницу «Города Беларуси», на которой изображена карта Беларуси с гербами городов и указаны даты их основания (рис. 1.44).

Откроем редактор MS FrontPage. С помощью кнопки на панели инструментов или меню **Таблица** вставим таблицу из четырех строк и трех столбцов.

The screenshot shows a Microsoft FrontPage document titled 'goroda.htm'. The page contains a large green map of Belarus with its cities highlighted. Overlaid on the map is a 4x3 table structure. The first column contains three rows with city names and founding dates: 'МИНСК 1067', 'ГРОДНО 1127', and 'БРЕСТ 1019'. The second column contains two rows: 'ГРОДНО' with its coat of arms and 'БРЕСТ' with its coat of arms. The third column contains three rows with city names and founding dates: 'ВИТЕБСК 974', 'МОГИЛЕВ 1267', and 'ГОМЕЛЬ 1142'. In the top left corner is the coat of arms of Belarus, and in the top right corner is the flag of Belarus. The bottom of the screen shows the FrontPage ribbon with tabs 'Конструктор' (Designer), 'С разделением' (Split View), 'Код' (Code), and 'Просмотр' (Preview).

МИНСК 1067	ГРОДНО 1127	ВИТЕБСК 974
ГРОДНО	БРЕСТ	МОГИЛЕВ 1267
БРЕСТ 1019	БРЕСТ	ГОМЕЛЬ 1142

Рис. 1.44

В средней ячейке первой строки наберем заголовок «Города Беларуси». Оставшиеся ячейки второго столбца объединим. Для этого выделим объединяемые ячейки и выполним команду **Таблица → Объединить ячейки**. В полученную ячейку вставим изображение карты Беларуси из файла belarus2.jpg.

В ячейки 1 и 3 первой строки вставим изображения герба и флага Беларуси. В остальные ячейки введем названия городов и даты их основания. Отформатируем надписи и установим размеры ячеек в соответствии с рисунком 1.44. Проще всего изменять ширину столбцов, перетаскивая их границы с помощью мыши.

Сделаем невидимыми границы ячеек. Выделим таблицу и вызовем диалоговое окно **Свойства таблицы** (рис. 1.45). В поле **Положение** установим **Выравнивание: по центру**, и в поле **Границы — Размер: 0**. Такие границы в редакторе

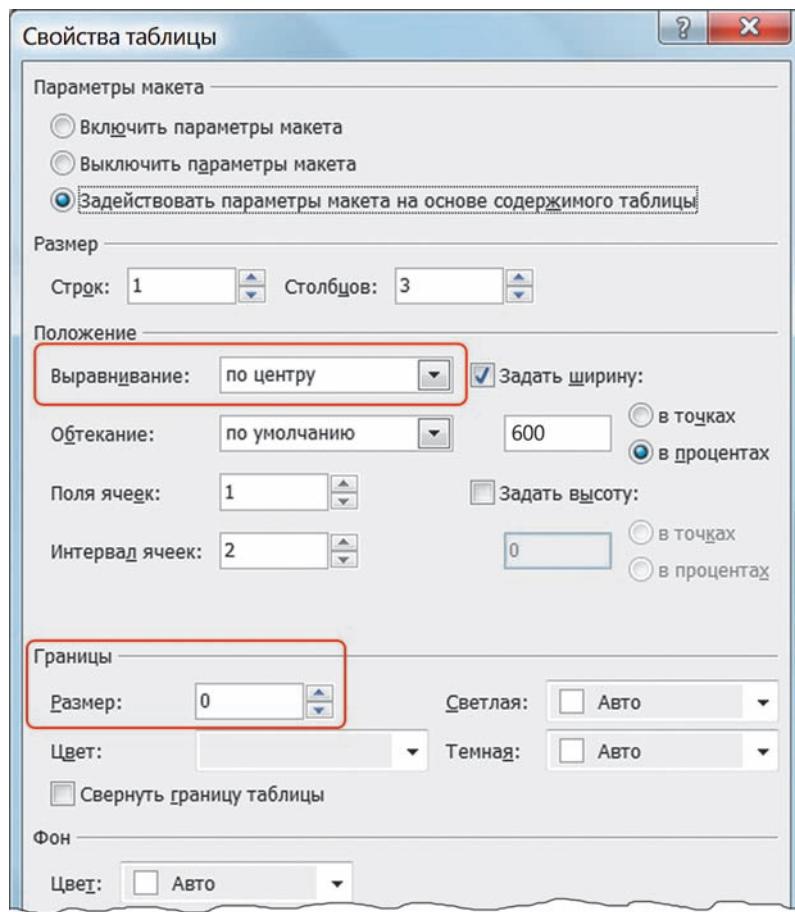


Рис. 1.45

FrontPage отмечаются пунктирными линиями, а при просмотре веб-страницы в браузере не видны.

Теперь страница будет выглядеть так, как на рисунке 1.44. Сохраним ее под именем goroda.htm и просмотрим в браузере.



Для построения таблицы в языке HTML используются теги `<table>` — таблица; `<tr>` — строка таблицы; `<td>` — ячейка таблицы.

В этих тегах можно использовать атрибуты `width` — ширина таблицы или ячейки, `bgcolor` — цвет фона таблицы или ячеек, `border` — толщина рамки и `bordercolor` — цвет рамки и другие (см. Приложение 1).

Ширина таблицы может быть задана в пикселях или в процентах по отношению к ширине страницы, например `width=450` или `width=40%`.

По умолчанию при отображении таблицы создается эффект выпуклости границы. Если задать атрибуты `border=1` и `bordercolor=green`, то эффект выпуклости исчезнет и таблица будет ограничена двойной рамкой зеленого цвета. Атрибут `cellspacing=0` задает одинарную рамку.



Пример 2. В окне кода редактора FrontPage создать таблицу из четырех столбцов и одной строки шириной 600 пикселей с рамкой зеленого цвета толщиной 1 пиксель. Разместить в ячейках изображения (рис. 1.46).

The screenshot shows the Microsoft FrontPage interface. At the top, there's a title bar with the file name "tabl2.htm". Below it is a toolbar with icons for various operations. The main area is divided into two sections: the "Конструктор" (Designer) view at the top and the "Код" (Code) view at the bottom. In the Designer view, there is a table with four columns. Each column contains a different image: a red and white patterned ornament, a close-up of a daisy flower, a blue cornflower, and another red and white patterned ornament. The table has a green border. In the Code view, the HTML code for this table is displayed:

```
1 <html>
2 <head> <title>таблица 2</title> </head>
3 <body>
4   <table width="600" border="1" bordercolor=green>
5     <tr>
6       <td><img src = ornament.gif></td>
7       <td><img src = romashka.jpg></td>
8       <td><img src = vasilek.jpg></td>
9       <td><img src = ornament.gif></td>
10      </tr>
11    </table>
12  </body>
13 </html>
```

Рис. 1.46

Откроем редактор MS FrontPage. Перейдем в режим **С разделением**. В верхнем окне наберем HTML-код. В нижнем окне будет отображен результат (см. рис. 1.46). Сохраним документ под именем tabl2.htm. Просмотрим веб-страницу в браузере.



1. Для чего используются таблицы на веб-страницах?
2. Как вставить таблицу в редакторе FrontPage?

Упражнения

1. Создайте веб-страницу «Разложение белого света в спектр».

Для этого используйте таблицу из четырех столбцов и семи строк. В объединенные ячейки левого и правого столбцов вставьте изображения призмы и радуги из файлов *prizma.jpg* и *raduga.jpg*. В ячейки второго столбца поместите названия цветов и залейте их соответствующим фоном. В ячейках третьего столбца расположите mnemonicеское правило следования цветов спектра.

красный	каждый
оранжевый	охотник
желтый	желает
зеленый	знать
голубой	где
синий	сидит
фиолетовый	фазан

2. Создайте веб-страницу по одной из тем: Природа родного края. Растения. Животные. Птицы. Рыбы. Дары леса. Грибы. Ягоды. Наш сад. Овощи. Фрукты. Цветы.

Тексты и изображения из файлов разместите в таблице из 4—6 ячеек. В качестве образца оформления используйте веб-страницы из примеров 1 и 2.

5.4. Разработка веб-сайта

Рассмотрим особенности разработки веб-сайта в редакторе FrontPage на примере сайта «Песняры беларускай зямлі».

Этап проектирования сайта не отличается от описанного ранее. Создадим сайт из четырех страниц (главной и трех персональных: «Якуб Колас», «Янка Купала» и «Максім Багдановіч»). Все страницы сайта будем сохранять в файлах с соответствующими именами в одной папке, например *risptep*. Структуру сайта изобразим в виде двухуровневой схемы (рис. 1.47).

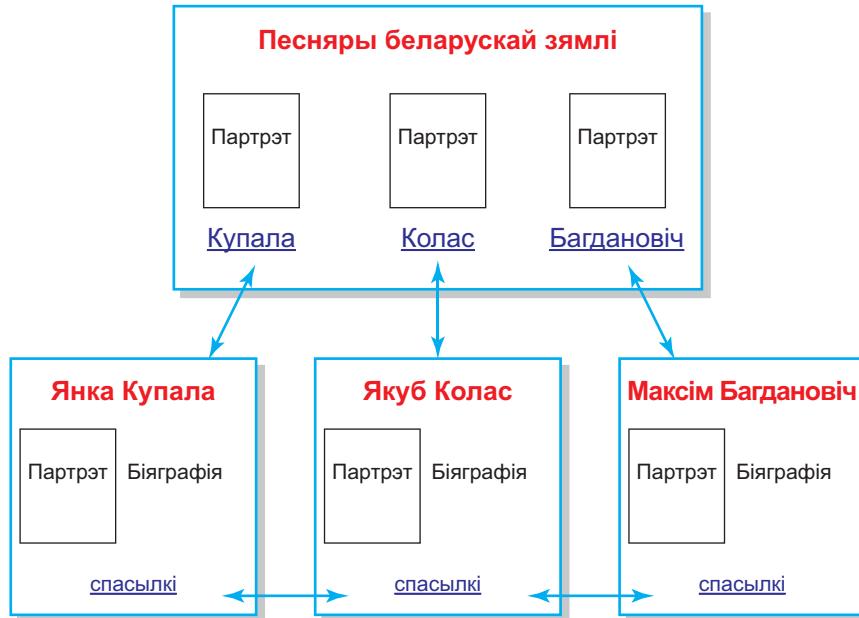


Рис. 1.47

На главной странице разместим заголовок «Песніры беларускай зямлі» и гиперссылки, позволяющие открывать страницы второго уровня. В качестве гиперссылок будем использовать не только текст (например, фамилии), но и изображения (например, небольшие портреты).

Каждая персональная страница будет содержать текстовую (биографические сведения) и графическую информацию (портреты, фотографии памятных мест). Внизу расположим гиперссылку, обеспечивающую возврат на главную страницу сайта, и две ссылки для перехода с одной персональной страницы на другую.

Заранее подготовим все изображения. Поместим их во вложенную папку *Ris/*.

Непосредственное конструирование начнем с персональных страниц, которые оформим в едином стиле.

Напомним, что для облегчения подбора элементов оформления можно использовать готовый шаблон оформления. В редакторе FrontPage предлагается несколько шаблонов — **Тем**. Тему можно применять к отдельным страницам, а также ко всему сайту. В последнем случае при создании каждой новой страницы тема устанавливается автоматически.

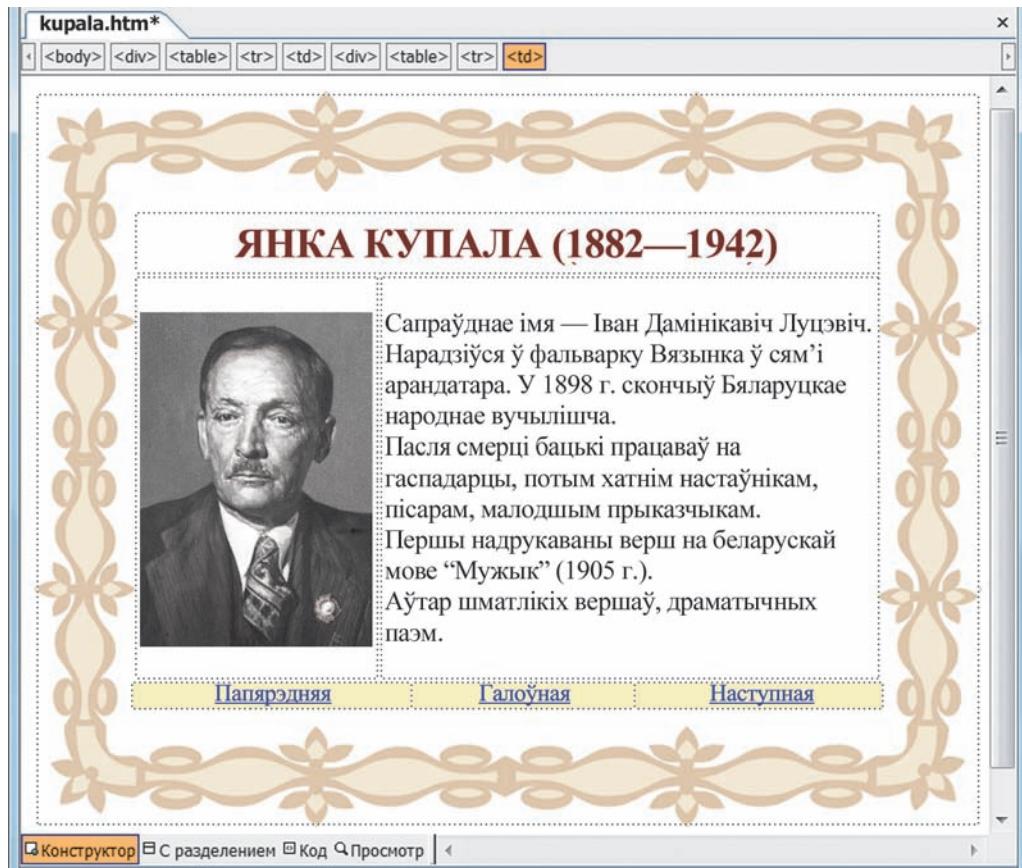


Рис. 1.48

Пример 1. Создать веб-страницу «Янка Купала» (рис. 1.48).

Запустим редактор FrontPage. Откроется пустая страница с установленными по умолчанию параметрами. Проверим кодировку (**Windows-1251 — Кириллица**).

Цвет фона или фоновый рисунок всей страницы задаются в диалоговом окне **Свойства страницы** на вкладке **Форматирование**, которая вызывается с помощью меню **Формат** → **Фон**. Мы создадим свой шаблон оформления страниц на основе изображения рамки. Используем его в качестве фона таблицы. Вставим таблицу из одной ячейки. В диалоговом окне **Свойства таблицы** зададим ее ширину (720 пикселей) и высоту (560 пикселей) в соответствии с размерами рамки, установим флажок **Использовать фоновый рисунок** и укажем имя файла ramka.gif. В эту таблицу вставим еще три таблицы шириной 80 %: из одной

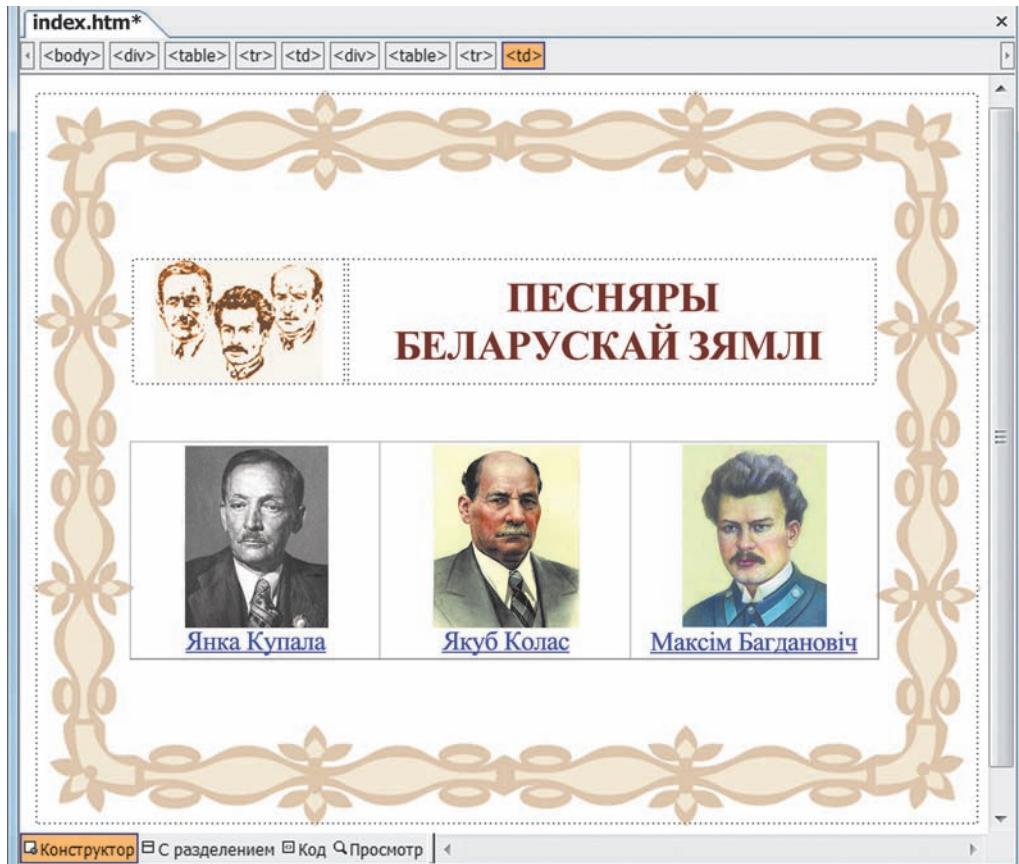


Рис. 1.49

ячейки — для размещения заголовка, из двух — для рисунка и биографии, и из трех — для текстовых гиперссылок.

Оформим страницу в соответствии с рисунком 1.48: скопируем заранее подготовленный текст из файла biograf.txt, вставим рисунок из файла kupala.jpg. Заметим, что можно использовать и одну таблицу с объединенными ячейками, но в ней труднее размещать объекты в требуемых местах. Сохраним страницу под именем Kupala.htm в папке pismen.

Аналогично оформим персональные страницы «Якуб Колас» (файл Kolas.htm) и «Максім Багдановіч» (файл Bagdanovich.htm). Главную страницу (файл index.htm) оформим в соответствии с рисунком 1.49.

Напомним, что сайт представляет собой совокупность веб-страниц, связанных гиперссылками. Для создания гиперссылки требуется выделить текст или ри-

сунок и задать адрес перехода (адрес веб-страницы, имя файла или документа) по данной ссылке. В качестве гиперссылок могут выступать слова, группы слов, изображения.

Текстовые ссылки обычно выделяются цветом и подчеркиванием, а графические — рамкой. Благодаря этому они отличаются от обычного текста и других элементов страницы. По умолчанию ссылки на непросмотренные страницы отображаются синим цветом, а на просмотренные — фиолетовым.

Пример 2. Создать текстовые и графические гиперссылки между страницами веб-сайта «Песняры беларускай зямлі».

Для создания гиперссылки выделим портрет Янки Купалы и подпись. С помощью кнопки **Добавление гиперссылки** на стандартной панели инструментов или команды **Вставка** → **Гиперссылка** вызовем диалоговое окно, в котором укажем адрес перехода Kupala.htm (рис. 1.50). Аналогичным образом создадим все ссылки в соответствии со схемой навигации по сайту (см. рис. 1.47).

Вызванная щелчком мыши по ссылке страница по умолчанию открывается в текущем окне браузера. Можно задать ее открытие в новом окне браузера. Для этого в диалоговом окне **Добавление гиперссылки** щелкают по кнопке **Выбор рамки** и в открывшемся окне **Конечная рамка** выбирают вариант **Новое окно**.

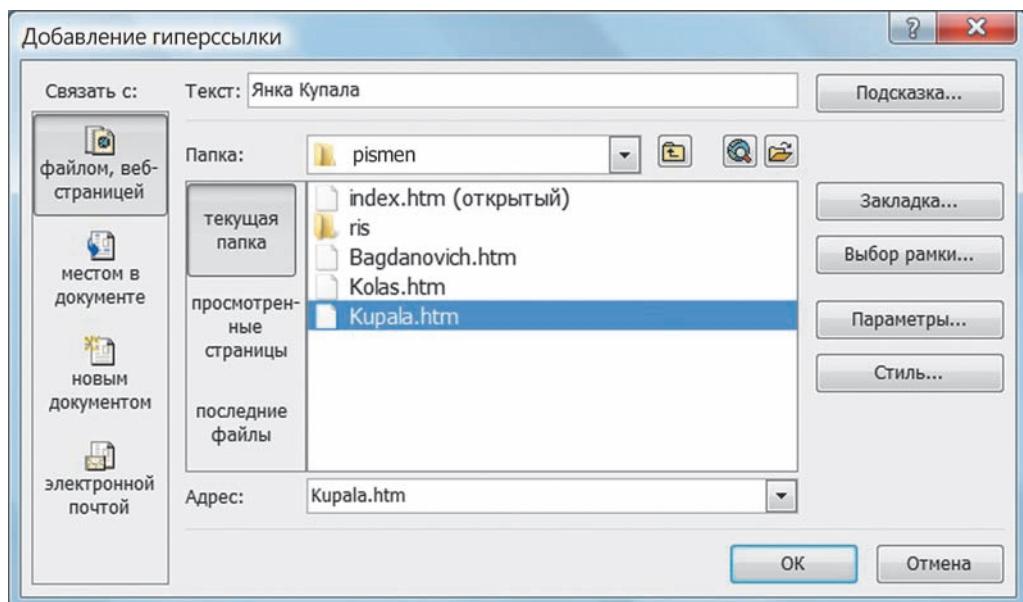


Рис. 1.50

Заметим, что в редакторе FrontPage создание гиперссылок выполняется в режиме работы **Конструктор**, а проверить работу гиперссылок можно в режиме **Просмотр**.

Ссылки могут обеспечивать переход к веб-страницам или иным документам, например рисункам, презентациям, видеофрагментам, расположенным как на данном сайте (*внутренние ссылки*), так и на других сайтах (*внешние ссылки*). Переходы внутри документа выполняют *внутристричные* гиперссылки. Они облегчают навигацию по странице, например быстрый переход из конца страницы в начало.



Перед созданием внутристричных ссылок нужно сначала расставить **закладки**. Для этого курсор мыши устанавливают в том месте страницы, куда предполагается переход по ссылке. Затем с помощью команды **Вставка** → **Закладка** вызывают диалоговое окно **Закладка**, в котором вводят имя закладки, например **начало** (рис. 1.51). Далее в окне **Добавление ссылки** выбирают вариант **Связать с местом в документе** и указывают требуемую закладку (рис. 1.52).

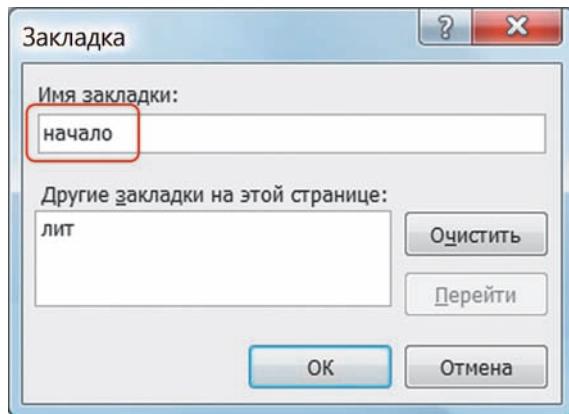


Рис. 1.51

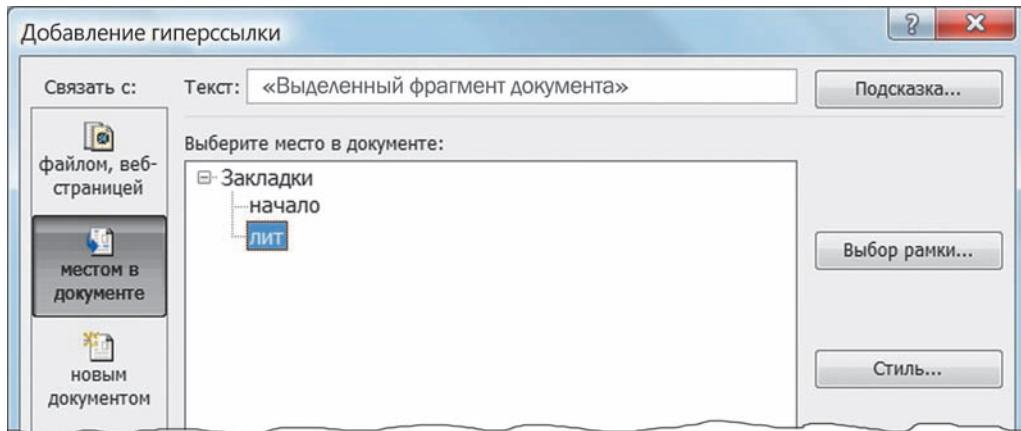


Рис. 1.52



В качестве гиперссылок на веб-страницах нередко используют кнопки, которые при наведении на них курсора могут изменять свои цвет и форму. В редакторе FrontPage предусмотрена возможность размещения таких кнопок. Для этого в меню **Вставка** → **Веб-компонент** выбирают компонент **Интерактивная кнопка**. Затем в появившемся диалоговом окне **Меняющиеся кнопки** выбирают кнопку нужной формы и задают ее параметры и эффекты.



1. Какие элементы могут выступать в качестве гиперссылок?
2. На какие объекты может указывать гиперссылка?

Упражнение

1. Создайте фрагмент веб-сайта «Галерея ученых» (писателей, спортсменов, музыкантов и т. п.), оформив его по приведенным образцам. Гиперссылки на главной странице представьте в виде миниатюр с подписями внизу.

	НИКОЛАЙ КОПЕРНИК (1473—1543)
---	-------------------------------------

<p>Николай Коперник</p> <p>первым предложил гелиоцентрическую модель Солнечной системы, согласно которой в центре находится Солнце, а Земля и другие планеты движутся вокруг по замкнутым орбитам, а не держатся на невидимых кристаллических сферах. Запрещал публикацию своих работ до смерти.</p>	
---	---

 К списку ученых	 Назад	 Вперед
---	---	--

 Аристотель	 Коперник	 Галилей	 Ньютона
---	---	--	---

5.5. Публикация сайта

Публикацией сайта называют его размещение на сервере или локальном компьютере с возможностью вызова из сети (глобальной или локальной).

Процесс публикации сайта заключается в переносе файлов сайта на сервер. Адрес перенесенного сайта может быть, например, таким: www.poets.webserver.by.

Для публикации сайта, подготовленного в редакторе FrontPage, необходимо с помощью команды **Файл → Опубликовать узел** вызвать диалоговое окно **Свойства удаленного веб-узла**.

Для размещения созданного сайта на сервере в строке **Расположение удаленного веб-узла** следует указать адрес ftp-узла, который для всех пользователей является одинаковым, и нажать кнопку **OK**.

В появившемся окне **Требуются имя и пароль** вводят имя и пароль, которые понадобятся в дальнейшем для обновления, добавления или удаления размещенных на сайте материалов.

Нажатие кнопки **OK** открывает окно, на левой панели которого (**Локальный веб-узел**) (рис. 1.53) отображаются файлы и папки, содержащие страницы и ри-

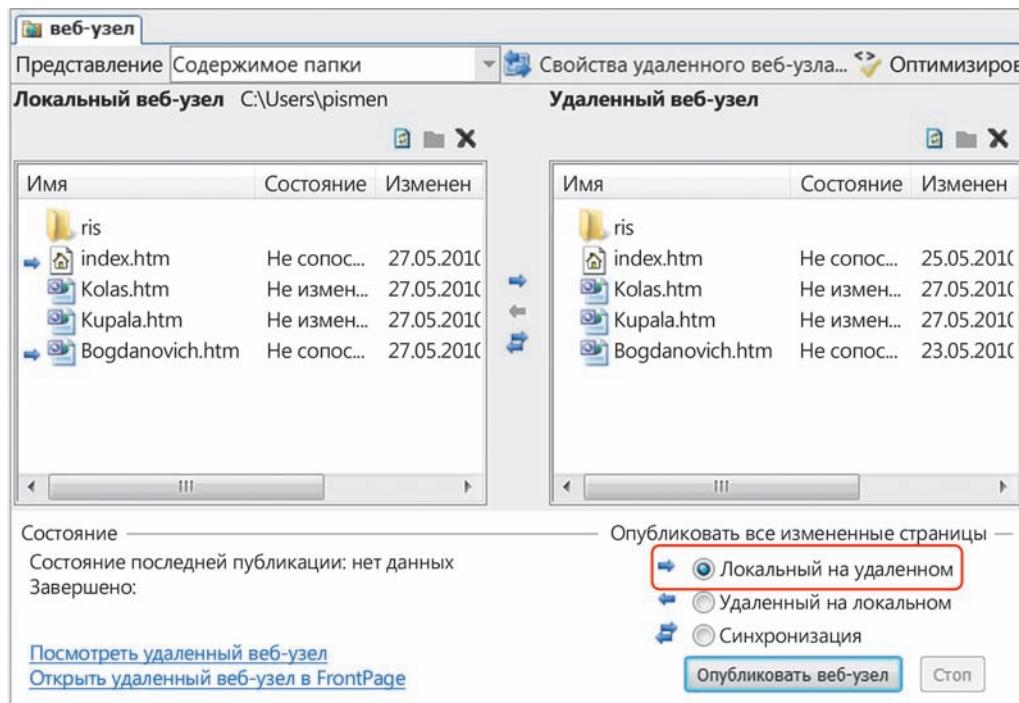


Рис. 1.53

сунки созданного фрагмента веб-сайта. На правой панели этого окна (**Удаленный веб-узел**) отображаются файлы и папки, уже размещенные на web-сервере, например страницы сайта «Песняры беларускай зямлі» (см. рис. 1.53).

Если публикация выполняется впервые, правая панель пуста. Выберем требуемый режим, например **Все файлы локального веб-узла копировать на сервер**, и нажмем кнопку **Опубликовать веб-узел**.

Как только копирование всех файлов завершится, пользователь сможет просмотреть размещенный в сети сайт. Для этого в строке браузера вводится адрес, например www.poets.webserver.by.

- ?**
1. Что понимают под публикацией сайта?
 2. Какие действия надо выполнить для публикации сайта, подготовленного в редакторе FrontPage?

§ 6. Создание фрагментов сайтов по различным предметным областям

Изучив этапы разработки веб-сайтов и основные приемы работы в редакторе Front Page, мы можем создавать сайты, посвященные различным предметным областям.

Пример. Создать фрагмент веб-сайта «Замки Беларуси» (рис. 1.54, 1.55).



Рис. 1.54



Рис. 1.55

Создадим фрагмент сайта из пяти страниц: главной страницы с маленькими картинками и четырех страниц, посвященных замкам. Все страницы будем сохранять в файлах с соответствующими именами в одной папке, например *zamki*. Заранее подготовленные изображения поместим во вложенную папку *zamki /ris/*.

Оформим страницы в одном стиле в соответствии с рисунком 1.55. Вставим таблицу с одной ячейкой. Установим ширину 640 и высоту 480 пикселей. В качестве фона таблицы используем изображение свитка из файла *svitok.gif*.

Вставим еще одну таблицу из одного столбца и четырех строк. В первой строке разместим заголовок. Во вторую строку вставим рисунок, например из файла *nesvizh.jpg*, в третью — поместим текст из файла *zamki.txt*. В четвертой строке разместим разделительную линию, тексты ссылок и изображения стрелок из файлов или **Коллекции клипов**.

Главную страницу оформим в соответствии с рисунком 1.54. В таблице из двух столбцов и двух строк разместим используемые в качестве гиперссылок картинки с подписями. Создадим гиперссылки для переходов со страницы на страницу.

Просмотрим страницы в браузере. Проверим работу гиперссылок.

Упражнение

Создайте фрагмент сайта (не менее 3—4 страниц, содержащих текст и изображения) по одной из тем учебных предметов.

Беларуская літаратура. Жыццё Ефрасінні Полацкай. Жыццё і творчасць Кірылы Тураўскага. Францыск Скарына — першадрукар і асветнік. Мікола Гусоўскі. Песня пра зубра. Адам Міцкевіч. Францішак Багушэвіч. Қандрат Крапіва. Васіль Быкаў. Беларуская паэзія XX стагоддзя.

Русская литература. А. С. Пушкин. М. Ю. Лермонтов. Н. В. Гоголь. Н. А. Некрасов. А. Н. Островский. И. С. Тургенев. Ф. М. Достоевский. Л. Н. Толстой. А. П. Чехов. М. Горький. Серебряный век русской культуры. Русская поэзия XX века.

Всемирная история. Первая мировая война. Октябрьская революция. Образование СССР. Начало Великой Отечественной войны. Блокада Ленинграда. Сталинградская битва. Курская дуга. Партизанское движение. Операция «Багратион». Капитуляция Германии. Основание ООН. Распад СССР. Образование СНГ.

Гісторыя Беларусі. Грунвальдская бітва. Першая сусветная вайна на беларускіх землях. Гады грамадзянскай вайны. Вялікая Айчынная вайна. Партызанскі рух. Вызваленне Беларусі. Помнікі гісторыі. Стварэнне Рэспублікі Беларусь.

География. Великие географические открытия. Страны и континенты (Природные условия. Погода и климат. Население. Промышленность. Сельское хозяйство). Геоэкологические проблемы. Изменение климата. Загрязнение окружающей среды. Охрана природы.

Биология. История открытия клетки. Строение клетки. Обмен веществ. Фотосинтез. Экосистема. Происхождение жизни. Биологическая эволюция. Происхождение и эволюция человека. Биосфера — живая оболочка планеты. Человек и окружающая среда. Влияние различных факторов среды на человека.

Математика. Графики функций. Тригонометрические функции. Показательная и логарифмическая функции. Правильные многоугольники. Пространственные фигуры. Многогранники и их изображения. Тела вращения.

Физика. Электромагнитная индукция. Гармонические колебания. Маятники. Волны. Звук. Электромагнитные колебания. Производство и передача электрической энергии. Электромагнитные волны. Шкала электромагнитных волн. Интерференция. Дифракция. Законы преломления. Оптические приборы.

Химия. Периодическая система химических элементов Д. И. Менделеева. Атомы и молекулы. Неорганические соединения. Кислород. Сера. Азот. Фосфор. Углерод. Кремний. Металлы. Химическое строение органических соединений. Углеводороды. Переработка нефти. Синтетические моющие средства. Углеводы. Азотсодержащие органические соединения. Аминокислоты. Синтетические высокомолекулярные соединения. Применение полимеров. Белки.

ГЛАВА 2

ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

В этой главе рассмотрены примеры решения задач из различных предметных областей.

Решение любой задачи с помощью компьютера содержит несколько этапов: анализ исходных данных и возможных результатов, построение математической модели изучаемого объекта, выбор метода решения, составление алгоритма решения, написание и отладка программы, тестирование программы.

Особый интерес представляет решение задач с использованием графических возможностей языка программирования. Графические элементы улучшают наглядность и выразительность представляемой информации в любой предметной области.

§ 7. Выполнение практических заданий из различных предметных областей

7.1. Построение геометрических фигур

Мы уже умеем работать с растровой и векторной графикой в графических редакторах, а также программировать рисование простых изображений. Применим наши умения для построения геометрических фигур.

Начнем с чертежа простой детали. Чертеж является основным документом любого производства. По чертежам изготавливают детали, собирают машины, строят дома, шьют одежду. На уроках черчения Вы учились выполнять чертежи на бумаге. Современное же производство немыслимо без компьютерных систем автоматического проектирования (например, AutoCAD, Кompas).

Напомним, что в среде программирования PascalABC изображения можно формировать из простых геометрических фигур (графических примитивов), например точек, отрезков, прямоугольников, окружностей, эллипсов. Рисование осуществляется в графическом окне. Начало системы координат находится в левом верхнем углу окна, ось абсцисс направлена вправо, а ось ординат — вниз. Библиотека стандартных графических процедур хранится в модуле GraphABC (Приложение 3).

При работе с изображениями в среде PascalABC удобно все данные вводить и выводить в графическом окне. Совмещать работу с текстом и графикой в одном окне можно, подключив модули crt и GraphABC одновременно.

Пример 1. Составить программу, которая выполняет чертеж детали (рис. 2.1) и вычисляет ее площадь. Масштаб: 1 клетка — 20 пикселей. Радиус отверстия r в сантиметрах вводится с клавиатуры, $AB = BC = 8$ см.

Определим исходные данные. Радиус отверстия будем вводить в переменную r , размер AB и площадь фигуры будем хранить в переменных a и S . Все переменные будут иметь тип `real`.

Чертеж данной детали можно выполнить рисованием пяти графических примитивов: отрезков AB , BC , CD , окружности с центром O и дуги AED .

Определим экранные координаты точек. Пусть точка O имеет координаты (160, 120) (см. рис. 2.1). Тогда с учетом масштаба координаты остальных точек следующие: $A(80, 120)$, $B(80, 280)$, $C(240, 280)$, $D(240, 120)$. Радиус дуги AED равен 80. Радиус окружности на экране вычислим умножением введенного значения r на масштаб. Дробную часть отбросим, чтобы обеспечить целочисленность координат для процедуры рисования: `trunc(r*20)`.

Площадь детали складывается из площади квадрата $ABCD$ со стороной a и площади полукруга AED диаметром a за вычетом площади круга радиусом r .

Программа может выглядеть так:

```
program Figural1;
uses crt, GraphABC;                                {Подключение модулей}
var r, a, S: real;
begin
  SetWindowSize(320,320);                           {Размеры окна}
  write('Введите r (от 1 до 3) ');
  read(r);                                         {Ввод радиуса}
  SetPenWidth(3);                                  {Толщина пера}
  line(80,120, 80,280);                           {Рисование отрезков}
  line(80,280, 240,280);
  line(240,280, 240,120);
  circle(160,120, trunc(r*20));                  {Рисование окружности}
  arc(160,120, 80, 0,180);                        {Рисование дуги}
  a:=8; S:=a*a+Pi*a*a/8-Pi*r*r;                 {Вычисление площади}
  write('S = ', S:2:2);                           {Форматный вывод}
end.
```

Результат работы программы для $r = 2,4$ представлен на рисунке 2.2.

Конечно, рисовать подобные фигуры можно в любом из изученных Вами графических редакторов и даже с помощью инструментов приложений MS Office. Программирование необходимо в тех случаях, когда требуется изменять свойст-

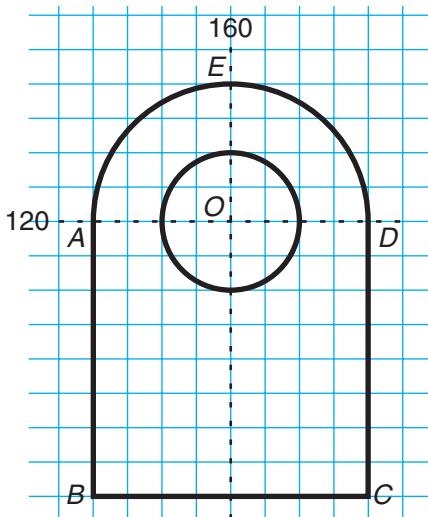


Рис. 2.1

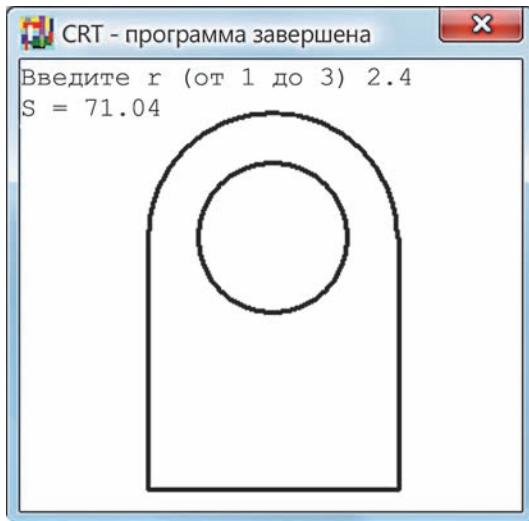


Рис. 2.2

ва объектов или производить вычисления. Так, в нашем примере с клавиатуры вводятся значения радиуса отверстия и вычисляется площадь детали. Нетрудно, например, вычислить еще и массу детали, задав толщину и плотность материала.



На уроках математики Вы научились изображать пространственные фигуры: призму, пирамиду, цилиндр, конус, шар. Рассмотрим построение пространственных фигур в среде PascalABC.



Пример 2. Составить программу, которая рисует цилиндр (рис. 2.3) и вычисляет его объем. Масштаб: 1 клетка — 20 пикселей. Радиус цилиндра — 4 см. Высоту цилиндра в сантиметрах вводится с клавиатуры.

Определим исходные данные. Высоту цилиндра будем вводить в переменную h , радиус и объем будем хранить в переменных r и V . Переменные h и r будут иметь тип `integer`, а V — тип `real`.

При построении пространственных фигур окружности, не лежащие в плоскости чертежа, изображаются эллипсами. Таким образом, построение цилиндра сводится к рисованию четырех графических примитивов: двух отрезков AB , CD и двух эллипсов с центрами O_1 и O_2 .

Заметим, что в задачах с построением фигур следует сначала вычислять требуемые величины, используя заданные и введенные значения, и лишь затем производить переход к экранным координатам с учетом масштаба.

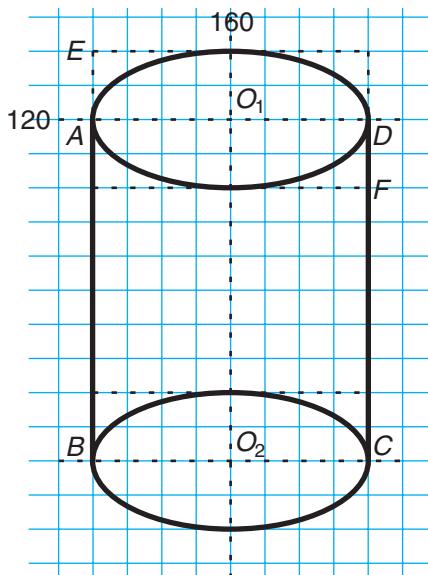


Рис. 2.3

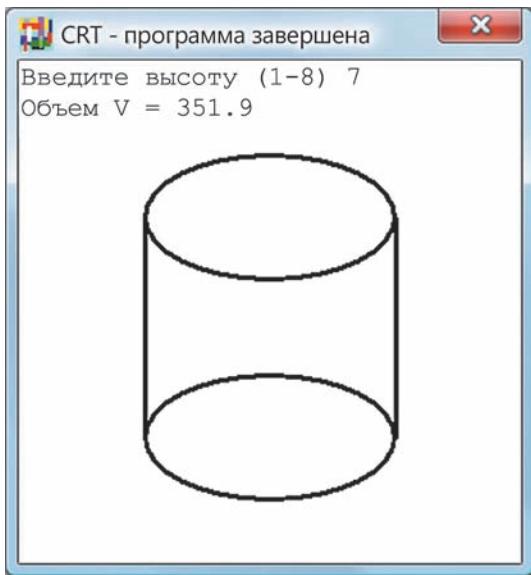


Рис. 2.4

Вычислим объем цилиндра, как произведение площади основания (круга) и высоты $V = \pi r^2 h$.

Определим экранные координаты точек. Пусть точка O_1 имеет координаты $x = 160$, $y = 120$. Произведем пересчет радиуса и высоты с учетом масштабов: $r = 20r$ и $h = 20h$. Поскольку масштабы учтены, координаты точек выражаются простыми формулами: $O_2(x, y + h)$, $A(x - r, y)$, $B(x - r, y + h)$, $C(x + r, y + h)$, $D(x + r, y)$.

Наконец, нарисуем отрезки и эллипсы. Процедура `Ellipse` рисует эллипс, который задается описанным около него прямоугольником с координатами противоположных вершин, например: $E(x - r, y - 40)$ и $F(x + r, y + 40)$.

Программа может выглядеть так:

```
program Figura2;
uses crt, GraphABC;
var x, y, h, r: integer; V: real;
begin
  SetWindowSize(320,320);           {Размеры окна}
  write('Введите высоту (1-8) ');
  readln(h);                      {Ввод высоты}
  r:=4; V:=Pi*r*r*h;              {Вычисление}
```

```

write('Объем V = ', V:2:1);           {Вывод}
SetPenWidth(3);                      {Толщина пера}
SetBrushStyle(bsClear);               {Стиль пера}
x:=160; y:=100;                      {Экранные координаты}
r:=20*r; h:=20*h;                    {Учет масштаба}
Ellipse(x-r,y-40, x+r,y+40);        {Рисование эллипсов}
Ellipse(x-r,y+h-40, x+r,y+h+40);
line(x-r,y, x-r,y+h);               {Рисование отрезков}
line(x+r,y, x+r,y+h);
end.

```

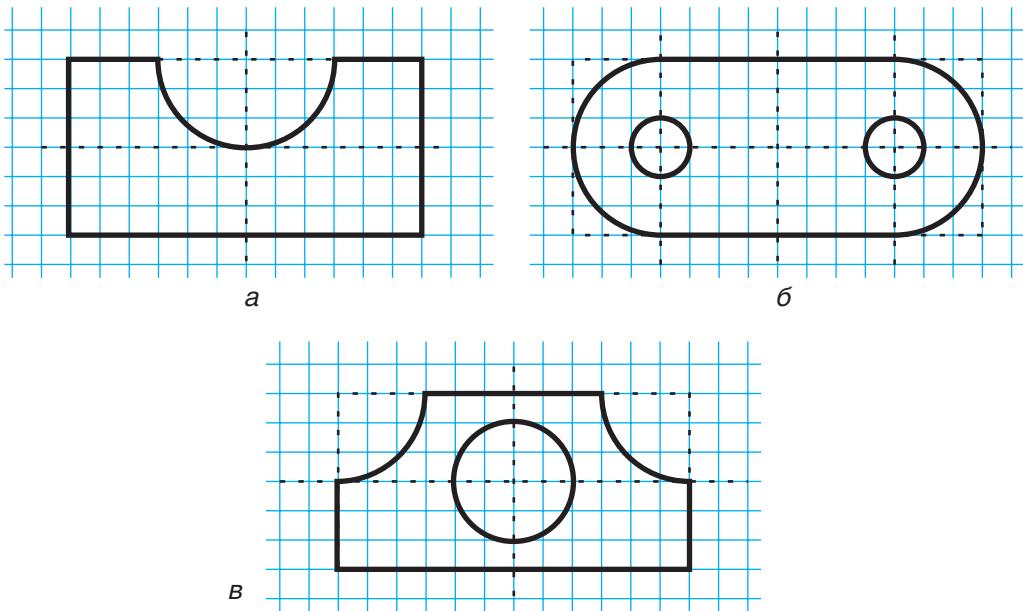
Результат работы программы для $h = 7$ представлен на рисунке 2.4.



1. Какие графические примитивы можно использовать при построении изображений?
2. Какие модули необходимо подключать, чтобы работать с текстом и графикой в одном окне?

Упражнения

1. Составьте программу, которая выполняет чертеж одной из деталей и вычисляет ее площадь. Масштаб: 1 клетка — 20 пикселей.





2. Составьте программу, которая рисует одну из пространственных фигур:

а) куб (длина ребра вводится с клавиатуры);

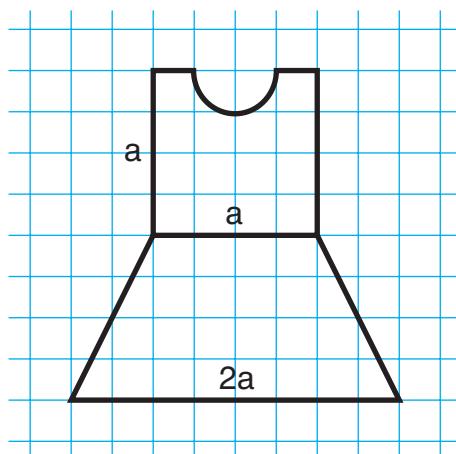
б) конус (значения радиуса и высоты вводятся с клавиатуры).



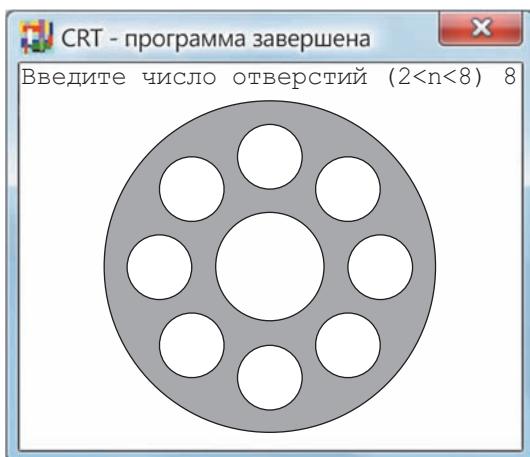
3. Составьте программу, которая рисует:

а) выкройку платья (размер a вводится с клавиатуры);

б) деталь в виде кольца внешним радиусом 150 и внутренним радиусом 50, в котором вырезаны отверстия радиусом 30. Их центры находятся на расстоянии 100 от центра кольца (число отверстий n вводится с клавиатуры).



a



b

7.2. Использование растровых изображений

Графические возможности языка программирования расширяет использование растровых изображений. Особенно интересно использование фотографий реальных объектов и явлений живой и неживой природы (машин, механизмов, людей, животных, растений), которые практически невозможно создать с помощью стандартных графических примитивов.

Процедуры и функции для работы с растровыми изображениями содержатся в модуле *GraphABC*.

Функция `LoadPicture(fname)` загружает рисунок из файла с именем `fname` в оперативную память. Загружать можно рисунки в форматах BMP, JPG, GIF, PNG. Каждому загруженному рисунку, хранящемуся во время работы программы в оперативной памяти, присваивается целочисленный описатель (дескриптор) `nm := LoadPicture(fname)`. Он передается в качестве первого параметра во все остальные процедуры и функции работы с рисунками.

Процедура `DrawPicture(nm,x,y)` выводит рисунок с описателем `nm` в позицию `(x,y)` графического окна, а процедура `DrawPicture(nm,x,y,w,h)` еще и масштабирует изображение, устанавливая его ширину `w` и высоту `h`. Если `w<0`, то рисунок зеркально отражается относительно вертикальной прямой, проходящей через `x`, если же `h<0`, то рисунок зеркально отражается относительно горизонтальной прямой, проходящей через `y`.

Пример 1. Составить программу, иллюстрирующую принцип действия проекционного аппарата.

В проекционном аппарате предмет (слайд, или рисунок на прозрачной пленке) помещают между фокусом и двойным фокусом линзы. На экране получается увеличенное обратное изображение. Для получения на экране прямого изображения слайд переворачивают.

Алгоритм программы заключается в загрузке из файла растрового изображения и выводе его на экран после необходимых преобразований (увеличение и переворот). Единственная переменная `pic` будет иметь тип `integer`.

Прежде всего зададим размеры экрана 640×520 и черный цвет фона. Нарисуем «линзу»: `Ellipse(106,220,122,300)`.

Загрузим изображение из файла (например, `robotA.jpg`) и поместим его описатель в переменную `pic:=LoadPicture('robotA.jpg')`.

С помощью процедуры `DrawPicture(pic,60,310,-50,-100)` выведем перевернутое изображение размером 50×100 пикселей в позицию с координатами верхнего левого угла $(60, 310)$, поставив перед значениями 50 и 100 знаки «минус». Еще раз выведем это изображение, но уже в позицию с координатами $(380, 10)$, изменив его ширину до 250 пикселей, а высоту — до 500 пикселей, т. е. увеличив его в 5 раз: `DrawPicture(pic,380,10,250,500)`.

Осталось провести вспомогательные линии, иллюстрирующие построение изображения, и откорректировать положение «линзы».

Программа может выглядеть так:

```
program Proector;
uses GraphABC;
var pic: integer;
begin
  SetWindowSize(640,520);           {Размеры окна}
  ClearWindow(clBlack);           {Черный фон}
  SetBrushColor(clSkyBlue);
  Ellipse(106,220,122,300);       {Линза}
  pic:=LoadPicture('robotA.jpg');  {Загрузка изображения
                                  из файла}
```

```
DrawPicture(pic,60,310,-50,-100); {вывод изображения, переворот}
DrawPicture(pic,380,10,250,500); {увеличение}
SetPenColor(clYellow);           {Вспомогательные линии}
line(60,310, 380,10); line(60,210, 380,510);
SetPenColor(clWhite);
line(10,310, 630,10); line(10,210, 630,510);
end.
```

Результат работы программы представлен на рисунке 2.5.

Растровые изображения выводятся в прямоугольные области, которые при создании графических композиций могут перекрывать друг друга. Процедура SetPictureTransparent(*nm*,*b*) устанавливает режим прозрачности изображения с описателем *nm*. По умолчанию режим прозрачности отключен (*b=False*).

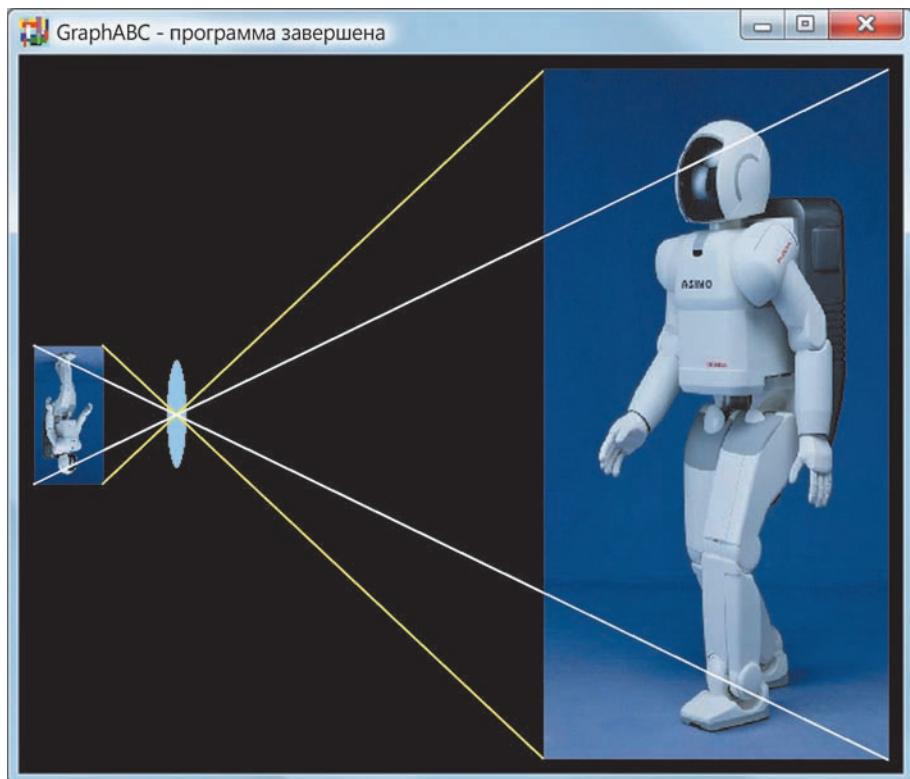


Рис. 2.5

Если `b=True`, то при рисовании фон не отображается. Фоновым считается цвет левого нижнего пикселя рисунка.

Растровые изображения можно использовать в качестве образцов для закрашивания фигур кистью. Процедура `SetBrushPicture(fname)` закрашивает фигуры фоновым рисунком, используя образец, хранящийся в файле `fname`.



В последнее время широкое распространение получили программы, предназначенные для автоматической генерации разнообразных типовых документов, например поздравительных открыток, рекламных плакатов, финансовых документов (накладных, счетов), презентаций. Алгоритмы их работы основаны на использовании шаблонов, которые можно дополнять некоторым содержанием и элементами оформления.



Пример 2. Составить программу, которая рисует поздравительные открытки. С клавиатуры выбирается тема, и вводится текст поздравления.

Будем оформлять все открытки по шаблону: слева — залитый растровым изображением прямоугольник, справа — рисунок сказочного персонажа, сверху — текст поздравления.

Номер темы будет вводиться с клавиатуры (переменная `tema`). Для каждой темы с помощью алгоритмической конструкции выбора `case...of` зададим имена файлов (переменные `fzal`, `fris`), в которых будут храниться подходящие изображения. Например, для темы 1 (зима) используем снежинки (`fzal:='snow.gif'`) как фоновый рисунок и Буратино (`fris:='buratino.gif'`), для темы 2 (весна) — солнышко (`fzal:='sun.gif'`) и Попугая (`fris:='popugai.gif'`), для темы 3 (лето) — ромашки (`fzal:='daisy.gif'`) и Карлсона (`fris:='karlson.gif'`), для темы 4 (осень) — листочки (`fzal:='list.gif'`) и Лису (`fris:='lisa.gif'`).

Затем будет вводиться текст поздравления (переменная `s`), например «С Новым годом!».

Переменные `tema` и `ris` будут иметь тип `integer`, а переменные `fzal`, `fris`, `s` — тип `string`.

Программа может выглядеть так:

```
program Otkrytka;
uses crt, GraphABC;
var tema, ris: integer; fzal, fris, s: string;
begin
  SetWindowSize(600,400);
  write('Введите номер темы: 1-зима, 2-весна, 3-лето,
4-осень'); read(tema);
```

```

write('Введите поздравление '); read(s);
  case tema of
    1: begin fzal:='snow.gif'; fris:='buratino.gif'; end;
    2: begin fzal:='sun.gif'; fris:='popugai.gif'; end;
    3: begin fzal:='daisy.gif'; fris:='karlson.gif'; end;
    4: begin fzal:='list.gif'; fris:='lisa.gif'; end;
  end;
ClearWindow;
SetPenColor(clLtGray);           {Выбор имен файлов}
SetBrushPicture(fzal);          {Цвет пера}
SetBrushStyle(bsClear);          {Установка заливки}
Rectangle(0,0,120,400);          {Рисование прямоугольника}
ris:=LoadPicture(fris);          {Загрузка изображения из файла}
DrawPicture(ris,380,70, 200,320); {Вывод изображения}
SetFontColor(clRed);            {Прозрачный фон кисти}
FontSize(34);                  {Цвет шрифта}
SetFontStyle(fsBold);           {Размер шрифта}
TextOut(60,40,s);               {Начертание}
{Выход текста}
end.

```

Результаты работы программы представлены на рисунке 2.6.

Во многих задачах требуется исследовать протекание процессов и явлений во времени. Для наглядной демонстрации решения таких задач используют методы анимации. Напомним, что **анимацией** называют имитацию движения и



Рис. 2.6

изменения формы и других видимых свойств объектов с течением времени. В 9-м классе мы научились создавать анимации в специальном графическом редакторе.

В среде PascalABC изображения можно «оживить» путем их многократного рисования и стирания. Избежать моргания экрана позволяет использование процедуры перерисовки `Redraw`.

Пример 3. Составить программу, которая имитирует движение автомобиля.

Пусть автомобиль перемещается на фоне здания и деревьев слева направо на расстояние 400 пикселей.

Загрузим изображения фона и автомобиля из файлов `gorod.gif` и `avto.jpg`, поместив их описатели в переменные `fon` и `avto`. Установим прозрачность фона для изображения автомобиля: `SetPictureTransparent(avto,true)`. Зададим начальные координаты (x,y), ширину `w` и высоту `h` изображения автомобиля. Все переменные будут иметь тип `integer`.

Процедуры рисования и стирания будем повторять в цикле с предусловием `While` до тех пор, пока автомобиль не переместится на 400 пикселей. На каждом шагу цикла координату `x` левого верхнего угла изображения будем увеличивать на 10. Ширину изображения будем уменьшать на 2 пикселя, а высоту — на 1 пиксель для его уменьшения при удалении.

Заметим, что плавность и длительность полученной демонстрации зависит от выбора величины шага (увеличения координаты `x`), количества кадров (повторений цикла), времени показа кадра (задержки), а также быстродействия компьютера.

Программа может выглядеть так:

```
program Avto3;
uses GraphABC;
var fon, avto, x, y, w, h: integer;
begin
  SetWindowSize(600,300);
  fon:= LoadPicture('gorod.gif'); {Загрузка изображений}
  avto:= LoadPicture('avto.jpg');
  SetPictureTransparent(avto,true); {Прозрачный фон}
  x:=0; y:=170; w:=240; h:=100;
  While x<400 do
    begin
      ClearWindow; DrawPicture(fon,0,0);
      DrawPicture(avto,x,y,w,h); {Выход изображений}
```

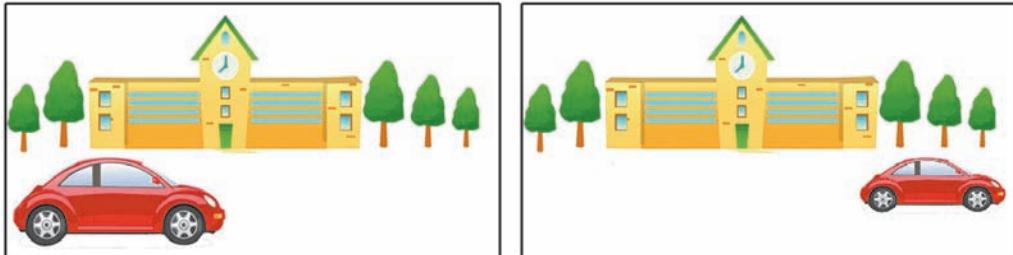


Рис. 2.7

```

x:=x+10; w:=w-2; h:=h-1;
sleep(20); {Задержка на 20 мс}
Redraw; {Перерисовка}
end;
end.

```

Первый и последний кадры «анимации» (начальное и конечное положения автомобиля) представлены на рисунке 2.7.

Программа в среде PascalABC может манипулировать несколькими изображениями, загружая их сразу или поочередно. При создании нескольких графических объектов в одном окне и импортировании различных изображений можно независимо управлять их свойствами.



Интересные фотoreалистичные модели получаются при совместном использовании растровой и векторной графики.



Пример 4. Составить программу «Секундомер»: на фоне фотографии циферблата секундомера поместить движущиеся стрелки, которые останавливаются при нажатии любой клавиши.

Загрузим изображение циферблата из файла Timer.jpg, поместив описатель в переменную `fon`. Зададим координаты центра вращения стрелок (`x0, y0`) и начальные значения секунд `sec:=0` и минут `min:=0`.

На каждом шагу цикла с постусловием `repeat...until` будем увеличивать значение секунд на 1 до тех пор, пока время не превысит 60 мин, или не будет нажата любая клавиша (`keyPressed`). Значения минут будем вычислять целочисленным делением секунд на 60 (`min:=sec div 60`).

Секундную стрелку будем рисовать линией длиной 120 и толщиной 3 пикселя, а минутную — 100 и 7 пикселей. Радианская мера угла поворота секундной стрелки равна $\text{Pi} * \text{sec} / 30$, а минутной — $\text{Pi} * \text{min} / 30$. Координаты концов стрелок

(x, y) будем вычислять по формулам тригонометрии и округлять до целых. Например, для секундной стрелки используем формулы:

```
x:=x0+Round(120*sin(Pi*sec/30));
y:=y0-Round(120*cos(Pi*sec/30)).
```

Программа может выглядеть так:

```
program Timer;
uses crt, GraphABC;
var fon, x0, y0, x, y, sec, min: integer;
begin
  SetWindowSize(360,480);           {Размеры окна}
  HideCursor;                     {Скрытие курсора}
  x0:=173; y0:=300; sec:=0; min:=0;
  fon:=LoadPicture('timer.jpg');   {Загрузка изображения}
  repeat
    ClearWindow;
    DrawPicture(fon,0,0,360,480);   {Рисование циферблата}
    sec:=sec+1; min:=sec div 60;    {Расчет секунд и минут}
    x:=x0+Round(120*sin(Pi*sec/30)); {Секундная стрелка}
    y:=y0-Round(120*cos(Pi*sec/30)); {Координаты конца}
    SetPenWidth(3); Line(x0,y0,x,y); {Рисование}
    x:=x0+Round(100*sin(Pi*min/30)); {Минутная стрелка}
    y:=y0-Round(100*cos(Pi*min/30)); {Координаты конца}
    SetPenWidth(7); Line(x0,y0,x,y); {Рисование}
    sleep(1000);
    Redraw;
  until (min>=60) or keyPressed;
end.
```

Результат работы программы представлен на рисунке 2.8.



1. Изображения каких форматов можно загружать в среде PascalABC?
2. С помощью какой процедуры выводятся растровые изображения?
3. Как можно имитировать движение изображений в среде PascalABC?
4. Как избежать моргания экрана при имитации движения изображений?

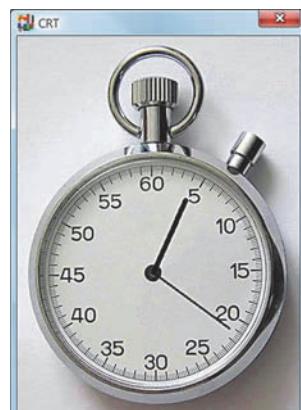
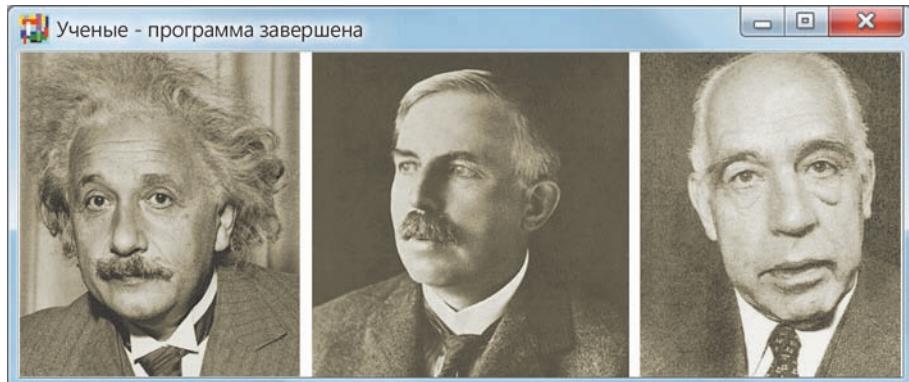


Рис. 2.8

Упражнения

1. Составьте программу, которая загружает из файлов и выводит на экран:
- портреты ученых (например, Эйнштейна, Резерфорда, Бора), писателей (например, Купалы, Коласа, Быкова), художников, композиторов;
 - изображения животных, растений, памятников архитектуры.



2. Составьте программу, которая:

- рисует рекламные плакаты (турфирмы, концерта, спортивного праздника);
- генерирует меню школьной столовой.



3. Составьте программу, которая имитирует падение брошенного с башни яблока.



7.3. Построение графиков и диаграмм

Исследование законов природы и общества немыслимо без построения разнообразных графиков и диаграмм. На уроках математики Вы исследуете функциональные зависимости и строите графики. Замечательными возможностями обработки и представления данных в виде разнообразных графиков и диаграмм обладают электронные таблицы MS Excel, которые Вы изучали в 10-м классе. Познакомимся с построением графиков и диаграмм в среде PascalABC.

Пример 1. Составить программу, которая строит график зависимости $y(x) = 0,4x + 2\sin 3x$ на промежутке $[-15; 15]$.

Определим исходные данные. Прежде всего зададим размеры окна и построим оси координат. Пусть начало отсчета имеет экранные координаты $x_0=320$, $y_0=200$.

Для построения графика необходимо выбрать масштаб. Зададим множитель $k=20$ (единице на графике соответствуют 20 пикселей на экране).

Выполнять вычисления и построения будем в цикле `While`, пока $x <= 15$. Сначала запишем функцию $y := 0.4 * x + 2 * \sin(3 * x)$. Затем перейдем к экранным координатам: $x_1 := \text{trunc}(x_0 + x * k)$; $y_1 := \text{trunc}(y_0 - y * k)$. Дробную часть отбросим, чтобы обеспечить целочисленность. На каждом шагу цикла будем рисовать окружности небольшого радиуса (например, 2) с координатами центра (x_1, y_1) , которые сольются в линию. Для этого приращение аргумента выберем маленьким, например $x := x + 0.02$.

Начальное значение $x = -15$ зададим перед циклом `While`.

Переменные x_0 , y_0 , x_1 , y_1 , k будут иметь тип `integer`, а переменные x , y — тип `real`.

Программа может выглядеть так:

```
program Grafik;
uses GraphABC;
var x0, y0, x1, y1, k: integer; x, y: real;
begin
  SetWindowSize(640,400);
  x0:=320; y0:=200; k:=20;
  line(20,y0, 620,y0);           {Ось абсцисс}
  line(x0,20, x0,380);          {Ось ординат}
  setPenColor(clRed);           {Цвет пера}
  x:=-15;                        {Начальное значение аргумента}
  While x<=15 do
```

```

begin
    y:=0.4*x+2*sin(3*x);      {Функция}
    x1:=trunc(x0+x*k);        {Экранные координаты}
    y1:=trunc(y0-y*k);
    circle(x1,y1,2);
    x:=x+0.02;
end;
end.

```

Результат работы программы показан на рисунке 2.9.

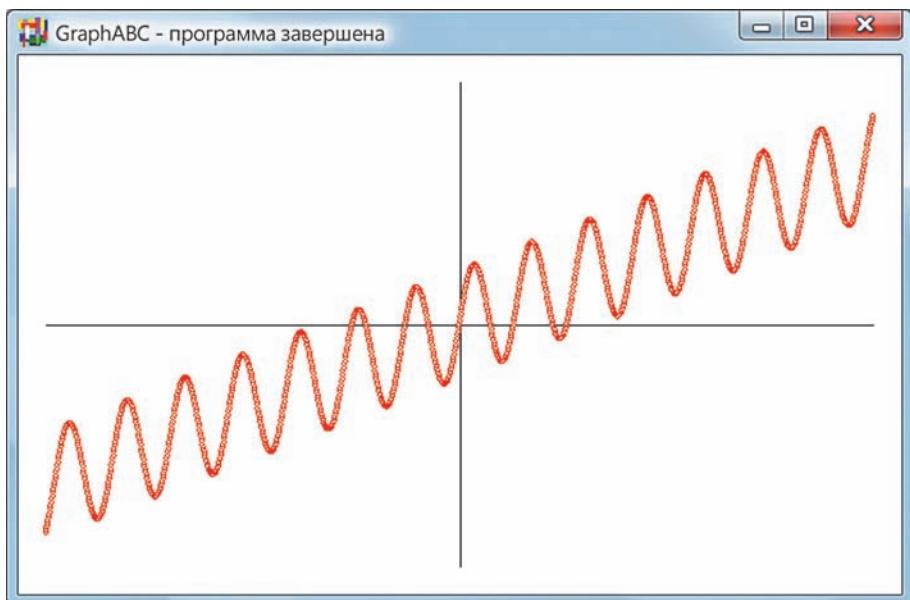


Рис. 2.9



При необходимости на осях координат несложно нарисовать деления. Например, с помощью цикла `for` с параметром `i`, который изменяется от 0 до 30, на оси абсцисс можно отметить 31 деление через 20 пикселей. На оси ординат с помощью цикла `for` с параметром `i`, который изменяется от 0 до 20, можно отметить 21 деление.

Фрагмент программы может выглядеть так:

```

for i:=0 to 30 do line(20+k*i,195, 20+k*i,205);
                    {Деления по оси абсцисс}
for i:=0 to 20 do line(315,20+k*i,325,20+k*i);
                    {Деления по оси ординат}

```



Эффективным средством наглядного представления данных являются диаграммы. Они облегчают сравнение данных, помогают выявлению закономерностей их изменения. Вместо кропотливого сравнения чисел достаточно бросить взгляд на диаграмму, чтобы увидеть, снижается или растет температура, урожайность, цена или объем продаж.

В 10-м классе Вы познакомились с построением диаграмм в электронных таблицах. Рассмотрим теперь пример программирования круговых диаграмм.

Круговые диаграммы применяют в случаях, когда необходимо показать долю каждой составляющей в общем объеме данных, например структуру посевов, распределение оценок по классам и предметам, цену каждого товара в общей стоимости, состав сплава или смеси и т. п.

Напомним, что данные, которые предполагается использовать многократно, удобно хранить в массивах.



Пример 2. В массивах хранятся данные о площадях посевов зерновых культур. Составить программу, которая вычисляет, какой процент в общей площади посевов зерновых составляет площадь посевов каждой культуры, выводит данные и строит круговую диаграмму.

Данные о площадях посевов зерновых культур будем хранить в массиве целочисленных констант A, а их названия — в массиве строковых констант B:

```
Const A: array[1..5] of integer = (47, 63, 22, 27, 12);
    B: array[1..5] of string = ('Рожь ', 'Пшеница',
    'Овес ', 'Ячмень', 'Гречиха');
```

Круговая диаграмма будет состоять из 5 секторов. Градусная мера каждого сектора будет определяться значением соответствующего элемента массива.

Прежде всего в цикле `for` вычислим суммарную площадь всех посевов S (ей соответствует угол 360°). Площади i -й культуры будет соответствовать угол $A[i]*360/S$.

Для построения каждого сектора нужно знать значения двух углов: начального и u_0 и конечного u .

Первый сектор будем строить от начального угла $u_0 := 0$. В цикле `for` будем вычислять значение конечного угла $u := u_0 + round(A[i]*360/S)$. Следующий сектор будем строить уже от этого угла, поэтому его начальный угол будет равен конечному углу предыдущего сектора $u_0 := u$.

Цвет сектора будем задавать случайным образом.

Переменные u , u_0 , S , i , а также элементы массива A будут иметь тип `integer`, элементы массива B — тип `string`.

Программа может выглядеть так:

```

program Diagram2;
uses crt, GraphAbc;
const A: array[1..5] of integer = (47, 63, 22, 27, 12);
      B: array[1..5] of string = ('Рожь ','Пшеница', 'Овес ',
        'Ячмень', 'Гречиха');
var u, u0, S, i: integer;
begin
  SetWindowSize(400,400);  S:=0;  u0:=0;
  for i:=1 to 5 do S:=S+A[i];    {Вычисление суммарной
                                    площади посевов}
  for i:=1 to 5 do
    begin
      Writeln(B[i],' ',A[i],' га ',round(100*A[i]/S), '%');
      SetBrushColor(rgb(random(255),
        random(255),random(255)));
      u:=u0+round(A[i]*360/S);    {Вычисление конечного
                                    угла сектора}
      Pie(220,240,140,u0,u);    {Рисование сектора}
      u0:=u;                      {Новое значение началь-
                                    ного угла следующего сек-
                                    тора}
    end;
end.
```

Результат работы программы приведен на рисунке 2.10.

В некоторых задачах может требоваться определенный цвет каждого сектора. Его удобно определять в массиве целочисленных констант, например так:

```

col: array[1..5] of integer =
  =(clYellow,clLime,clOlive,
    clBrown,ClRed);
```

и задавать в цикле так:

```
SetBrushColor(col[i]);
```

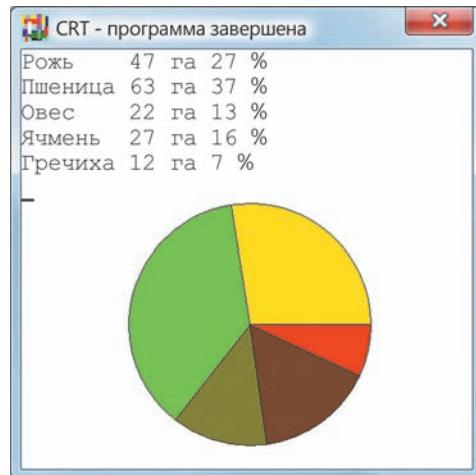


Рис. 2.10



В случаях, когда необходимо указывать не только численное значение, но и направление исследуемых величин, используются **лепестковые диаграммы**. Такие диаграммы Вы использовали в географии при построении розы ветров.



Пример 3. Составить программу, которая строит розу ветров (рис. 2.11). Данные метеорологических наблюдений за месяц хранятся в массиве.

Данные наблюдений о количестве дней с преобладающими ветрами определенного направления будем хранить в массиве констант N типа `integer`.

Диаграмму будем строить на фоне изображения контурной карты Беларуси. Загрузим рисунок из файла `karta.gif`, поместив его описание в переменную `p`. Линии на карте соответствуют восьми основным направлениям ветров (`B`, `СВ`, `С`, `СЗ`, `З`, `ЮЗ`, `Ю`, `ЮВ`), а радиусы окружностей — количеству дней с преобладанием этих ветров.

Пусть центр диаграммы имеет координаты x_0, y_0 . С помощью цикла `for` с параметром `k`, который изменяется от 1 до 8 в соответствии с количеством направлений ветра, будем вычислять координаты точек, соответствующих количеству дней с ветрами каждого направления: $x := x_0 + \text{trunc}(20 * N[k] * \cos(u))$; $y := y_0 - \text{trunc}(20 * N[k] * \sin(u))$; где направление ветра определяется углом `u` (отсчитывается от восточного). Соединим эти точки линиями: `lineTo(x, y)`.

Программа может выглядеть так:

```
program roza_vetrov;
uses crt, GraphABC;
const N: array[1..8] of integer = (2, 1, 4, 7, 4, 2, 4, 3);
var x0, y0, x, y, k, p: integer; u: real;
begin
  SetWindowSize(420,420); HideCursor;
  p:=LoadPicture('karta.gif');
  DrawPicture(p,10,10); { Вывод изображения карты }
  SetPenColor(clRed); SetPenWidth(5);
```

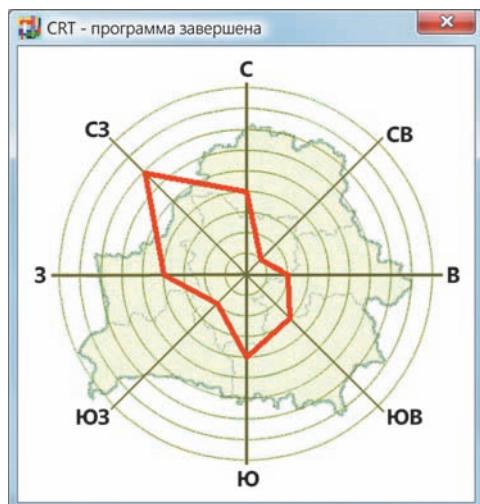


Рис. 2.11

```

x0:=210; y0:=210;           {Начало системы координат}
u:=0;  moveTo(x0+20*N[1],y0);
for k:=1 to 8 do begin
  x:=x0+trunc(20*N[k]*cos(u));
  y:=y0-trunc(20*N[k]*sin(u));
  lineTo(x,y); u:=u+pi/4;
end;
lineTo(x0+20*N[1],y0);
end.

```

Результат работы программы приведен на рисунке 2.11.

Упражнения

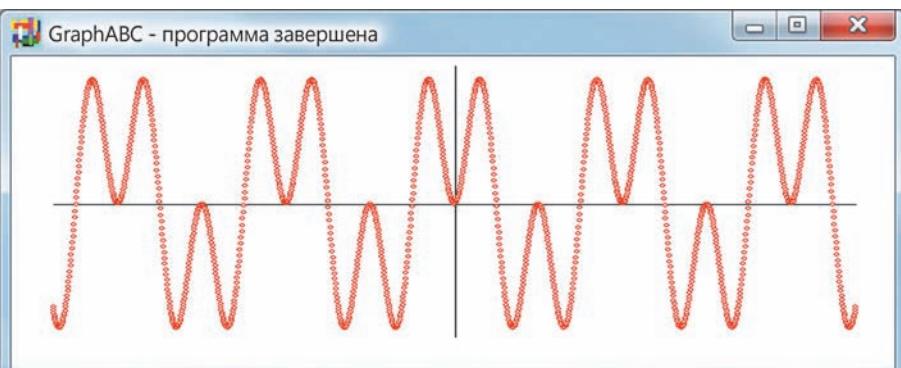
1. Составьте программу, которая строит график зависимости:

- $y(x) = 0,5x \cdot \cos 2x$ на промежутке $[-12; 12]$;
- $y(x) = 8 \sin x \cdot \sin 2x$ на промежутке $[-15; 15]$.

a



б





2. Составьте программу, которая строит круговую диаграмму, иллюстрирующую:

а) распределение результатов тестирования по трем предметам (данные вводятся с клавиатуры);

б) химический состав земной коры (данные вводятся с клавиатуры: кислород — 47,2 %, кремний — 27,6 %, алюминий — 8,3 %, железо — 5,1 %, кальций — 3,6 %, другие элементы — 8,2 %).

§ 8. Выполнение практических заданий по темам учебных предметов

8.1. Астрономия

Пример 1. Составить программу, демонстрирующую фазы Луны (изображения Луны загружаются из файлов).

Демонстрация фаз Луны заключается в смене на экране 28 изображений Луны, соответствующих каждому дню лунного месяца. Эти изображения будут поочередно загружаться из файлов и демонстрироваться с задержкой в 500 мс.

Используем цикл `for` с параметром `i`, который изменяется от 1 до 28 в соответствии с номером дня. Имя файла `imf` будет иметь тип `string`. Сформируем его путем слияния слова `'luna'`, преобразованного в строку номера дня `IntToStr(i)`, и расширения имени файла с изображением в формате `.gif` (перед «`.gif`» не забудем поставить точку): `imf:='luna'+IntToStr(i)+'.gif'`.

Программа может выглядеть так:

```
program Luna;
uses GraphABC;
var p, i: integer; imf: string;
begin
  SetWindowSize(300,300);           {Размеры окна}
  for i:=1 to 28 do
    begin
      imf:='luna'+IntToStr(i)+'.gif'; {Формирование имени файла}
      p:=LoadPicture(imf);          {Загрузка изображения из файла}
      DrawPicture(p,0,0);           {Вывод изображения}
      sleep(500);
    end;
end.
```



Рис. 2.12

Протестируем программу. На рисунке 2.12 показаны кадры 4, 8, 14, 20 и 24 (фазы Луны в соответствующие дни лунного месяца).

При моделировании и демонстрации различных процессов и явлений их параметры могут задаваться путем ввода с клавиатуры.



Пример 2. Составить программу, которая демонстрирует увеличение изображения Луны в телескопе. Коэффициент увеличения вводится с клавиатуры.

Загрузим изображения звездного неба и Луны из файлов `sky.gif` и `luna2.gif`, поместив их описатели в переменные `nebo` и `luna`. Установим прозрачность фона для изображения Луны: `SetPictureTransparent(luna, true)`.

Зададим начальные координаты (`x:=200`; `y:=200`), ширину и высоту изображения (`w:=100`; `h:=100`).

Коэффициент увеличения будем вводить с клавиатуры и помещать в переменную k . Переменные $nebo$, $luna$, x , y , w , h будут иметь тип `integer`, переменная k — тип `real`.

Процедуры рисования и стирания будем повторять в цикле с предусловием `While`. Пусть на каждом шагу цикла ширина и высота изображения увеличиваются на 10 пикселей, пока не будет достигнуто требуемое увеличение k , т. е. пока ширина изображения не станет равной $k * 100$. Чтобы положение центра изображения Луны при этом оставалось неизменным, координаты левого верхнего угла изображения на каждом шагу будем уменьшать на 5.

Программа может выглядеть так:

```
program Teleskop;
uses crt, GraphABC;
var nebo, luna, x, y, w, h: integer; k: real;
begin
  SetWindowSize(500,500);
  nebo:=LoadPicture('sky.gif'); DrawPicture(nebo,0,20,500,500);
```

```

luna:=LoadPicture('luna2.gif');
SetPictureTransparent(luna,true);
x:=200; y:=200;                                {Начальное положение}
w:=100; h:=100;                                 {Размеры изображения}
DrawPicture(luna,x,y,w,h);
write('Введите увеличение (<=5). k = ');
read(k);                                         {Ввод увеличения}
while w<k*100 do
begin
  w:=w+10; h:=h+10; x:=x-5; y:=y-5;
  DrawPicture(luna,x,y,w,h);
  sleep(20);                                     {Задержка на 20 мс}
  Redraw;                                         {Перерисовка}
end;
end.

```

Изображения Луны на первом и последнем шагу демонстрации при $k = 4$ представлены на рисунке 2.13.



Рис. 2.13



Пример 3. Создать программу, демонстрирующую движение Луны вокруг Земли по круговой орбите.

Параметры орбиты будем задавать ее радиусом R и координатами (x, y) левого верхнего угла прямоугольника с изображением планеты.

Координаты движущейся по окружности точки относительно центра вращения (x_0, y_0) будем вычислять по формулам тригонометрии:

$x := x_0 + R * \cos(\pi * u / 180)$ и $y := y_0 - R * \sin(\pi * u / 180)$, где u — угол в градусах, и округлять до целых чисел.

Для одного периода обращения процедуры рисования и стирания будем повторять 360 раз с шагом 1 градус в цикле For.

Все переменные будут иметь тип integer.

Программа может выглядеть так:

```
program planety;
uses GraphABC;
var nebo, zem, luna, x0, y0, R, x, y, u: integer;
begin
  SetWindowSize(600,400);
  x0:=300; y0:=200; {Координаты центра}
  R:=160; {Радиус орбиты}
  nebo:=LoadPicture('sky.gif'); {Загрузка изображений}
  zem:=LoadPicture('earth.gif'); {из файлов}
  luna:=LoadPicture('luna3.gif');
  for u:=1 to 360 do
    begin
      DrawPicture(nebo,0,0,600,400);
      DrawPicture(zem,200,120,200,200);
      x:=round(x0+R*cos(pi*u/180)); {Координаты Луны}
      y:=round(y0-R*sin(pi*u/180));
      DrawPicture(luna,x,y,50,50); {Рисование Луны}
      sleep(10); {Задержка на 10 мс}
      Redraw; {Перерисовка}
    end;
end.
```

Фрагмент демонстрации движения Луны вокруг Земли показан на рисунке 2.14.

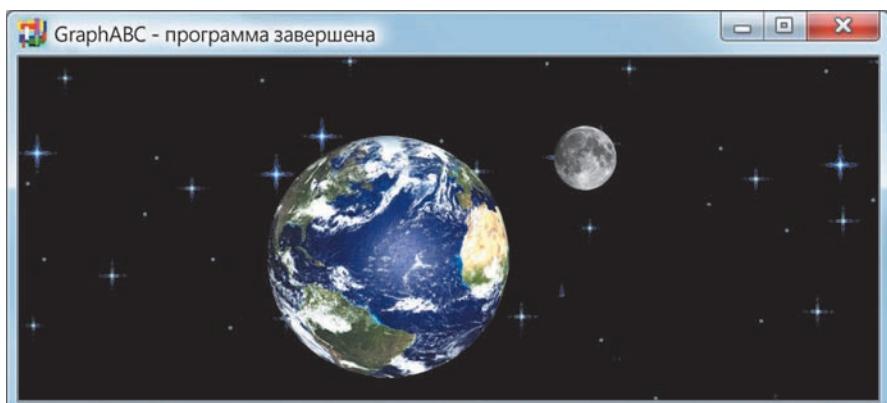
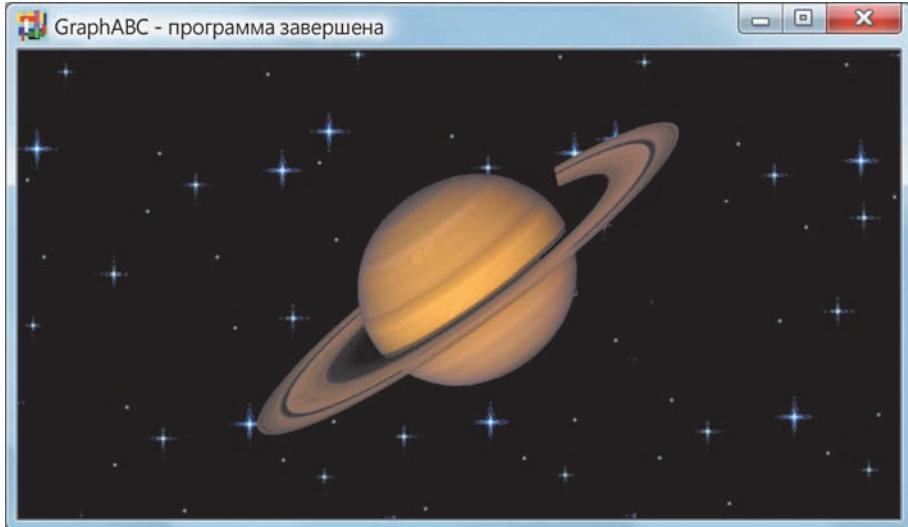


Рис. 2.14

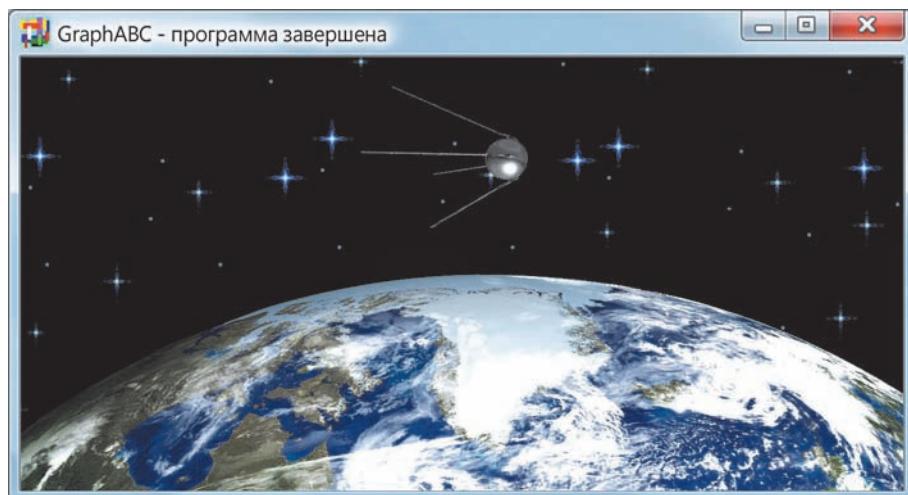
Упражнения

1. Составьте программу, которая загружает из файлов и выводит на экран изображение одной из планет Солнечной системы на фоне звездного неба.



2. Составьте программу, которая демонстрирует:

- а) прямолинейное движение спутника на фоне звездного неба и Земли;
- б) движение планет вокруг Солнца по круговым орбитам.



8.2. География

Пример 1. Составить программу, обозначающую на карте Беларуси города кругами, площадь которых пропорциональна количеству жителей (Минск — 1700 тыс. чел., Гомель — 500 тыс. чел., Могилев — 330 тыс. чел., Витебск — 340 тыс. чел., Гродно — 320 тыс. чел., Брест — 300 тыс. чел.).

Загрузим изображение контурной карты Беларуси из файла belarus.gif, поместив его описатель в переменную `karta`.

В простейшем случае будем последовательно присваивать переменной `A` значения количества жителей городов и записывать процедуры рисования кругов соответствующего радиуса R . Радиус будем вычислять из условия $A = k\pi R^2$, где коэффициент пропорциональности k подбирается для получения наглядности. Примем $k = 1$ и округлим значение радиуса $R := \text{round}(\sqrt{A/\pi})$. Координаты центров кругов соответствуют координатам городов на карте и подбираются опытным путем.

Программа может выглядеть так:

```
program Geograf1;
uses GraphABC;
var karta, R, A: integer;
begin
  SetWindowSize(520,480);
  karta:=LoadPicture('belarus.gif');
  DrawPicture(karta,10,10);
  setBrushColor(clRed);
  A:=1700; R:=round(sqrt(A/pi)); circle(250,225,R); {Минск}
  A:=500; R:=round(sqrt(A/pi)); circle(405,345,R); {Гомель}
  A:=330; R:=round(sqrt(A/pi)); circle(375,215,R); {Могилев}
  A:=340; R:=round(sqrt(A/pi)); circle(360,105,R); {Витебск}
  A:=320; R:=round(sqrt(A/pi)); circle( 55,235,R); {Гродно}
  A:=300; R:=round(sqrt(A/pi)); circle( 40,385,R); {Брест}
end.
```

Результат работы программы представлен на рисунке 2.15.



Пример 2. Составить программу, которая выводит физическую карту Беларуси, данные о длине рек и строит линейчатую диаграмму.

Данные о длине рек на территории Беларуси будем хранить в массиве констант `A` типа `integer`, а их названия — в массиве `B` типа `string`.

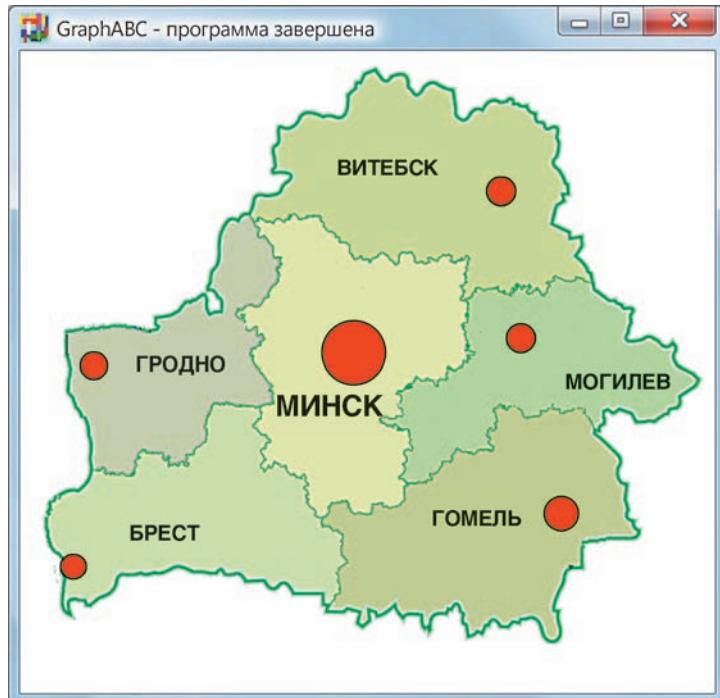


Рис. 2.15

Изображение физической карты Беларуси загрузим из файла `belfiz.gif`, поместив его описатель в переменную `karta`.

Вывод данных и линий диаграммы выполним с помощью цикла `For`. Для линии пропорциональна длине реки `A[i]`. Коэффициент пропорциональности примем равным 0,5, абсциссу левого конца — 160. По вертикали линии будем располагать через 20 пикселей. Их ординаты выразим через параметр цикла `i`.

Программа может выглядеть так:

```
program Geograf2;
uses crt, GraphABC;
const A: array[1..5] of integer = (690, 613, 495, 493, 459);
      B: array[1..5] of string = ('Днепр ', 'Березина', 'Припять ',
      'Сож ', 'Неман ');
var karta, i: integer;
begin
  SetWindowSize(600,520);
  karta:=LoadPicture('belfiz.gif');
  DrawPicture(karta,30,120); {Карта}
```

```

SetPenColor(clBlue);           {Цвет пера}
SetPenWidth(5);               {Толщина пера}
Writeln('Длина реки на территории Беларуси, тыс. км');
For i:=1 to 5 do begin
  Writeln(B[i],', ',A[i]);      {Вывод данных}
  Line(160, 10+20*i, 160+round(0.5*A[i]), 10+20*i);
                                {Рисование линий}
end;
end.

```

Результат работы программы представлен на рисунке 2.16.

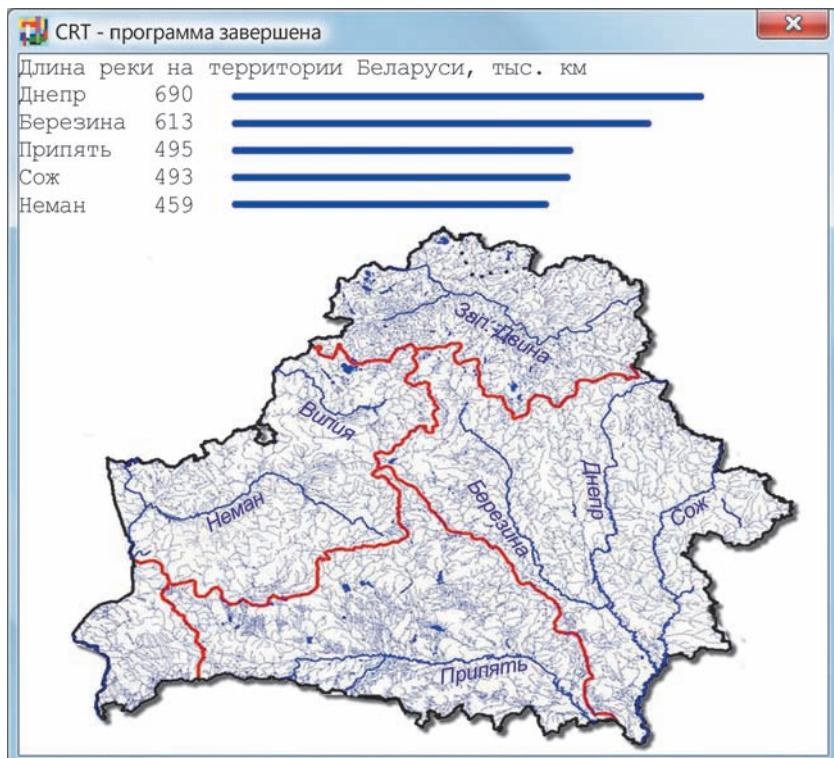


Рис. 2.16



Хранящиеся в массивах данные можно подвергать дальнейшей обработке. Например, по количеству жителей и площади территории страны (области, района) можно вычислить среднюю плотность населения.



Пример 3. Составить программу, которая вычисляет и показывает на карте среднюю плотность населения областей Беларуси.

Необходимые данные будем хранить в массивах констант: названия областей — в массиве $A[i]$, численность населения — в массиве $N[i]$, площадь — в массиве $S[i]$, координаты символов — в массивах $x[i]$ и $y[i]$.

Загрузим изображение карты и символ человечка из файлов *belarus.gif* и *man.gif*, поместив их описатели в переменные *karta* и *ris*. Установим прозрачность фона для изображения человечка.

Вычисления плотности населения и вывод данных, масштабирование и размещение символов на карте будем выполнять с помощью цикла *For*. Примем ширину символа равной k , а высоту — $2k$.

Программа может выглядеть так:

```
program Geograf3;
uses crt, GraphABC;
const A: array[1..6] of string = ('Брестская', 'Витебская',
    'Гомельская', 'Гродненская', 'Минская', 'Могилевская');
    N: array[1..6] of integer = (1440, 1300, 1500, 1100,
    1500, 1200);
    S: array[1..6] of real = (32.8, 40.1, 40.4, 25.1,
    40.8, 29.1);
    x: array[1..6] of integer = (140, 315, 365, 65, 200,
    380);
    y: array[1..6] of integer = (320, 80, 290, 230, 180,
    175);
var karta, ris, k, i: integer; p: real;
begin
    setWindowSize(520,480);
    karta:=LoadPicture('belarus.gif');
    DrawPicture(karta,10,30); {Карта}
    ris:=LoadPicture('man.gif');
    SetPictureTransparent(ris,true);
    Writeln('Плотность населения, тыс. чел. на кв. км');
    for i:=1 to 6 do begin
        p:=N[i]/S[i]; k:=round(p); {Расчет плотности и
        масштаба}
        Writeln(A[i]:11, p:5:1); {Вывод}
        DrawPicture(ris,x[i],y[i],k,2*k); {Вывод изображений}
    end;
end.
```



Рис. 2.17

Результат работы программы представлен на рисунке 2.17.

Упражнение

Составьте программу, которая:

- а) выводит карту Беларуси и обозначает национальные парки квадратами, размеры которых пропорциональны площади парка: Беловежская пуща — 87,5 тыс. га, Браславские озера — 71,5 тыс. га, Припятский — 82,4 тыс. га, Нарочанский — 94 тыс. га;

- б) выводит карту Беларуси, строит таблицу с данными о площади и глубине озер, столбчатую или линейчатую диаграмму (по указанию учителя);
 в) выводит карту Европы и помечает страны (столицы государств) символами, размеры которых пропорциональны численности населения.

8.3. Биология и экология

Пример 1. Составить программу, которая вычисляет, через сколько дней уровень радиации в зоне заражения упадет до безопасного значения в 10 единиц, если каждый день он уменьшается на 5 % по отношению к предыдущему.

му дню. Начальное значение — от 100 до 10 000 единиц — вводится с клавиатуры.

Начальное значение уровня радиации будем вводить в переменную `ur`, а процент ежедневного уменьшения уровня, безопасное значение и количество дней будем хранить соответственно в переменных `p`, `ur0` и `t`. Переменные `ur`, `ur0` будут иметь тип `real`, а процент `p` и количество дней `t` — тип `integer`.

Каждый день уровень радиации уменьшается на $ur * p / 100$ по отношению к предыдущему дню, т. е. определяется выражением `ur:=ur-ur*p/100`.

Вычисления будем повторять в цикле `while`, пока выполняется условие `ur>=ur0`, т. е. пока не будет достигнут безопасный уровень радиации.

Программа может выглядеть так:

```
program Ecolog;
var ur0, ur: real; p, t: integer;
begin
  write('Начальный уровень ');
  readln(ur);
  p:=5; ur0:=10; t:=0; {Начальные значения}
  while ur>=ur0 do
    begin
      ur:=ur-ur*p/100; t:=t+1;
    end;
  Write('Через ', t, ' дней.')
end.
```

Результат работы программы может выглядеть так:

Начальный уровень 3000
Через 112 дней.



Пример 2. Составить программу, которая имитирует наблюдение клетки под микроскопом. Для увеличения используются клавиши управления курсором.

Загрузим изображение клетки из файла `kletka.jpg`, поместив его описатель в переменную `img`. Зададим начальные координаты `x,y` левого верхнего угла изображения, его ширину `w` и высоту `h`.

Процедуры рисования будем повторять в цикле с постусловием `Repeat`. С помощью функции `ReadKey` символ нажатой клавиши будет считываться и помещаться в переменную `c` типа `char`. При нажатии клавиши управления курсором «стрелка вверх» (`c=#72`) ширина и высота изображения увеличатся на $2*d=10$ пикселей (`d=5`), а при нажатии клавиши «стрелка вниз»

($c=\#80$) — уменьшается ($d=-5$). Чтобы положение центра изображения при этом оставалось неизменным, будем изменять соответствующим образом координаты его левого верхнего угла.

При нажатии клавиши Esc ($c=\#27$) работа программы завершится.

Переменные `img`, `x`, `y`, `w`, `h`, `d` будут иметь тип `integer`, переменная `c` — тип `char`.

Программа может выглядеть так:

```
program Microscop;
uses crt, GraphABC;
var img, x, y, w, h, d: integer; c: char;
begin
  SetWindowSize(400,260);
  img:=LoadPicture('kletka.jpg');
  x:=0; y:=0; {Начальные координаты}
  w:=400; h:=260; {Размеры изображения}
  repeat
    ClearWindow(clSkyBlue);
    DrawPicture(img,x,y,w,h); {Вывод изображения}
    d:=0; c:=ReadKey;
    if c=#72 then d:=5; {Стрелка вверх — увеличение}
    if c=#80 then d:=-5; {Стрелка вниз — уменьшение}
    w:=w+2*d; h:=h+2*d; x:=x-d; y:=y-d;
  until c=#27;
end.
```

Результаты работы программы представлены на рисунке 2.18.

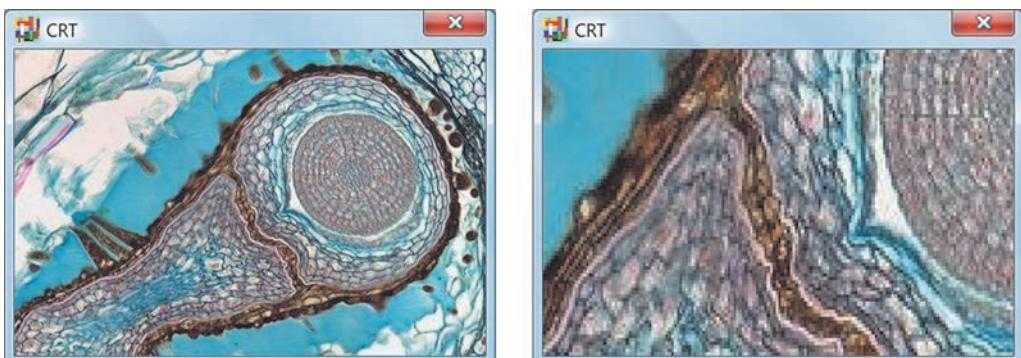


Рис. 2.18



Пример 3. Первоначально участок леса содержит 80 тыс. куб. м древесины. Ежегодный естественный прирост составляет $P\%$, а вырубка — D тыс. куб. м. Составить программу, которая вычисляет ежегодные объемы древесины для T лет и результаты выводят в таблицу (рис. 2.19). Значения P (от 2 до 8 %), D (от 3 до 15 тыс. куб. м) и T (от 2 до 20 лет) вводятся с клавиатуры.

Значения объема древесины будем хранить в переменной V , а прирост, объем вырубки и количество лет будем вводить с клавиатуры в переменные P , D и T . Переменные P , D , V будут иметь тип `real`, а количество лет T и счетчик цикла i — тип `integer`.

За счет естественного прироста объем древесины каждый год увеличивается на $V*P/100$ по сравнению с предыдущим, а вследствие вырубки уменьшается на D , т. е. определяется выражением

$$V := V * (1 + P / 100) - D; .$$

Вычисления и вывод результатов в строки таблицы будем повторять в цикле `while` до тех пор, пока не будет достигнуто заданное количество лет ($i \leq T$) и лес не будет вырублен ($V >= 0$).

Программа может выглядеть так:

```
program Les;
uses crt, GraphABC;
var P, D, V: real; T, i: integer;
begin
  writeln('Прирост (от 2 до 8 %) ');
  readln(P); {Ввод данных}
  writeln('Вырубка (от 5 до 15 тыс. куб. м) ');
  readln(D);
  writeln('Время (от 2 до 20 лет) ');
  readln(T);
  v:=80; i:=0; {Начальные значения}
  writeln('-----');
  writeln(' Год Объем, тыс. куб. м'); {Заголовок таблицы}
```

CRT - программа завершена	
Прирост (от 2 до 8 %)	3
Вырубка (от 5 до 15 тыс. куб. м)	11
Время (от 2 до 20 лет)	15
<hr/>	
Год	Объем, тыс. куб. м
0	80.0
1	71.4
2	62.5
3	53.4
4	44.0
5	34.3
6	24.4
7	14.1
8	3.5

Рис. 2.19

```

while (i<=T) and (V>=0) do
begin
  writeln(i:3, V:12:1);           {Вывод результатов}
  V:=V*(1+P/100)-D; i:=i+1;
end;
end.

```

Результат работы программы представлен на рисунке 2.19.

Заметим, что повторение цикла прекращено раньше введенного времени (15 лет), поскольку уже к 9-му году лес будет вырублен.

Упражнение

Составьте программу, которая вычисляет:

а) через сколько дней уровень загрязнения воды отходами уменьшится в N раз, если каждый день он уменьшается на $P\%$ (значения N и P вводятся с клавиатуры);

б) через сколько дней количество атомов радиоактивного изотопа Иод-131 уменьшится в 100 раз, если период полураспада (время уменьшения количества атомов вдвое) составляет 8,14 дня;

в) через сколько дней выздоровеет больной, т. е. через сколько дней концентрация болезнетворных бактерий в крови уменьшится с 50 до 12 единиц. В результате применения лекарства концентрация бактерий ежедневно уменьшается на 20 % по сравнению с предыдущим днем.

8.4. Физика

Построение изображения в линзе. На уроках физики изображения в линзах строятся с помощью линейки и карандаша. При этом для каждого нового положения предмета приходится повторять все вычисления и построения. С помощью компьютерной программы можно легко строить изображения, изменяя расстояние d от линзы до предмета, а также его высоту h (рис. 2.20).

Пример 1. Составить программу, демонстрирующую построение изображения в собирающей линзе.

Загрузим из файла Linza.jpg изображение собирающей линзы на фоне координатной сетки $p:=LoadPicture('Linza.jpg')$. С помощью процедуры $DrawPicture(p, 0, 0)$ выведем это изображение в позицию с координатами левого верхнего угла сетки $(0, 0)$.

Размеры изображения будут соответствовать размерам графического окна 800×480 , одно деление сетки будет равно 40 пикселям. Пусть центр линзы O

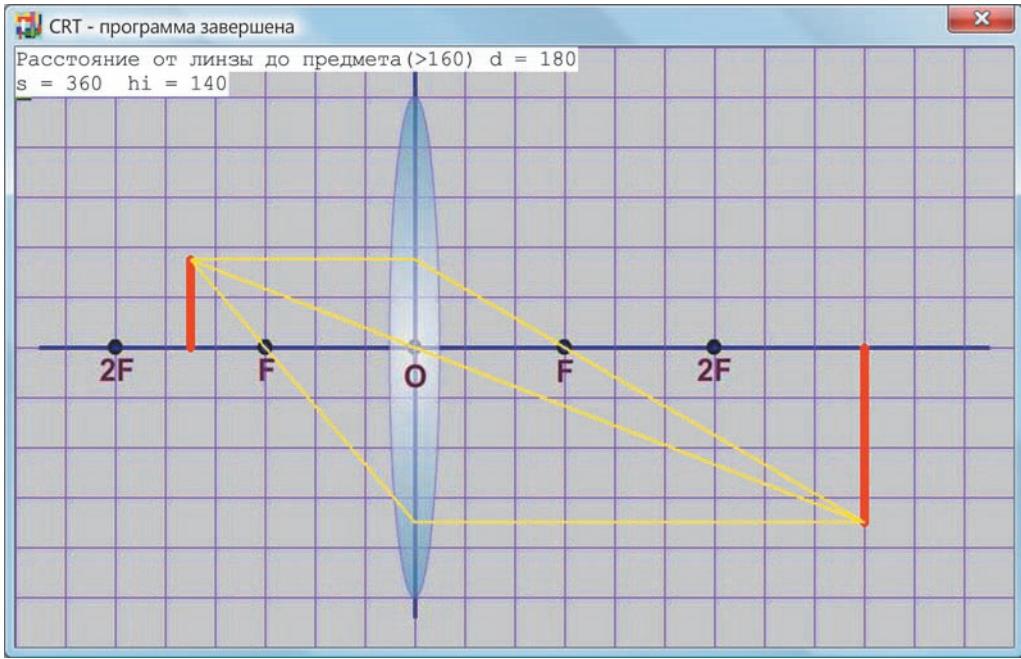


Рис. 2.20

имеет координаты $x0=320$, $y0=240$, фокусное расстояние равно $F = 120$. На оптической оси линзы отмечены фокусы и двойные фокусы.

Будем помещать предмет высотой h (например, отрезок красного цвета толщиной 7 и высотой 70 пикселей) на расстоянии d от линзы: `line(x0-d,y0, x0-d, y0-h)`. Высоту предмета будем изменять в программе, а расстояние d будем вводить с клавиатуры. Границы изменения d и h подберем так, чтобы изображение не выходило за пределы экрана, например для $h = 70$ расстояние $d > 160$.

На основании формулы тонкой линзы вычислим расстояние от линзы до изображения $s := F * d / (d - F)$ и высоту изображения $hi := h * s / d$. Округлим полученные значения до целых и выведем на экран.

Построим изображение — отрезок `line(x0+s,y0, x0+s,y0+hi)`.

Наконец проведем вспомогательные линии, которые проходят через фокусы или оптический центр линзы и иллюстрируют построение изображения.

Программа может выглядеть так:

```
program Linza;
uses crt, GraphABC;
var p, x0, y0, F, h, d, s, hi: integer;
```

```

begin
    SetWindowSize(800,480);
    p:=LoadPicture('linza.jpg'); DrawPicture(p,0,0); {Линза}
    x0:=320; y0:=240; F:=120;
    h:=70; {Высота предмета}
    write('Расстояние от линзы до предмета (>160) d = ');
    readln(d);
    SetPenColor(clRed); SetPenWidth(7);
    line(x0-d,y0, x0-d,y0-h); {Рисование предмета}
    s:=round(F*d/(d-F)); {Расчеты и округление}
    hi:=round(h*s/d);
    writeln('s = ',s, ' hi = ',hi); {Вывод значений s и hi}
    line(x0+s,y0, x0+s,y0+hi); {Рисование изображения}
    SetPenColor(clYellow); SetPenWidth(2);
    line(x0-d,y0-h, x0+s,y0+hi); {Рисование хода лучей}
    line(x0-d,y0-h, x0,y0-h); line(x0,y0-h, x0+s,y0+hi);
    line(x0-d,y0-h, x0,y0+hi); line(x0,y0+hi, x0+s,y0+hi);
end.

```

Результат работы программы приведен на рисунке 2.20.



Наиболее интересный и удобный для пользователя вариант программы можно получить, если заменить ввод параметров предмета с клавиатуры на их изменение клавишами управления курсором.

С этой целью процедуры рисования и расчеты поместим в цикл с постусловием `Repeat`. С помощью функции `ReadKey` будем считывать символ нажатой клавиши и помещать в переменную `s` типа `char`. При нажатии клавиши «стрелка вверх» (`s=#72`) высота изображения увеличится, а при нажатии клавиши «стрелка вниз» (`s=#80`) — уменьшится. Нажатие клавиши «стрелка влево» (`s=#75`) приведет к удалению предмета от линзы, нажатие клавиши «стрелка вправо» (`s=#77`) — к его приближению. При нажатии клавиши `Esc` (`s=#27`) работа программы завершится.

Модифицированный фрагмент программы может выглядеть так:

```

h:=70; d:=200; {Параметры предмета}
Repeat
    DrawPicture(p,0,0); {Вывод изображения линзы}
    SetPenColor(clRed); SetPenWidth(7);

```

```

... {Расчеты, рисование предмета, изображения и хода лучей}
c:=ReadKey;
if c=#77 then d:=d+5; {Приближение предмета}
if c=#75 then d:=d-5; {Отдаление предмета}
if c=#72 then h:=h+2; {Увеличение высоты предмета}
if c=#80 then h:=h-2; {Уменьшение высоты предмета}
Until c=#27;

```



Вместо вывода значений s , hi с помощью процедуры `writeln` удобнее отображать параметры предмета (d, h) и изображения (s, hi) в заголовке окна. Для этого преобразуем их к строчному типу и с помощью операций слияния поместим в переменную `str` типа `string`:

```

str:='d='+IntToStr(d)+'h='+IntToStr(h)+'s='
+IntToStr(s)+'hi='+IntToStr(hi);
SetWindowCaption(str);

```



Построение интерференционной картины. В курсе физики Вы познакомились с явлением интерференции. Напомним, что **интерференцией** называют явление сложения когерентных волн, приводящее к образованию в пространстве устойчивой картины чередующихся максимумов и минимумов.

Зависимость смещения s колеблющейся точки, находящейся на расстоянии r от источника колебаний, от времени t описывается уравнением волны:

$$s(t,r) = A \sin(\omega t - kr), \text{ где } k = 2\pi/\lambda, \lambda \text{ — длина волны.}$$



Пример 2. Создать программу, демонстрирующую явление интерференции.

При построении любой модели приходится принимать некоторые допущения и упрощения. Так, в нашей модели используются точечные источники волн. Амплитуда колебаний в пределах области наблюдения считается неизменной.

Для простоты будем рассматривать источники колебаний одинаковой амплитуды и частоты с нулевой начальной фазой в момент времени $t = 0$, которые находятся в точках с координатами (x_1, y_1) и (x_2, y_2) .

Запишем уравнения волн в точке, которая отстоит от источников колебаний на расстояния r_1 и r_2 :

$$s_1(r_1) = A \sin(-2\pi r_1/\lambda) \quad \text{и} \quad s_2(r_2) = A \sin(-2\pi r_2/\lambda).$$

Осталось связать амплитуду результирующего колебания с яркостью пикселей на экране монитора. Можно использовать серый цвет (яркость изменяется от черного до белого) или цвет одной из трех составляющих в цветовой модели RGB. Если минимальному значению смещения поставить в соответствие черный цвет (уровень = 0), максимальному — белый (уровень = 255), среднему — серый (уровень ≈ 120), то промежуточные значения будут отображаться разными оттенками серого. Мы используем красный цвет, приняв значение 120 за его среднюю яркость.

Отметим, что на быстроту прорисовки существенно влияют размер графического окна и шаг изменения координат. Зададим размеры окна 260×300 пикселей (300 строк по 260 пикселям). Чтобы произвести расчеты и рисование всех пикселей, используем вложенный цикл. Во внутреннем цикле с параметром x , значения которого изменяются от 0 до 260, будут выполняться действия для каждой точки строки. Во внешнем цикле с параметром y , значения которого изменяются от 0 до 300, будут выполняться действия для всех строк. Блок-схема этого алгоритма приведена на рисунке 2.21.

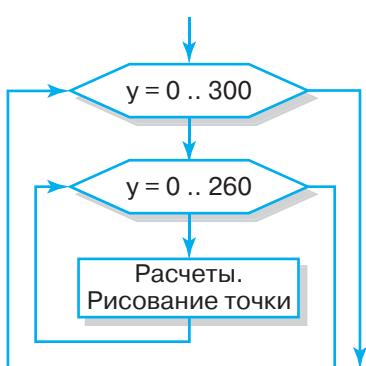


Рис. 2.21

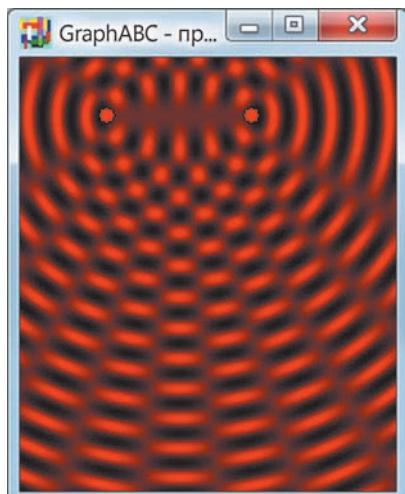


Рис. 2.22

Программа может выглядеть так:

```

program interfer;
uses graphABC;
var x, y, x1, y1, x2, y2, s, A, L: integer; r1, r2: real;
begin
  SetWindowSize(260,300);
  SetBrushColor(clRed); {Красный цвет}
  x1:=60; y1:=40; x2:=160; y2:=40; {Координаты источников}

```

```

A:=60; L:=20;                                {Амплитуда и длина волны}
for y:=0 to 300 do
  for x:=0 to 260 do begin
    {Вычисление расстояний до источников}
    r1:=sqrt((x-x1)*(x-x1)+(y-y1)*(y-y1));
    r2:=sqrt((x-x2)*(x-x2)+(y-y2)*(y-y2));
    s:=round(120+A*sin(-2*PI*r1/L)+A*sin(-2*PI*r2/L));
    SetPixel(x,y,rgb(s,0,0)); {Рисование красной точки
                               яркостью s}
  end;
  circle(x1,y1,6);                         {Источники волн}
  circle(x2,y2,6);
end.

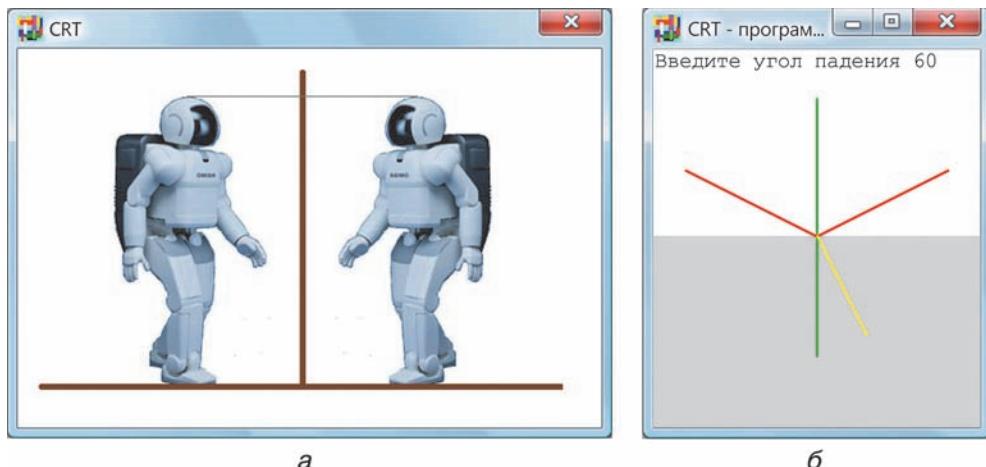
```

Полученная интерференционная картина представлена на рисунке 2.22.

Упражнение

Составьте программу, которая:

- строит изображения предмета в плоском зеркале (расстояние от предмета до зеркала вводится с клавиатуры);
- демонстрирует отражение и преломление света на плоской поверхности (угол падения вводится с клавиатуры).



ГЛАВА 3

ОБРАБОТКА ИНФОРМАЦИИ В СИСТЕМЕ УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

§ 9. Базы данных и системы управления базами данных

В современном мире количество информации постоянно возрастает. Для упорядоченного хранения и обработки больших объемов информации создаются и используются базы данных.

База данных — это совокупность взаимосвязанных и организованных определенным образом данных, отображающих состояние объектов и отношений между ними в какой-либо предметной области.

Первыми компьютерными базами данных, которые создал человек, были табличные базы данных, где размещалась в основном числовая информация. Затем началось освоение человеком текстовых баз данных. С повышением быстродействия компьютеров и увеличением их внешней и оперативной памяти начали создаваться и использоваться графические и мультимедийные базы данных.

Информация, хранимая в базах данных, должна быть структурирована, т. е. организована определенным образом (рис. 3.1), актуальна и доступна пользователю.



Рис. 3.1

Фактически база данных, наполненная информацией, является файлом.

Среди существующих моделей организации информации в базах данных наибольшее распространение получила **реляционная модель** (от англ. *relation* —

отношение). В этой модели данные организованы в виде совокупности таблиц, между которыми установлены связи. Назначение таких связей и порядок их задания мы рассмотрим в § 11.

Система управления базами данных (СУБД) — это совокупность программных и языковых средств, предназначенных для создания, хранения и обработки баз данных.

В настоящее время специалистами разработано около сотни разных СУБД. Все они могут быть разделены по способу доступа к базам данных на две большие группы: *настольные и серверные*.

Настольные СУБД ориентированы на обслуживание одного пользователя, работающего на определенном компьютере с базами данных в каждый реальный момент времени. К настольным СУБД относят Microsoft Access, Paradox, dBase, FoxPro.

Недостатки настольных СУБД стали очевидными, когда возникла необходимость одновременной работы с ними большого числа пользователей. Поэтому следующим этапом развития СУБД стало создание серверных СУБД.

Серверные СУБД используют принцип централизованного хранения и обработки данных, который основан на архитектуре клиент-сервер. СУБД, хранящая данные, и прикладная программа, интерпретирующая эти данные, являются разными приложениями, т. е. существуют приложение-сервер и приложение-клиент. К серверным СУБД относят Microsoft SQL Server, Informix, Sybase, DB2, Oracle и др.

В этой главе мы рассмотрим популярную и широко используемую СУБД Access 2003.

Программа Access после ее установки может загружаться с **Рабочего стола** двойным щелчком мыши по ярлыку  или через систему меню: **Пуск → Программы → Microsoft Office → Microsoft Access**.

После запуска СУБД Access с помощью команды **Файл → Открыть** можно открыть базу данных. Опишем основные элементы интерфейса открывшегося окна (рис. 3.2).

На панели **Стандартная** размещаются основные инструменты, используемые при работе с базой данных. Некоторые инструменты этой панели аналогичны инструментам панели **Стандартная** текстового редактора Word. Часть инструментов предназначена исключительно для работы с базой данных. Применение этих инструментов мы рассмотрим в дальнейшем.

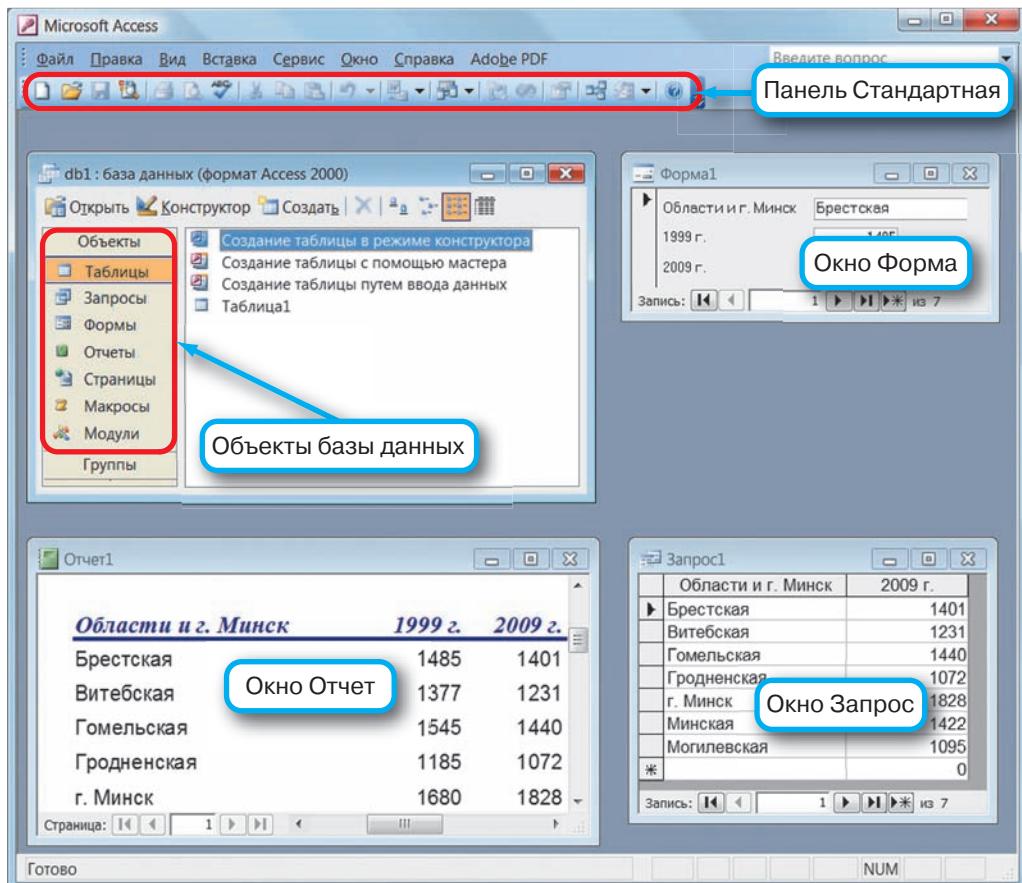


Рис. 3.2

В окне **База данных** расположены ее объекты: **Таблицы**, **Запросы**, **Формы**, **Отчеты** и др. Все эти объекты, если они создавались, хранятся в общем файле базы данных на диске с расширением .mdb. Работа с каждым из объектов осуществляется в своем окне, как показано на рисунке 3.2.

Рассмотрим назначение каждого объекта.

Объект **Таблицы** является основным объектом базы данных. Он предназначен для хранения данных. На основе таблицы (рис. 3.3) создаются остальные объекты базы данных.

Как правило, для решения сложных задач одной таблицы недостаточно. Поэтому в базе данных обычно присутствует сразу несколько таблиц, связанных между собой.

Имена полей

	Области и г. Минск	1999 г.	2009 г.
Запись 1	► Брестская	1485	1401
Запись 2	Витебская	1377	1231
Запись 3	Гомельская	1545	1440
Запись 4	Гродненская	1185	1072
Запись 5	г. Минск	1680	1828
Запись 6	Минская	1559	1422
Запись 7	Могилевская	1214	1095
*		0	0

Рис. 3.3

Запись реляционной базы данных представляет собой строку в таблице, в которой собрана информация об одном объекте базы данных. На рисунке 3.3 в таблице таких записей семь.

Поле реляционной базы данных — это столбец таблицы, который содержит значения определенного свойства об объектах базы данных. В таблице на рисунке 3.3 три поля: **Области и г. Минск, 1999 г., 2009 г.**.

Для описания поля используются следующие характеристики: имя, тип, размер, формат данных поля.

В разных полях таблицы могут храниться данные разных типов: текстовые, числовые, логические и др. Но в одном поле таблицы содержатся данные только одного типа. Каждая таблица имеет свою определенную структуру.

Более полное описание типов полей приведено в § 10.

Объект **Запросы** предоставляет возможность отбора данных из таблиц на основании определенных условий. На рисунке 3.2 представлен запрос с именем **Запрос 1**.

Объект **Формы** отображает данные из таблиц или запросов. С помощью форм удобно вводить данные в таблицы. На рисунке 3.2 представлена форма с именем **Форма 1**.

Объект **Отчеты** предназначен для создания документа, который отражает информацию, отобранную из базы данных. Отчет можно просмотреть на экране или распечатать на принтере. На рисунке 3.2 представлен отчет с именем **Отчет 1**.

- 1. Что называют базой данных?
- 2. Что называют записью и полем в таблице базы данных?
- 3. Какие системы называют системами управления базами данных?
- 4. Какие объекты присутствуют в окне **База данных MS Access**? Для чего предназначены эти объекты?

§ 10. Создание таблицы базы данных

10.1. Проектирование базы данных

Изучение возможностей СУБД Access начнем с проектирования базы данных. Выделяют следующие этапы создания базы данных:

- создание таблиц базы данных, включающее разработку структуры каждой таблицы и ввод в эти таблицы необходимой информации;
- определение связей между несколькими таблицами.

Пример. Спроектировать базу данных «Библиотека» (файл Library book.mdb), содержащую сведения о выдаче книг из библиотеки пользователя.

Основной задачей при использовании базы данных «Библиотека» является отслеживание выдачи книг всем клиентам.

Проведя анализ необходимой для хранения информации, попытаемся сначала расположить ее в одной таблице, поля которой разделим на три группы: «Сведения о клиентах библиотеки», «Сведения о книгах» и «Сведения о выдаче книг». Таблица базы данных в этом случае должна иметь следующую структуру:

Имя поля	Тип данных	
Фамилия	Текстовый	Сведения о клиентах библиотеки
Имя	Текстовый	
Адрес	Текстовый	
Телефон	Текстовый	
Электронная почта	Текстовый	
Автор книги	Текстовый	Сведения о книгах
Название книги	Текстовый	
Год издания	Числовый	
Стоимость книги	Денежный	Сведения о выдаче книг
Дата выдачи	Дата/время	
Отметка о возврате	Логический	

Работать с такой таблицей достаточно неудобно. Например, при выдаче нескольких книг одному клиенту будет необходимо многократно повторять инфор-

мацию о нем: фамилию, имя, адрес и т. д., что приведет к неоправданному увеличению размера таблицы и может повысить вероятность появления ошибок при вводе информации.

Для повышения эффективности при работе с создаваемой базой разделим одну таблицу на три: «Клиенты», «Книги», «Выдача книг». Опишем структуру каждой таблицы.

Таблица «Клиенты»

Имя поля	Тип данных
Код клиента	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Адрес	Текстовый
Телефон	Текстовый
Электронная почта	Текстовый

Таблица «Книги»

Имя поля	Тип данных
Код книги	Числовой
Автор книги	Текстовый
Название книги	Текстовый
Год издания	Числовой
Стоимость книги	Денежный

Таблица «Выдача книг»

Имя поля	Тип данных
Код выдачи	Числовой
Код клиента	Числовой
Код книги	Числовой
Дата выдачи	Дата/время
Отметка о возврате	Логический

В таблице «Клиенты» используется тип данных **Счетчик**. Он применяется для хранения целых числовых значений, которые Access увеличивает при переходе к каждой новой записи. **Счетчик** может использоваться в качестве уникального идентификатора записи таблицы, в которой нет другой величины такого типа. В нашем случае **Код клиента** будет уникальным идентификатором, который позволит легко отличить одного клиента от другого.

СУБД Access допускает также использование типов данных, описанных в следующей таблице.

Тип данных	Описание
Числовой	Поле числового типа, которое содержит произвольные числовые значения.
Текстовый	Поле текстового типа, содержащее текстовые, числовые данные, не требующие вычислений. Длина поля не превосходит 255 символов.
Поле МЕМО	Поле текстового типа, в котором размещаются данные большого объема. Длина поля не превосходит 65 535 символов.
Дата/время	Поле, содержащее дату и время в различных форматах.
Денежный	Поле, имеющее числовые денежные значения.
Логический	Поле, содержащее логические значения.
Гиперссылка	Поле-ссылка на некоторый документ или файл.
Объект OLE	Поле, содержащее документы различных типов (документы Word, таблицы Excel, точечные рисунки и др.).

Перед созданием таблиц в СУБД Access необходимо создать новую базу данных. Это можно сделать с помощью последовательности команд: **Файл** → **Создать** → **Новая база данных** → в окне **Файл новой базы данных** указать имя файла базы, например Library book.

СУБД Access позволяет создавать структуру таблицы тремя способами: в режиме **Конструктора**, с помощью **Мастера** или путем ввода данных (режим **Таблицы**) (рис. 3.4).

В режиме **Таблицы** данные вводят в пустую таблицу.

Рассмотрим подробнее работу режимов **Мастер** и **Конструктор**.

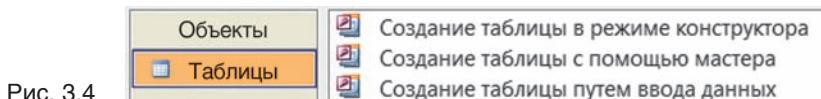


Рис. 3.4

10.2. Создание структуры таблицы

В режиме **Мастера** удобно создавать структуру таблицы, в которой используются стандартные имена полей и типы данных в этих полях. **Мастер** вначале предлагает выбрать образец таблицы, а затем отобрать поля для этой таблицы.

Пример 1. Создать в режиме **Мастера** структуру таблицы «Клиенты».

Выберем объект **Таблицы** в окне **База данных** и дважды щелкнем мышью по строке **Создание таблицы с помощью мастера**.

Выберем образец таблицы **Клиенты**. Образцы полей перенесем с помощью кнопки из окна **Образцы полей** в окно **Поля новой таблицы**, как показано на рисунке 3.5. Имя перенесенного поля может быть изменено с помощью кнопки **Переименовать поле**. Для перехода к следующему шагу щелкнем мышью по кнопке **Далее >**.

Зададим имя таблицы, например **Клиенты**, и установим флагок **Пользователь определяет ключ самостоятельно**. Нажмем кнопку **Далее**. Выберем поле с уникальными для каждой записи данными, для нашего примера — это поле **Код**

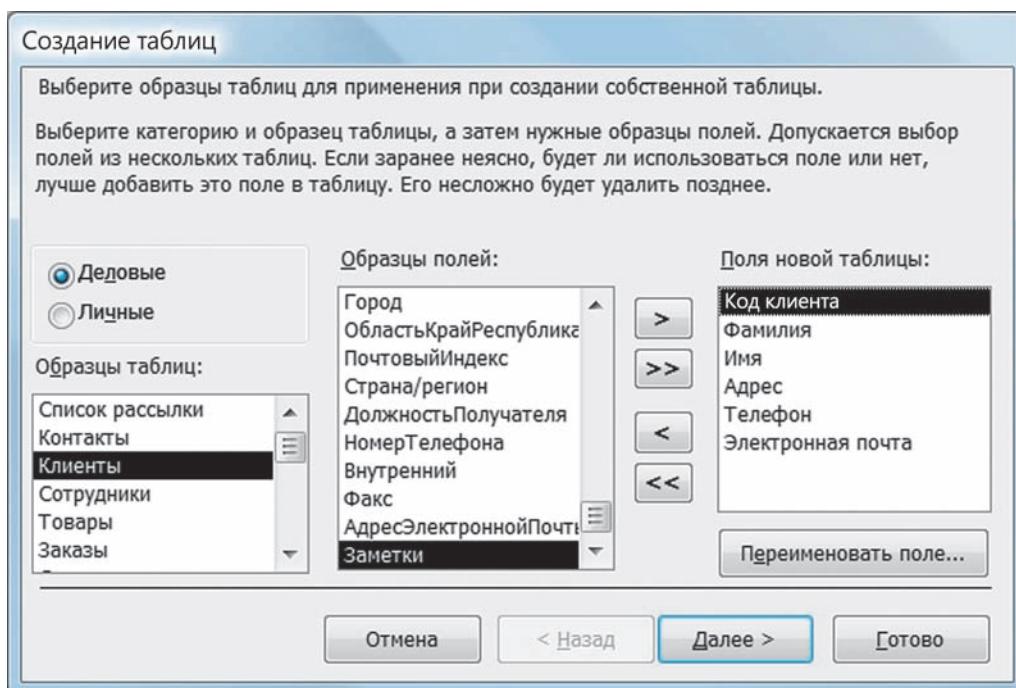


Рис. 3.5

Клиенты : таблица						
	Код клиента	Фамилия	Имя	Адрес	Телефон	Электронная почта
▶	(Счетчик)					
Запись: 1 из 1						

Рис. 3.6

клиента. Нажмем кнопку **Готово**. В результате откроется окно таблицы «Клиенты», готовой для заполнения информацией (рис. 3.6).

Щелкнув мышью по кнопке **Конструктор** Конструктор (см. рис. 3.2), просмотрим структуру созданной с помощью **Мастера** таблицы «Клиенты» (рис. 3.7).

Клиенты : таблица	
Имя поля	Тип данных
Код клиента	Счетчик
Фамилия	Текстовый
Имя	Текстовый
Адрес	Текстовый
Телефон	Текстовый
Электронная почта	Текстовый

Рис. 3.7

Желательно, чтобы каждая таблица имела **ключ** — одно или несколько полей, содержимое которых уникально для каждой записи. Для таблицы «Клиенты» таким является поле **Код клиента**. Тип данных этого поля — **Счетчик**. В нашей таблице каждый клиент будет иметь уникальный номер и никакие две записи не будут одинаковыми.

Мастер определяет тип данных полей самостоятельно в зависимости от названия без предварительного определения этих типов пользователем. В дальнейшем в режиме **Конструктора** тип поля может быть изменен.

Режим **Конструктора** предоставляет пользователю возможность самостоятельно описывать и изменять структуру таблицы.

Пример 2. Создать в режиме **Конструктора** структуру таблицы «Книги».

Двойным щелчком мыши по строке **Создание таблицы в режиме конструктора** (см. рис. 3.4) откроем окно **Таблица 1**, в котором необходимо описать поля.

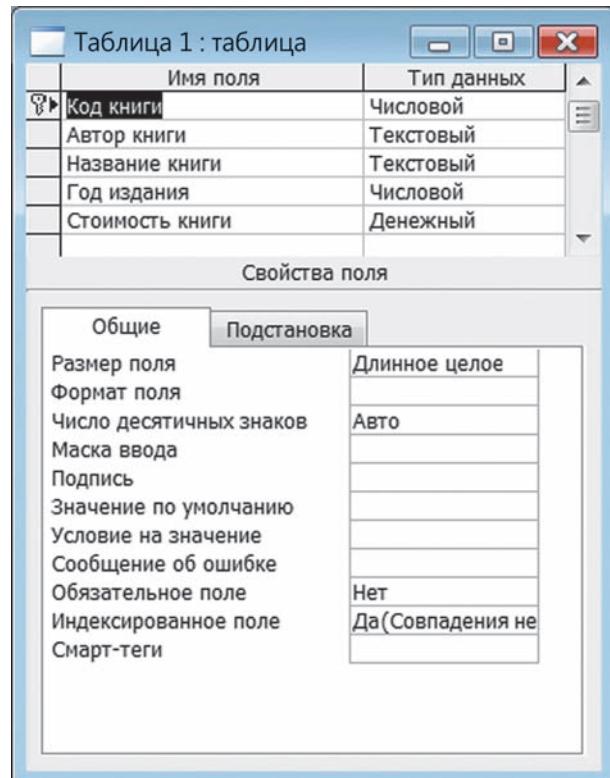


Рис. 3.8

Таблица «Книги» будет содержать пять полей. Введем для каждого из них имя и тип данных. В нижней части окна определим свойства поля, как показано на рисунке 3.8.

При вводе типов данных и свойств полей будем применять раскрывающиеся списки (см. рис. 3.8).

В таблице «Книги» ключевым является поле **Код книги**. Чтобы это поле стало ключевым, достаточно в режиме **Конструктора** поместить курсор в выбранное поле и нажать кнопку **Ключевое поле** на панели **Базы данных** или выполнить команду **Правка → Ключевое поле**.

Если необходимо определить сразу несколько ключевых полей, то следует выделить нужные строки и нажать кнопку **Ключевое поле**. Повторные аналогичные действия отменяют признак ключевого поля.

Завершив описание таблицы, сохраним ее с помощью последовательности команд: **Файл → Сохранить как** → указать имя таблицы, например **Книги → OK**.

The screenshot shows a Microsoft Access database table named 'Клиенты : таблица'. The table has columns: Код клиента (Client ID), Фамилия (Last Name), Имя (First Name), Адрес (Address), Телефон (Phone), and Электронная почта (Email). There are 5 records displayed:

Код клиента	Фамилия	Имя	Адрес	Телефон	Электронная почта
1	Сидоров	Василий	ул. Плеханова, д. 20	2954321	Sidor@tut.by
2	Симончик	Валентина	ул. Васнецова, д.3, кв.89	2961736	sim@mail.ru
3	Титов	Алексей	ул. Жилуновича, д.6	2474331	Titov@tut.by
4	Смирнов	Александр	ул. Ванеева, д.6, кв. 5	2912345	mal@tut.by
5	Иванов	Павел	ул. Кошевого, д.13, кв.5	2916754	Pivan@mail.ru

Record 1 is selected. Navigation buttons at the bottom left indicate the current record is 1 of 5.

Рис. 3.9

Созданные нами таблицы «Клиенты» и «Книги» добавляются в окно **База данных**.

Таблица «Выдача книг» также может быть создана в режиме **Конструктора**.

Создав структуру таблицы, пользователь может в режиме **Таблицы** приступить к ее заполнению. Для этого в окне **Базы данных** надо дважды щелкнуть мышью по названию таблицы, например «Клиенты».

Ввод данных в таблицу и их редактирование осуществляются непосредственно в ячейках таблицы (рис. 3.9). Действия по вводу и редактированию аналогичны соответствующим действиям в MS Excel.

Для удаления целой записи в таблице необходимо в режиме **Таблица** установить курсор в любое поле удаляемой записи и щелкнуть мышью по кнопке **Удалить запись** . Для удаления сразу нескольких записей необходимо вначале их выделить. При удалении целой записи в таблице «Клиенты» поле **Код клиента** теряет уникальный номер клиента для этой записи в таблице.

Добавление записей в таблицу осуществляется в режиме **Таблица**. При этом новая запись размещается в конце таблицы.

После завершения ввода данных в таблицу или их редактирования таблица сохраняется с помощью команды **Файл → Сохранить**.



1. Что необходимо для проектирования базы данных?
2. Какие режимы используются для создания структуры таблицы?
3. Что понимается под ключевым полем в таблице? Для чего применяется ключевое поле?
4. Какие типы данных допускается использовать в СУБД Access?
5. Для чего используется тип данных **Счетчик**?

Упражнения

1. Создайте базу данных «Города Могилевской области» (файл MogilevCity.mdb), содержащую таблицу «Города_Могилевской_области».

Города_Могилевской_области : таблица				
	Города области	Население (тыс. чел.)	Основные реки	Основание города
▶	Бобруйск	219	Березина	1387
	Быхов	16,1	Днепр	1430
	Горки	34	Проня	1544
	Климовичи	14,5	Калиница	1581
	Кричев	26,8	Сож	1136
	Могилев	372	Днепр	1267
	Мстиславль	11,5	Вихра	1135
	Осиповичи	33,6	Свислочь	1872
	Славгород	8,2	Сож	1136
	Чаусы	10,4	Проня	1581
	Шклов	17,8	Днепр	1535
*				
Запись: [◀] [◀] 1 [▶] [▶] * из 11				

2. Создайте базу данных «Библиотека» (файл Library book.mdb), содержащую три таблицы: «Клиенты» (см. рис. 3.9), «Книги», «Выдача Книг». Образцы оформления таблиц «Книги» и «Выдача книг» приведены на следующих рисунках:

Книги : таблица					
	Код книги	Автор книги	Название книги	Год издания	Стоимость книги
+	17	В. Быков	Собр. сочинений. Т. 3	2001	15 000,00р.
♂	31	Я. Мавр	Полесские робинзоны	1998	10 000,00р.
+	67	Ж. Верн	Плавучий остров	1990	18 000,00р.
+	105	Л. Толстой	Война и мир	2004	7 000,00р.
+	203	А. Кристи	Собр. сочинений. Т. 10	2008	6 000,00р.
+	465	Н. Кун	Легенды и мифы	2002	9 000,00р.

Выдача книг : таблица					
	Код выдачи	Код клиента	Код книги	Дата выдачи	Отметка о возврате
▶	1	4	203	19.02.2009	<input checked="" type="checkbox"/>
	2	5	17	20.02.2009	<input checked="" type="checkbox"/>
	3	3	31	21.02.2009	<input checked="" type="checkbox"/>
	4	2	67	24.02.2009	<input checked="" type="checkbox"/>
	5	5	465	13.03.2009	<input type="checkbox"/>
	6	1	203	15.03.2009	<input type="checkbox"/>
	7	1	105	24.04.2009	<input type="checkbox"/>



3. Создайте базу данных «Экзамены» (файл Mark.mdb), содержащую четыре таблицы: «Учителя», «Ученики», «Предметы», «Экзамен». Образцы оформления таблиц представлены на следующих рисунках:

Учителя : таблица

	КодУчителя	Фамилия	Имя	Отчество	Адрес
+	1	Абрикосов	Андрей	Андреевич	г. Пинск, ул. Сидоренко, д. 5, кв. 10
▶	2	Ламусова	Вера	Васильевна	д. Сноба, Пинский р-н, д. 16
+	3	Петров	Игорь	Викторович	г. Пинск, ул. Мира, д. 34, кв. 1
*	(Счетчик)				

Запись: [◀◀] [◀] 2 [▶] [▶▶] [✳] из 3

Ученики : таблица

	КодУченика	Фамилия	Имя	Отчество	Класс	Адрес
▶	1	Акулич	Светлана	Игоревна	11 А	г. Пинск, ул. Красивая, д. 20
+	2	Ананич	Андрей	Иванович	11 А	г. Пинск, ул. Народная, д. 16, кв. 2
+	3	Бирюров	Павел	Сидорович	11 Б	г. Пинск, ул. Плеханова, д. 5, кв. 32
+	4	Брюсова	Анна	Ивановна	11 Б	г. Пинск, ул. Васнецова, д. 20, кв. 65
+	5	Булгакова	Кира	Яновна	11 А	г. Пинск, ул. Васнецова, д. 3, кв. 16
*	(Счетчик)					

Запись: [◀◀] [◀] 1 [▶] [▶▶] [✳] из 5

Предметы : таблица

	КодУчителя	КодПредмета	Наименование
▶		1	1 Белорусская литература
		1	2 Белорусский язык
	2		3 География
	3		4 Биология

Запись: [◀◀] [◀] 1 [▶] [▶▶] [✳] из 4

Экзамен : таблица

	КодУченика	КодПредмета	Отметка	Дата сдачи
			1	19.01.2009
			2	03.01.2009
			3	10.06.2009
			4	12.06.2009

Запись: [◀◀] [◀] 5 [▶] [▶▶] [✳] из 5

§ 11. Связывание таблиц базы данных

Установление связей в MS Access дает возможность автоматически соединять данные из разных таблиц. Так обеспечивается целостность базы данных.

Процесс установления связей между таблицами называют **построением Схемы данных**.

Для установления связей между двумя таблицами необходимо определить в каждой из них поля для связывания. **Эти поля могут иметь одинаковые или разные имена, но должны содержать данные одного типа!**

Рассмотрим связь между таблицами «Сотрудники» (рис. 3.10) и «Ведомость заработной платы» (рис. 3.11) в базе данных Salary.mdb. Структура каждой из этих таблиц представлена на рисунке 3.12.

	Табельный номер	Фамилия	Имя	Отчество	Адрес
▶ +	1	Алтухов	Павел	Семенович	ул. Некрасова, 23
▶ +	2	Артемкина	Алла	Леонидовна	ул. Плеханова, д.3, кв 5
▶ +	3	Белоус	Александр	Александрович	пр. Партизанский, д.43. кв. 1
▶ +	4	Бутик	Елена	Евгеньевна	ул. Бельского, д.34, кв. 67
▶ +	5	Веревка	Петр	Олегович	ул. Пешеходная, д.23
▶ +	6	Кровель	Ирина	Васильевна	ул. Народная, д.13, кв. 65
▶ +	7	Кишмяк	Сергей	Владимирович	ул. Плеханова, д. 46, кв. 78
▶ +	8	Рогова	Зоя	Петровна	пос. Озерный, д. 6

Рис. 3.10

	Код выдачи	Табельный номер	Дата выдачи	Размер	Налог
▶ +	1	1	04.09.2009	780 000,00р.	93 600,00р.
▶ +	2	2	04.09.2009	820 000,00р.	98 400,00р.
▶ +	3	3	05.09.2009	910 000,00р.	109 000,00р.
▶ +	4	4	04.09.2009	560 000,00р.	67 200,00р.
▶ +	5	5	06.09.2009	435 000,00р.	52 200,00р.
▶ +	6	6	04.09.2009	740 000,00р.	88 800,00р.
▶ +	7	7	06.09.2009	680 000,00р.	81 600,00р.
▶ +	8	8	05.09.2009	420 000,00р.	50 400,00р.
*	0	0		0,00р.	0,00р.

Рис. 3.11



Рис. 3.12

В таблице «Сотрудники» ключевым является поле **Табельный номер**. Данное поле имеет тип **Счетчик** и содержит уникальные значения для каждой записи этой таблицы. Назовем это поле *первичным ключом*.

Если в таблице «Ведомость заработной платы» каждый сотрудник может быть записан только один раз, то ключевым полем этой таблицы также может являться поле **Табельный номер**. В таблице «Ведомость заработной платы» ключевое поле **Табельный номер** будем называть *внешним ключом*.

В этом случае связь, установленную между первичным и внешним ключами, называют связью *один к одному*. Этот тип связи представлен на рисунке 3.13.



Рис. 3.13

Если в таблице «Ведомость заработной платы» один сотрудник может быть записан несколько раз, то поле **Табельный номер** уже не будет уникальным, так как может повторяться многократно. Тогда тип данных в этом поле может принимать числовые значения, а ключевым полем с уникальными значениями может быть определено поле **Код выдачи**.

В этом случае связь, установленную между одноименными полями **Табельный номер** в обеих таблицах, называют связью *один ко многим*. Этот тип связи представлен на рисунке 3.14.



Рис. 3.14

Пример. Построить Схему данных для базы данных «Библиотека», как показано на рисунке 3.15.

Откроем многотабличную базу данных «Библиотека», для которой будут устанавливаться связи между таблицами.

Щелкнем мышью по значку Схема данных  (см. рис. 3.2).

В окне Добавление таблицы выделим первую таблицу, для которой устанавливается связь. В нашем примере — это таблица «Клиенты».

Щелкнем мышью по кнопке Добавить. На экране будет отображена структура таблицы, которую мы добавили. Затем таким же образом добавим в Схему данных остальные связываемые таблицы («Выдача книг», «Книги») и щелкнем мышью по кнопке Закрыть.

В окне таблицы «Клиенты» щелкнем мышью по полю, которое будет использоваться для установления связи, например Код клиента, и перетащим его на одноименное поле таблицы «Выдача книг».

В окне Изменение связей убедимся, что связаны необходимые поля, и установим флажок Обеспечение целостности данных. В результате выполненных действий в окне Схема данных два связанных поля соединятся линией, как показано на рисунке 3.15.



Рис. 3.15

Аналогичным образом связем таблицы «Выдача книг» и «Книги» по полю **Код книги**.

Для сохранения созданных связей щелкнем мышью по кнопке **Сохранить** и закроем окно **Схема данных**. При закрытии окна **Схема данных** связи между таблицами сохраняются.

Созданные связи в дальнейшем можно удалить в окне **Схема данных**, выделив щелчком мыши линию связи и нажав клавишу Delete (Удалить). Двойной щелчок мышью по линии связи приводит к открытию окна **Изменение связей**, в котором связи могут быть отредактированы.

- ?** 1. Для чего необходимо связывать таблицы?
- 2. Какими свойствами должны обладать связываемые поля?
- 3. Какие типы связей между полями таблиц существуют?

Упражнение

Установите связи между таблицами баз данных, предложенных учителем:

- а) база данных «Библиотека» (файл Library book.mdb) с таблицами «Клиенты», «Выдача книг», «Книги»;
- б) база данных «Экзамены» (файл Mark.mdb) с таблицами «Учителя», «Ученики», «Предметы», «Экзамен».





§ 12. Создание и заполнение формы

Вы уже знаете, что данные таблицы пользователь вводит непосредственно в ее ячейки. Если его не устраивает табличный способ размещения данных, он может подготовить **форму**. В форме пользователь располагает данные на экране компьютера в порядке, удобном ему для работы. В созданной форме можно вводить, редактировать и просматривать данные.

Формы можно создавать на основе готовой таблицы или запроса. Рассмотрим, как создаются формы на основе таблиц.

СУБД Access предлагает три способа создания форм: **Создание формы в режиме мастера**, **Создание формы в режиме конструктора**, **Автоформы**.

При подготовке формы необходимо в окне **Базы данных** выполнить команду **Формы → Создать** или нажать кнопку **Создать**.

В открывшемся окне **Новая форма** предлагаются два основных способа **Конструктор** и **Мастер форм**, а также **Автоформы**: **в столбец**, **ленточная**, **табличная** и др. (рис. 3.16). Быстрое создание форм обеспечивают **Мастер форм** и **Автоформы**.

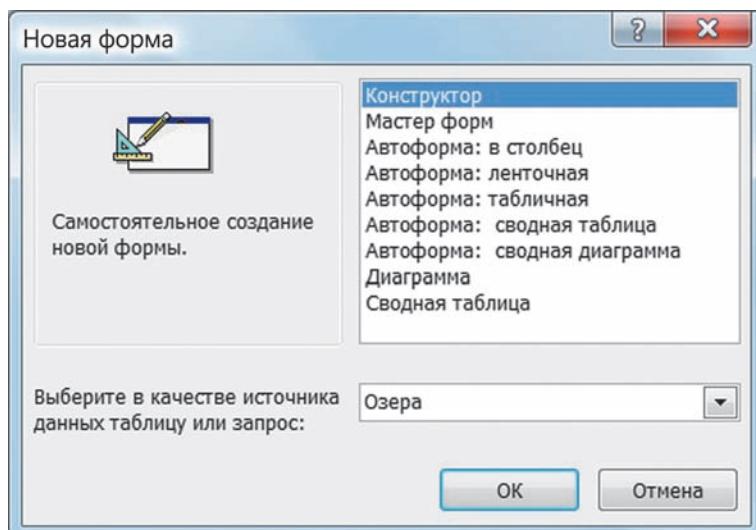


Рис. 3.16

Рассмотрим на примере работу с **Мастером форм**.

Пример 1. Создать с помощью **Мастера форм** по данным таблицы «Города_Могилевской_области» ленточную форму со стилем **Солнцестояние**, содержащую данные, показанные на рисунке 3.17.

Города_Могилевской_области

Города области	Население (тыс. чел.)	Основание города
Бобруйск	219	1387
Быхов	16,1	1430
Горки	34	1544
Климовичи	14,5	1581
Кричев	26,8	1136

Запись: |<|<|>|>>|>>>| из 11 |<|>|>>|>>>|

Рис. 3.17

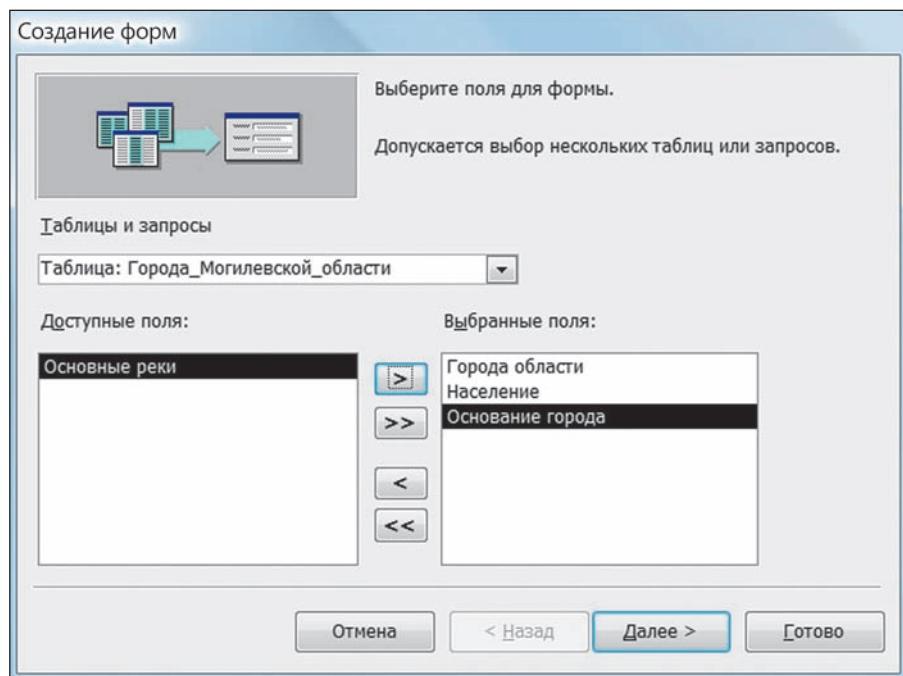


Рис. 3.18

Для создания формы с помощью **Мастера форм** дважды щелкнем мышью по строке **Создание форм с помощью мастера** в окне **База данных**. **Мастер форм** в несколько этапов откроет окно **Создание форм**, в котором мы зададим имя таблицы для формы и выберем доступные поля, как показано на рисунке 3.18, а затем определим внешний вид формы: выберем таблицу «Города_Могилевской_области» и три поля **Города области**, **Население (тыс. чел.)**, **Основание города**, внешний вид формы — **Ленточный**.

В дальнейшем в открывшемся окне укажем стиль формы **Солнцестояние**, зададим имя формы и щелкнем мышью по кнопке **Готово**.

В окне **Создание форм** можно открыть готовую форму для ввода данных.

Заметим, что, создавая форму, из таблицы можно переносить не все поля. При заполнении формы данные автоматически размещаются в ячейках таблицы, на основании которой данная форма создавалась.

Создание формы с помощью **Конструктора** является более сложным процессом, чем ее подготовка с помощью **Мастера форм**.

Пример 2. На основании таблицы «Города_Могилевской_области» создать форму в режиме **Конструктора**.

Для этого в окне **База данных** выберем пункт **Формы**. Затем нажмем кнопку **Создать** меню этого окна. В окне **Новая форма** выберем пункт меню **Конструктор**, а также источник данных — таблицу «Города_Могилевской_области», на основании которой строится форма.

В результате выполненных действий откроется окно для конструирования формы и специальное окно со списком полей таблицы (рис. 3.19).

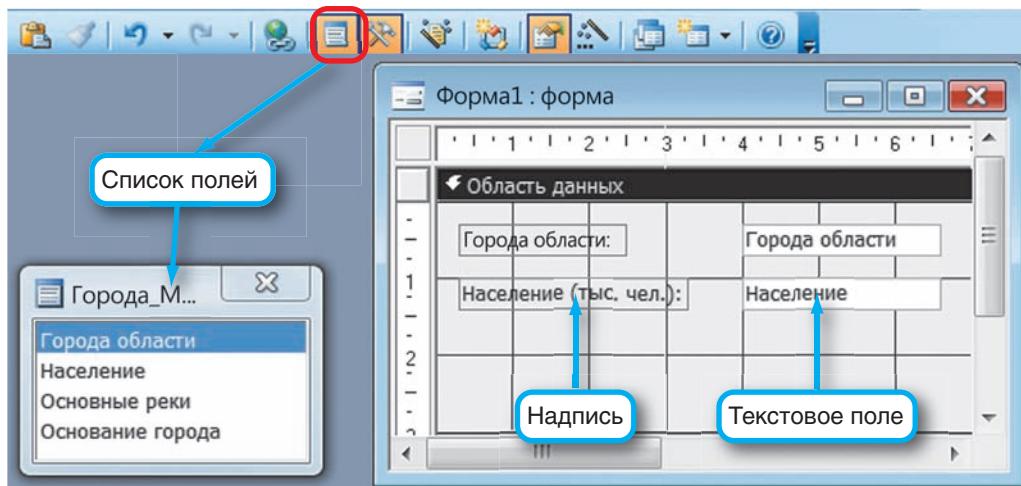


Рис. 3.19

Перенесем с помощью мыши из **Списка полей** в **Область данных** необходимые поля таблицы, например **Города области**, **Население (тыс. чел.)**, **Основание города**.

При переносе каждого поля в **Область данных** в нем располагаются **Надпись** (имя поля) и **Текстовое поле** (данные).

Завершим конструирование формы ее сохранением.

Формы, созданные с помощью **Мастера форм**, могут быть изменены в режиме **Конструктора**.

- ?** 1. Почему пользователи для ввода данных предпочитают применять форму вместо таблицы?
- 2. Какие способы создания формы предлагает программа Access?

Упражнение

Откройте готовую базу данных, предложенную учителем. Создайте формы по готовым таблицам, используя **Мастер форм** или **Конструктор**:

а) для базы данных «Тепловые электростанции Беларуси» (файл Ener.mdb), содержащей таблицу «Электростанции» со сведениями о крупных тепловых электростанциях нашей страны;

Таблица «Электростанции»

Электростанции : таблица				
	Электростанция	Мощность (тыс. кВт)	Используемое топливо	Удельный расход топлива (г/кВт*ч)
▶	Лукомльская ГРЭС	2500	мазут, природный газ	318
	Березовская ГРЭС	950	мазут	360
	Минская ТЭЦ-4	900	мазут, природный газ	235
	Новополоцкая ТЭЦ	505	мазут	220
	Минская ТЭЦ-3	395	природный газ, мазут	245
	Гомельская ТЭЦ-2	360	природный газ, мазут	236
	Могилевская ТЭЦ-2	345	мазут	254
	Светлогорская ТЭЦ	270	попутный газ	313

Форма «Электростанции»

Электростанция	Мощность (тыс. кВт)
Лукомльская ГРЭС	2500
Березовская ГРЭС	950
Минская ТЭЦ-4	900
Новополоцкая ТЭЦ	505

б) для базы данных «Красная книга Беларуси» (файл tigr.mdb), содержащей таблицу «Красная книга Беларуси» со сведениями о редких видах животного мира нашей страны.

Таблица «Красная книга Беларуси»

Код	Класс	Отряд	Семейство	Вид	Изображение
1	Птицы	Совообразные	Сипуховые	Сипуха	Bitmap Image
2	Млекопитающие	Парнокопытные	Полорогие	Европейский зубр	Bitmap Image
3	Млекопитающие	Хицные	Куницы	Барсук	Bitmap Image
4	Млекопитающие	Хицные	Кошачьи	Рысь европейская	Bitmap Image
5	Птицы	Аистообразные	Аистовые	Черный аист	Bitmap Image

Запись: [Navigation buttons] 2 [Next] из 5

Форма «Млекопитающие и птицы»

Млекопитающие и птицы : форма

Вид: Сипуха

Изображение:

Запись: [Navigation buttons] 1 [Next] из 6

§ 13. Поиск данных с помощью запросов

Поиск информации в базах данных выполняется через запросы. С помощью **запроса** СУБД Access выбирает и отображает наборы записей из таблиц базы данных, которые отвечают заданным условиям.

Запрос может формироваться на основе одной или нескольких связанных таблиц или запросов, построенных ранее.

СУБД Access поддерживает создание запросов с помощью **Мастера** и **Конструктора**. **Мастер** обеспечивает возможность создания простых запросов.

Рассмотрим, как создаются запросы на основе таблиц. Используем для этого таблицу базы данных «Аквариумные рыбки», представленную на рисунке 3.20.

Рис. 3.20

	Название рыбки	Место происхождения	Длина (см)
▶	Гамбузия	Южная Америка	5
	Платипецилус черный	Мексика	4
	Зеленый меченосец	Мексика	11
	Тетра-фон-рио	Южная Америка	2
	Бархатный гирадинус	Южная Америка	5
	Макропод	Китай	9
	Гурами жемчужный	Индия	10
	Лялиус	Индия	5
	Неон	Южная Америка	4

Пример 1. Создать с помощью Конструктора простой запрос, по которому будут представлены сведения о названиях рыбок и местах их происхождения.

Щелкнем мышью по кнопке **Запросы** в окне **Базы данных**.

Дважды щелкнем мышью по строке **Создание запроса в режиме конструктора** или выполним команду **Создать → Конструктор**.

В результате откроются два окна, как показано на рисунке 3.21.

В окне **Добавление таблицы** выберем таблицу с именем «Аквариумные рыбки» и щелкнем по кнопке **Добавить**, а затем — по кнопке **Закрыть**.

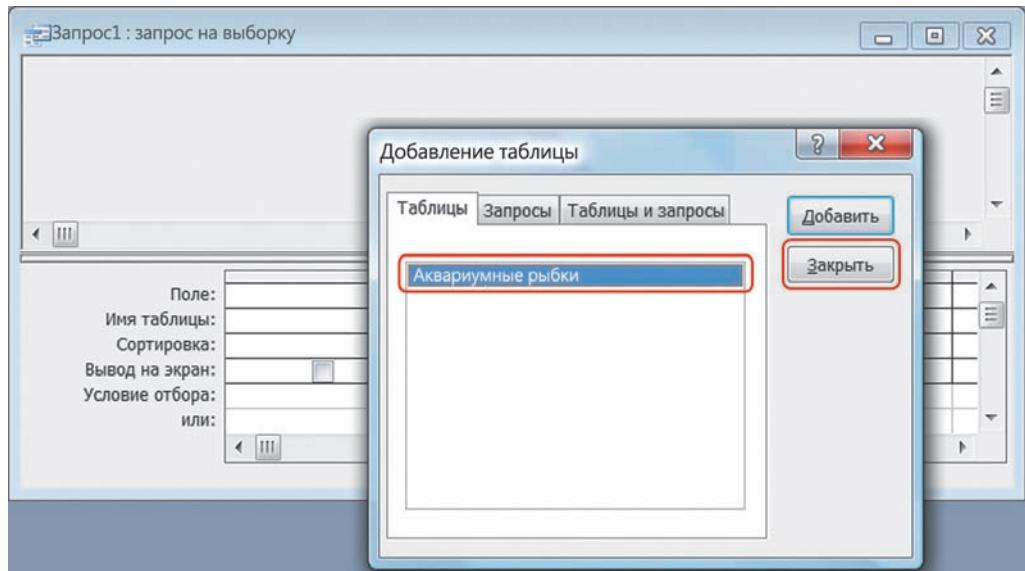


Рис. 3.21

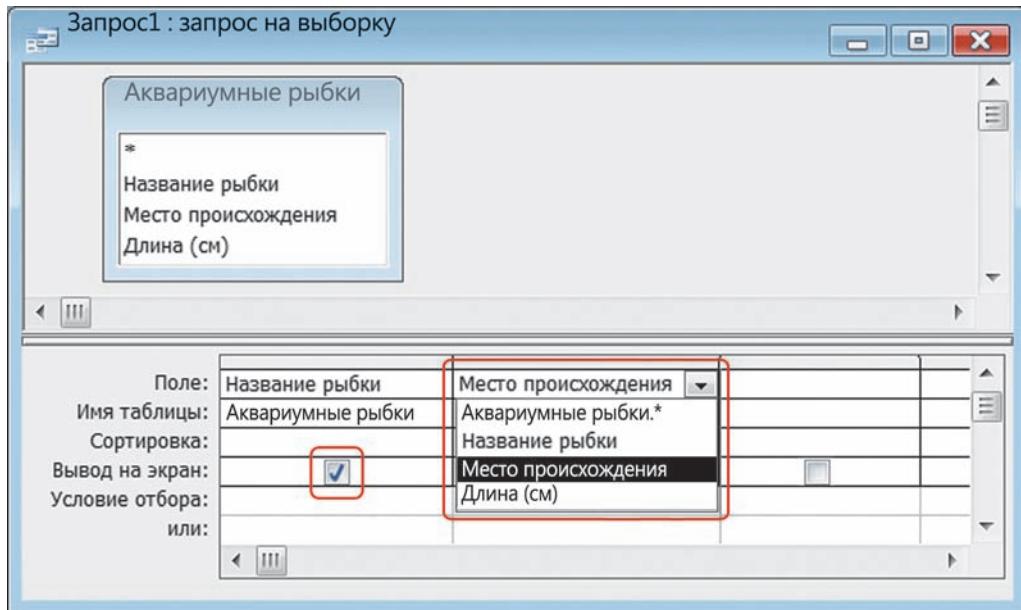


Рис. 3.22

В окне **Запрос на выборку** (рис. 3.22), двигаясь слева направо, в выпадающих списках строк **Поле** выберем поля для добавления в запрос: **Название рыбки**, **Место происхождения**.

Установка флажка в каждом отобранным столбце строки **Вывод на экран** позволяет вывести нужные столбцы. После завершения конструирования запроса сохраним его. Результатом выполнения запроса будет новая временная таблица (рис. 3.23).

Запрос1 : запрос на выборку	
▶ Гамбузия	Южная Америка
Плятипецилиус черный	Мексика
Зеленый меченосец	Мексика
Тетра-фон-рио	Южная Америка
Бархатный гиардинус	Южная Америка
Макропод	Китай
Гурами жемчужный	Индия
Лялиус	Индия
Неон	Южная Америка

Рис. 3.23

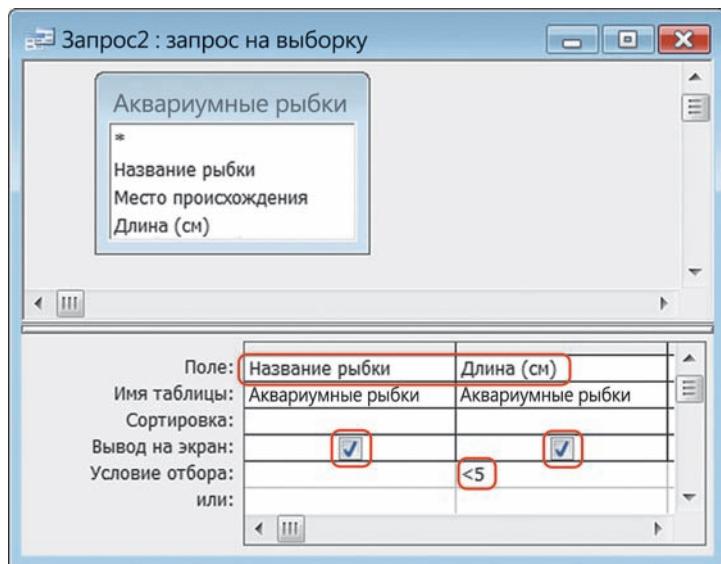


Рис. 3.24

Пример 2. Создать запрос, по которому будут представлены сведения о рыбках, длина которых меньше 5 см.

Откроем в режиме **Конструктора** новое окно для запроса. В строке **Условие отбора** опишем запрос, как показано на рисунке 3.24.

Запрос2 : запрос на выборку			
	Название рыбки	Длина (см)	
▶	Платипецилиус черный	4	
	Тетра-фон-рио	2	
	Неон	4	

Запись: [◀] [◀] [▶] [▶] [*] из 3

Рис. 3.25

отбора таблиц с источниками данных следует указать сразу несколько таблиц, данные из которых будут использоваться в запросе.



1. Для чего используются запросы?
2. Как строится простой запрос с помощью **Конструктора**?
3. Какие возможности предоставляет пользователю **Конструктор** для создания запроса на выборку с условиями?

Результатом выполнения запроса будет новая временная таблица (рис. 3.25).

Наиболее сложным является подготовка запросов сразу по нескольким связанным таблицам. Процесс создания таких запросов напоминает формирование запроса по одной таблице. Однако на этапе

Упражнения

1. Откройте таблицу базы данных «Аквариумные рыбки» и создайте запросы, представленные в примерах 1 и 2 этого параграфа.

2. Откройте таблицу базы данных «Тепловые электростанции Беларуси» и создайте по ней:

а) простой запрос, содержащий записи полей **Электростанция**, **Мощность (тыс. кВт)**, **Используемое топливо**;

б) запрос, имеющий записи с полями **Электростанция** и **Мощность (тыс. кВт)** < «400 тыс. кВт»;

в) запрос с записями **Электростанция** и **Используемое топливо = «мазут»**;

г) запрос с записями **Электростанция** и **Используемое топливо = «мазут» или Используемое топливо = «попутный газ»**.

3. Откройте базу данных «Библиотека» и создайте с помощью **Конструктора** запрос, показанный на рисунке.

Просмотрите таблицу, построенную по данному запросу.

Поле:	Фамилия	Имя	Название книги	Дата выдачи	Отметка о возврате
Имя таблицы:	Клиенты	Клиенты	Книги	Выдача книг	Выдача книг
Сортировка:					
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:					
или:					
	<input type="button" value=" <"/>	<input type="button" value="III"/>			<input type="button" value=" >"/>

§ 14. Сортировка записей в таблице

Для эффективного поиска информации в больших таблицах базы данных, в которых могут храниться сотни и тысячи записей, у пользователя часто возникает необходимость разместить записи в определенной последовательности, т. е. отсортировать их.

Под **сортировкой** записей в таблице базы данных понимают процесс их упорядочения в определенной последовательности по значению одного из полей.

В зависимости от типа данных, определенных для сортировки, все записи в таблице базы данных могут сортироваться:

- по величине числа, если тип данных числовой;
- по алфавиту, если тип данных текстовый (символьный);
- по дате и времени, если данные в поле содержат значения даты и времени.

Для сортировки записей по данным конкретного поля необходимо установить курсор в любой строке соответствующего столбца и щелкнуть мышью по одной из кнопок на панели **Стандартная**:



— сортировка по возрастанию;

— сортировка по убыванию.

Продемонстрируем процесс сортировки записей в таблице «Вулканы», фрагмент которой представлен на рисунке 3.26.

Пример. В таблице «Вулканы» расположить записи поля **Название** в алфавитном порядке.

Установим курсор мыши в любой строке поля **Название**. Щелкнем мышью по кнопке **Сортировка по возрастанию** или выполним последовательность команд: **Записи** → **Сортировка** → **Сортировка по возрастанию**.

Результат сортировки для представленного фрагмента показан на рисунке 3.27.

Вулканы : таблица					
	Название	Континент	Местонахождение	Высота над уровнем моря	Активность
▶	Этна	Европа	Сицилия	3340	<input checked="" type="checkbox"/>
	Везувий	Европа	Апеннины	1277	<input type="checkbox"/>
	Гекла	Европа	Исландия	1491	<input checked="" type="checkbox"/>
	Меру	Африка	Танзания	4567	<input checked="" type="checkbox"/>
	Керинчи	Азия	Хонсю	3805	<input checked="" type="checkbox"/>
	Кракатау	Азия	Камчатка	813	<input type="checkbox"/>
	Семеру	Азия	Минданао	3676	<input checked="" type="checkbox"/>

Рис. 3.26

Вулканы : таблица					
	Название	Континент	Местонахождение	Высота над уровнем моря	Активность
▶	Везувий	Европа	Апеннины	1277	<input type="checkbox"/>
	Гекла	Европа	Исландия	1491	<input checked="" type="checkbox"/>
	Керинчи	Азия	Хонсю	3805	<input checked="" type="checkbox"/>
	Кракатау	Азия	Камчатка	813	<input type="checkbox"/>
	Меру	Африка	Танзания	4567	<input checked="" type="checkbox"/>
	Семеру	Азия	Минданао	3676	<input checked="" type="checkbox"/>
	Этна	Европа	Сицилия	3340	<input checked="" type="checkbox"/>

Рис. 3.27



1. Для чего используется сортировка данных?
2. Какой процесс называют сортировкой записей таблицы базы данных?
3. Что необходимо сделать для выполнения сортировки записей в таблице базы данных?

Упражнения

1. Откройте таблицу «Клиенты» базы данных «Библиотека», предложенную учителем, выполните сортировку записей этой таблицы, разместив фамилии клиентов в алфавитном порядке.
2. Откройте базу данных «Города Могилевской области» (MogilevCity.mdb), содержащую одноименную таблицу, выполните сортировку записей этой таблицы:
 - а) записи поля **Города области** расположите в алфавитном порядке;
 - б) записи поля **Население (тыс. чел.)** упорядочьте по возрастанию.
3. Создайте базу данных «Реки Беларуси», фрагмент таблицы которой представлен на рисунке.

Реки : таблица

Название	Площадь бассейна (кв. км)	Длина (км)	Густота речной сети (км/кв. км)
Ловать	21900	536	0,45
Днепр	504000	2201	0,39
Вилия	25100	498	0,44
Западный Буг	39400	772	0,42
Сож	42140	648	0,38
Припять	121000	761	0,42
Березина	24500	613	0,35
Неман	98200	937	0,47
► Западная Двина	37900	1020	0,45

Выполните сортировку записей этой таблицы:

- а) записи поля **Длина (км)** упорядочьте по возрастанию;
- б) записи поля **Густота речной сети (км/кв. км)** расположите в порядке убывания.

§ 15. Создание отчетов

Отчеты являются удобным и гибким способом просмотра и распечатки итоговых сведений из базы данных.

Под **отчетом** понимается документ, содержание которого формируется по определенному запросу на основе информации, размещенной в базе данных.

В отчетах данные представляются в удобном виде. Они могут быть отсортированы, особым образом сгруппированы, а также могут содержать итоговые значения и т. д.

СУБД Access предлагает пользователю несколько способов создания отчета: **Автоотчет**, **Мастер отчетов** и **Конструктор**. Отчеты являются самостоятельными объектами базы данных.

После просмотра полученного отчета он может быть сохранен или распечатан.

Рассмотрим на примере многотабличной базы данных «Библиотека», как с помощью **Мастера** создаются отчеты.

Пример. Создать отчет, содержащий сведения из трех таблиц базы данных «Библиотека» (см. § 10). Из таблицы «Клиенты» выбрать фамилию и имя клиента, из таблицы «Книги» — автора и название книги, а из таблицы «Выдача книг» — отметку о возврате.

Откроем базу данных «Библиотека» и дважды щелкнем мышью по строке **Создание отчета с помощью мастера** в окне **База данных**.

В окне **Создание отчетов**, поочередно активизируя названия таблиц базы данных в списке **Таблицы и запросы**, перенесем нужные поля из окна **Доступные поля** в окно **Выбранные поля**, как показано на рисунке 3.28.

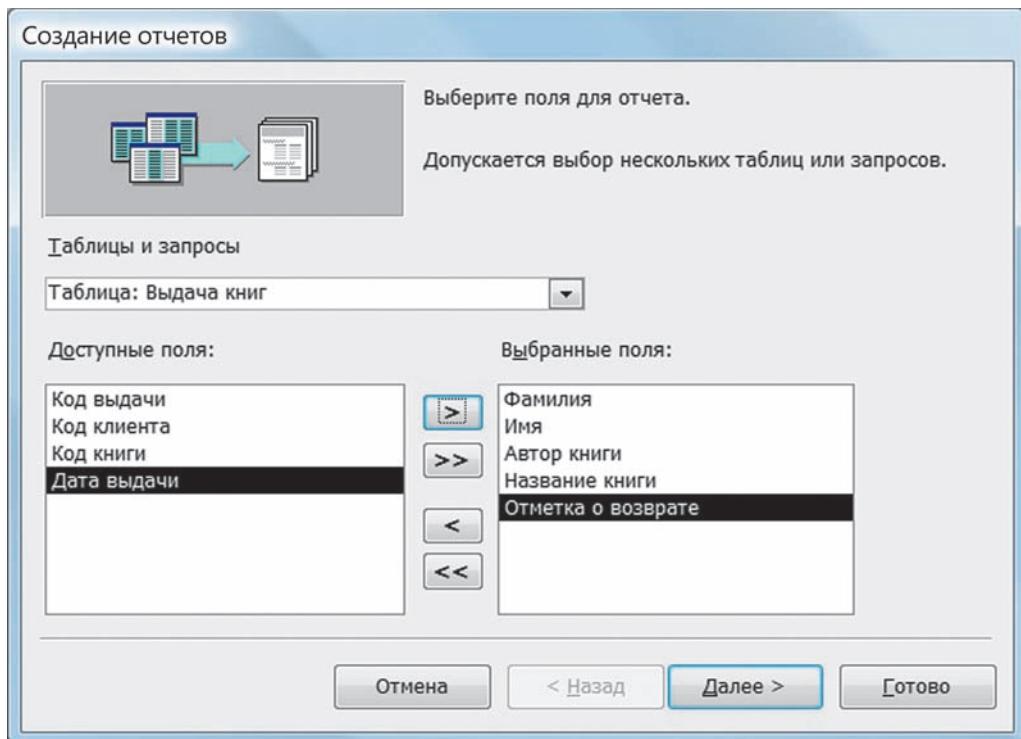


Рис. 3.28

Фамилия	Имя	Автор книги	Название книги	Отметка о возврате
Смирнов	Александр	А. Кристи	Собр. сочинений. Т. 10	<input checked="" type="checkbox"/>
Иванов	Павел	В. Быков	Собр. сочинений. Т. 3	<input checked="" type="checkbox"/>
Титов	Алексей	Я. Мавр	Полесские робинзоны	<input checked="" type="checkbox"/>
Симончик	Валентина	Ж. Верн	Плавучий остров	<input checked="" type="checkbox"/>
Иванов	Павел	Н. Кун	Легенды и мифы	<input type="checkbox"/>
Сидоров	Василий	А. Кристи	Собр. сочинений. Т. 10	<input type="checkbox"/>
Сидоров	Василий	Л. Толстой	Война и мир	<input type="checkbox"/>

Рис. 3.29

Последовательно нажимая кнопку **Далее** в окне **Создание отчетов**, укажем макет размещения данных, например **по левому краю**, и стиль оформления данных, например **Официальная**. В последнем окне нажмем кнопку **Готово**.

В результате мы получим отчет, фрагмент которого представлен на рисунке 3.29.

Созданный отчет может занимать несколько страниц. Его можно расположить в книжной или альбомной ориентации и вывести на печать, как документ текстового редактора Word, с помощью команды **Файл → Печать**.

Подготовка отчета с помощью **Конструктора** осуществляется несколько иначе. Вначале пользователь может создать и сохранить отчет с помощью **Мастера**. Затем в окне **Конструктора** можно изменить расположение элементов в отчете, удалить некоторые элементы, изменить или отредактировать надписи и т. д.



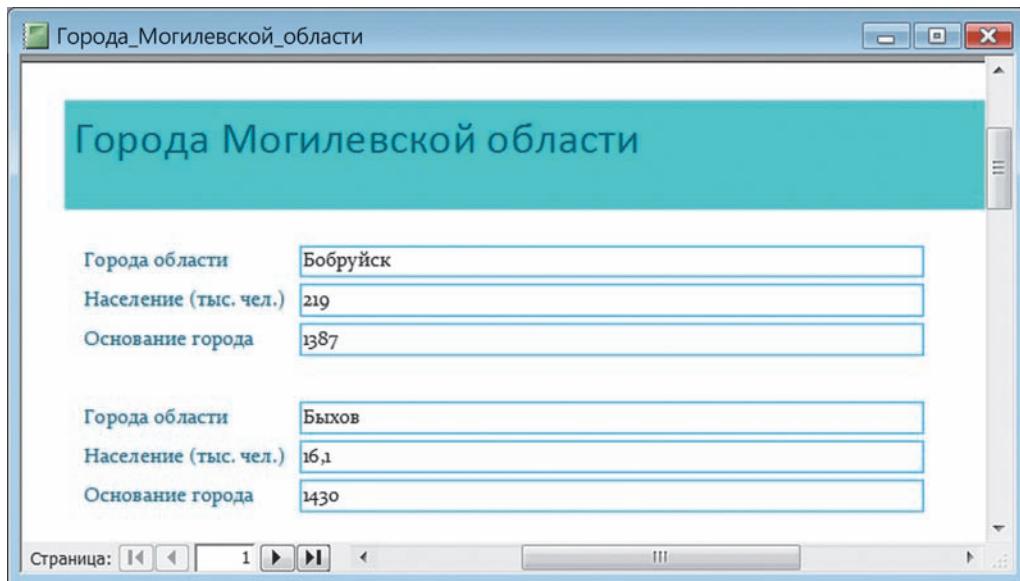
1. Для чего используются отчеты в базах данных?
2. Что называют отчетом в базе данных?

Упражнения

1. Создайте отчет на основе трех таблиц базы данных «Библиотека», как в примере этого параграфа.

При подготовке отчета расположите фамилии клиентов в алфавитном порядке.

2. Откройте таблицу базы данных «Города Могилевской области», предложенную учителем, и создайте отчет с помощью **Мастера**. Вид фрагмента отчета показан на рисунке.



ГЛАВА 4

ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ТЕХНОЛОГИИ

§ 16. Информационные системы

Человек соприкасается с различными по своему назначению системами в быту, на производстве, в образовании, медицине и других сферах деятельности.

Слово «система» означает целое, состоящее из частей.

Системы существуют в живой и неживой природе. Примером сложной системы в неживой природе является Солнечная система, элементы которой — Солнце, планеты и их спутники, множество комет и астероидов. Примером биологической системы являются растительные клетки.

Системы состоят из различных элементов: планет, рек, растений, животных, людей, технических устройств, деталей механизмов, информационных ресурсов, математических уравнений.

Мы часто встречаем такие слова и словосочетания, как биосистема, экологическая система, государственная система, система обучения, система социального обеспечения, система здравоохранения и т. д.

В курсе информатики мы уже знакомились с разными системами: операционными, файловыми, системами управления базами данных.

Добавление к понятию «система» слова «информационная» указывает на ее основное функциональное назначение — работа с информацией.

Информационная система (ИС) — целостная система взаимосвязанных средств и методов сохранения, обработки, поиска и распространения информации, обслуживаемая и используемая человеком.

Существуют различные классификации информационных систем: по функциональному назначению, по области применения, по уровню автоматизации.

По области применения, например, выделяют такие ИС: библиотечные, правовые, экономические, банковские, маркетинговые, медицинские, метеорологические и многие другие.

Среди ИС существует большой класс информационно-поисковых (справочных) систем (ИПС), которые предназначены для поиска информации. Поиск и отбор информации в ИПС осуществляется пользователем по заданным критериям или условиям — запросам. Запросы пользователя в ИПС строятся обычно на естественном для человека языке, например русском, белорусском, английском.

Многие ИПС в настоящее время доступны в сети Интернет. Они оперативно предоставляют пользователю доступ к огромному числу электронных информа-

ционных ресурсов. Обзор информационных ресурсов и информационных систем Беларуси размещен на сайте <http://infores.mpt.gov.by>. С некоторыми из них Вы уже знакомились при изучении информатики.

Вообще, проблемам разработки и эффективного использования национальных информационных ресурсов в нашей стране уделяется большое внимание на государственном уровне. На создание национальной автоматизированной информационной системы, основной задачей которой является формирование единого национального информационного ресурса, нацелена государственная программа «Электронная Беларусь». Для упрощения взаимодействия граждан с автоматизированными информационными системами государственных органов внедряется программный комплекс «Одно окно».

Национальный правовой интернет-портал Республики Беларусь <http://pravo.by> обеспечивает доступ к **Эталонному банку данных правовой информации**. Он представляет собой автоматизированную полнотекстовую информационно-поисковую систему по законодательству Республики Беларусь, позволяющую быстро найти необходимую информацию (рис. 4.1).

Рис. 4.1

Общее количество документов, доступных на этом сайте, превышает 120 тысяч. Тексты правовых актов представлены в их действующей редакции, т. е. с учетом всех внесенных в них изменений и дополнений, вступивших в силу.

Важное экономическое значение имеют **геоинформационные системы** (ГИС), которые обеспечивают сбор, хранение, отображение и распространение географических данных. Например, цифровые карты демонстрируют размещение объектов на местности для выявления отношений между ними, для исследования изменений, произошедших на изучаемой территории за определенный период времени. Основные области применения ГИС: экология и природопользование, землеустройство, региональное планирование, демография и исследование трудовых ресурсов.

Для решения задач в области землеустройства и земельного кадастра предназначена **Земельно-информационная система Республики Беларусь** (<http://landgis.by>).

Одной из наиболее популярных геоинформационных систем является система City Info, которая предоставляет электронную карту Минска или других областных городов и обеспечивает поиск необходимого объекта (рис. 4.2). После ввода в окне **Адрес** названия улицы и номера дома можно увидеть на карте в увели-

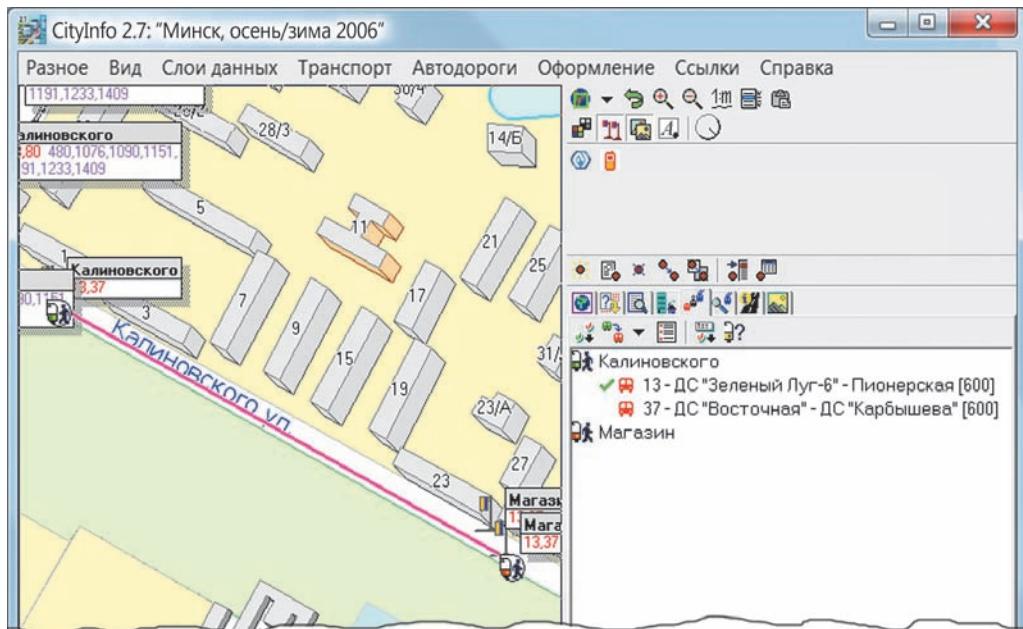


Рис. 4.2

ченном масштабе место расположения найденного системой здания. С помощью вкладки **Транспорт** можно узнать необходимый вид общественного транспорта и номер маршрута для перемещения из одного пункта города в другой. Для этого нужно щелкнуть левой кнопкой мыши по двум пунктам на карте города.

Все большее распространение получают системы спутниковой навигации (например, GPRS), которые на основе заложенных в них цифровых карт позволяют определять местоположение объектов и могут эффективно использоваться, например, для управления дорожным движением.

Решению актуальных экологических проблем помогают **биоинформационные системы**, например **Красная книга Республики Беларусь** (<http://redbook.minpriroda.by>) (рис. 4.3).

Красная книга Республики Беларусь содержит сведения о редких и находящихся под угрозой исчезновения на территории Беларуси видов животных и дикорастущих растений. Ее ресурсы являются общедоступными.



Рис. 4.3

Пользователь имеет возможность выполнить поиск необходимых сведений о животных или растениях по каталогу и по алфавиту.

Информатика совместно с кибернетикой, электроникой, биологией занимается разработкой интеллектуальных информационных систем, создаваемых на базе компьютера, для имитации решения человеком сложных интеллектуальных задач.

Интеллектуальные информационные системы наделены способностью видеть, слышать, говорить, думать и даже чувствовать. В рамках исследований в области создания таких систем разрабатываются естественные интерфейсы компьютера, создаются роботы и робототехнические комплексы.

Примером известной системы искусственного интеллекта является система, имитирующая зрительное восприятие. Система состоит из компьютерных программ и устройств распознавания образов. Она работает в двух режимах: обучения и распознавания. В режиме обучения человек или робот, играющие роль учителя, предъявляют системе различные объекты или предметы и сообщают о них все необходимые сведения. Затем в режиме распознавания человек показывает системе новые объекты, и она должна их правильно распознать или классифицировать.

Экспертные системы (ЭС) — это направление исследований по созданию искусственного интеллекта. Существуют экспертные системы, передающие опыт более подготовленных пользователей менее подготовленным; интеллектуальные обучающие, информационно-логические и робототехнические системы; консультирующие и тренажерные системы; системы поддержки принятия решений.

Основные области применения ЭС: медицина, электроника, вычислительная техника, геология, математика, космос, сельское хозяйство, управление, финансы, юриспруденция и др.

В медицине создано несколько сотен экспертных систем, которые используются для установления связи между нарушениями деятельности организма человека и их возможными причинами. С помощью расширенной базы знаний при получении всех возможных сведений о состоянии здоровья человека такие системы могут ставить диагноз на уровне врача-консультанта.



1. Какую систему называют информационной системой?
2. Приведите примеры информационных систем.

§ 17. Информационные технологии

Определение информационных технологий тесно связано с понятием «технология».

Термин «технология» происходит от греческого слова *techne* — *наука об умении, мастерстве, искусстве* — и имеет множество значений.

В узком смысле под технологией понимают процесс, определяемый совокупностью операций, приемов, с помощью которых функционируют конкретные орудия производства: механизмы, станки, различная аппаратура.

В более широком смысле с помощью технологий описываются многие производственные, экономические, социальные, культурные и другие процессы, происходящие в обществе.

Понятие «информационные технологии» впервые было применено в конце 50-х годов XX в. в Англии и США, однако его активное использование началось в 80-е годы XX в., когда под влиянием новых технологий в обществе начали широко использовать термин «информация».

В широком смысле под **информационными технологиями (ИТ)** понимается комплекс взаимосвязанных научных, технологических, инженерных дисциплин, изучающих методы эффективной организации труда людей, занятых обработкой и хранением информации во всех сферах человеческой деятельности: производственной, управлеченческой, финансовой, научной, социальной, культурной.

В Законе Республики Беларусь от 10 ноября 2008 г. «Об информации, информатизации и защите информации» дано следующее определение информационной технологии: ИТ — совокупность процессов, методов осуществления поиска, получения, передачи, сбора, обработки, накопления, хранения, распространения и (или) предоставления информации, а также пользования информацией и защиты информации.

В связи с активным внедрением вычислительной техники и персональных компьютеров в узком смысле под ИТ обычно подразумевают **компьютерные информационные технологии (КИТ)**.

Под **компьютерной информационной технологией** понимается процесс, позволяющий человеку осуществлять обработку, поиск, сбор, хранение и передачу информации с помощью компьютера.

Если в основу классификации КИТ положить процесс обработки информации, представленной в разной форме, то можно выделить ИТ обработки текстовой и графической информации, ИТ по работе с базами данных, электронными таблицами и др.

Следует понимать, что одна информационная технология может включать в себя другие. Например, мультимедийная технология включает в себя технологии обработки текстовой и графической информации, технологию создания анимаций, веб-технологию.

Информационную технологию, описывающую процесс обработки текстовой информации, можно представить в виде иерархической структуры, выделив *этапы, действия и элементарные операции* этого процесса (рис. 4.4).

Различие между информационными системами и информационными технологиями является очень важным. Следует четко понимать, что в основе информационной технологии лежит, прежде всего, *процесс*, выполнение которого построено на способах деятельности и операциях, а в основе информационной системы лежат используемые программные и технические *средства*.

Например, когда мы говорим о технологии обработки текстовой информации, мы можем не привязывать ее к конкретной компьютерной программе или системе, так как процесс редактирования текста является общим для любых компьютерных программ, в которых обрабатывается текстовая информация.

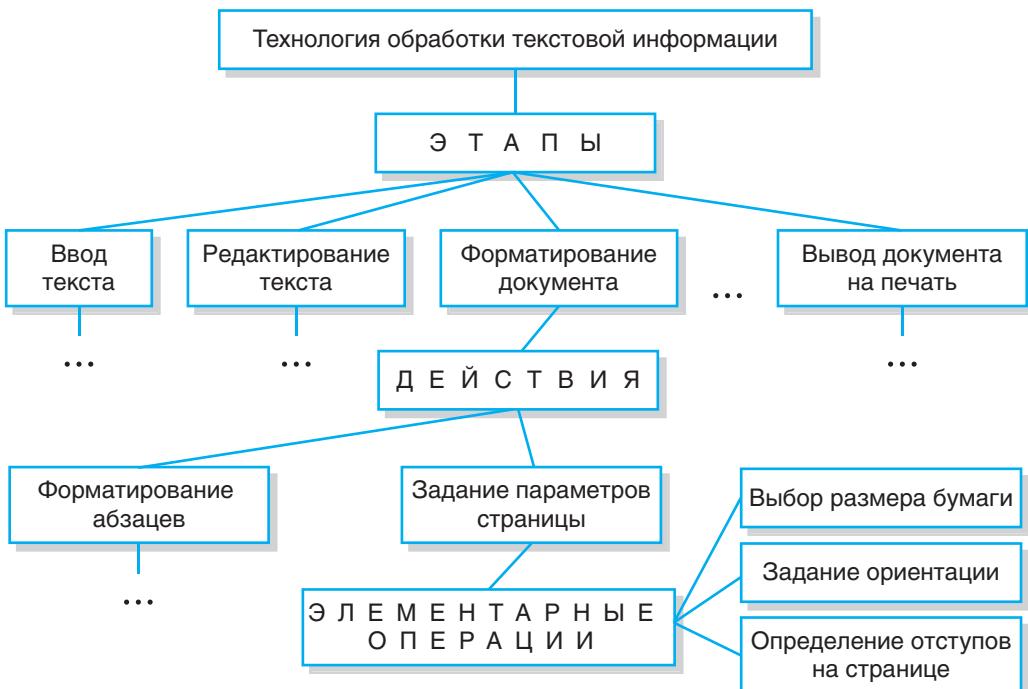


Рис. 4.4

Настоящее время характеризуется активным внедрением информационных систем и технологий в образование.

Практически все изученные Вами компьютерные технологии активно применяются в образовании. Например, на основе мультимедийных технологий создаются учебные компьютерные презентации. Использование аудио и видео облегчает изучение иностранных языков. Технология обработки данных в электронных таблицах помогает не только выполнять сложные расчеты, но и создавать учебные компьютерные модели из разных предметных областей и изучать эти виртуальные модели в курсах физики, химии, биологии и т. д.

Регулирование использования ИС и ИТ в образовании, перспективы и направления их развития находят свое отражение в программах «Комплексной информатизации системы образования Республики Беларусь», которые разрабатываются каждые 4—5 лет.

Широкое применение в белорусских школах получили электронные средства обучения (ЭСО) по различным учебным предметам.

По своему назначению ЭСО можно разделить на обучающие и тестовые программы, учебные информационно-поисковые системы, моделирующие и демонстрирующие программные средства, учебные игровые и досуговые программы и т. д.

Перечень рекомендуемых ЭСО для образовательных учреждений содержится на сайте Главного информационно-аналитического центра Беларуси (адрес в сети Интернет — <http://www.giac.unibel.by>).

Ежегодно в нашей стране проводится Республиканский конкурс «Компьютер. Образование. Интернет» по разработке ЭСО для общеобразовательных, профессиональных и специальных учебных заведений, в котором могут принимать участие учащиеся и педагоги.

Еще одним направлением использования ИС и ИТ является применение их в управлеченческой деятельности в системе образования. Разработанные и применяемые в Беларуси автоматизированные системы управления учреждениями образования содержат базы данных с различной информацией об учителях, учащихся, родителях и др.

С помощью веб-технологий в учреждениях образования создаются веб-сайты. Сайт школы обычно содержит историю и новости школы, сведения об учителях и учащихся, о достижениях коллектива, а также учебно-методические материалы.

В Беларуси разработан и используется банк данных одаренной и талантливой молодежи страны, который включает информацию о лауреатах и стипендиатах специального фонда Президента Республики Беларусь по социальной поддержке одаренных учащихся и студентов, о победителях, призерах и обладателях дипломов международных и республиканских олимпиад и конкурсов и т. д.

С целью оказания помощи учащимся в выборе профессии и учебного заведения для продолжения образования разработаны и используются системы инфор-

The screenshot shows the homepage of the Abiturient.by website. At the top, there's a navigation bar with links for Mathematics, Russian Language, Belarusian, Chemistry, Physics, and a search bar. Below the navigation, there are three main sections: 1) 'Кем быть?' (What to become?), 2) 'Где учиться?' (Where to study?), and 3) 'Как поступить?' (How to apply?). Each section has a brief description and a call-to-action button. On the left, a sidebar titled 'ВАШЕМУ ВНИМАНИЮ' lists various links like 'Главная', 'Новости', 'Каталог вузов', etc. In the center, there's a news section with an image of students studying and a specific news item about professional psychological counseling.

Рис. 4.5

мационно-педагогической поддержки старшеклассников, например Абитуриент.by (www.abiturient.by).

Эта система осуществляет мониторинг вступительной кампании в вузы Республики. Полный перечень специальностей каждого вуза, по которым производится набор абитуриентов на все формы обучения, приведен на вкладке **Где учиться?** (рис. 4.5) в разделе «План приема» соответствующего вуза.

Узнать результаты репетиционного и централизованного тестирования можно на сайте Республиканского института контроля знаний www.rikz.unibel.by. Здесь же размещены сведения о методических пособиях, которые помогут абитуриентам подготовиться к тестированию.

В заключение отметим, что интенсивность использования информационных систем и технологий во всех сферах человеческой деятельности с каждым годом неуклонно возрастает.

- 1. Какую технологию называют компьютерной информационной технологией?
- 2. Приведите примеры компьютерных информационных технологий.
- 3. Какие информационные технологии и системы используются в Вашем учебном заведении?

Структура HTML-документа

<html> </html>	Начало и конец HTML-документа
<head> </head>	Головная часть: служебная информация
<title> </title>	Заголовок окна, в котором отображается документ
<body> </body>	Тело HTML-документа

Фон и цвет страницы

<body background="имф">	Фоновая картинка: имф — имя файла или URL
<body bgcolor="цвет">	Цвет фона: цвет — название или код цвета
<body text="цвет">	Цвет текста: цвет — название или код цвета

Структура страницы

<p>	Абзац
<div>	Раздел (может содержать несколько абзацев)
<p align=left right center>	Горизонтальное выравнивание абзаца или раздела: по левому краю по правому краю по центру страницы
<div align=left right center>	Переход на новую строку
 	Горизонтальная разделительная линия
<hr>	Толщина линии в пикселях
<hr size=h>	Длина линии в пикселях или процентах от ширины окна
<hr width=w>	Горизонтальное выравнивание линии
<hr align=left right center>	

Форматирование текста

	Полужирное начертание
<i>	Курсивное начертание
<u>	Подчеркнутое начертание
<sup>	Верхний индекс
<sub>	Нижний индекс
<h1>... <h6>	Заголовки (6 уровней: 1 — 6)
	Задание шрифта: шрифт — название
	Размер шрифта. Абсолютный: 1 2 3 4 5 6 7 Относительный: -2 -1 0 +1 +2 +3 +4
	Цвет шрифта: цвет — название или код цвета
<pre>	Авторское форматирование. Текст в браузере отображается так, как записан в HTML-коде

Изображения на странице

<pre> </pre>	<p>Ссылка на рисунок: <code>imf</code> — имя файла или URL</p> <p>Ширина <code>w</code> и высота <code>h</code> изображения в пикселях</p> <p>Толщина рамки вокруг изображения в пикселях</p> <p>Вертикальное выравнивание рисунка относительно соседних объектов(текста): поверхнему краю по нижнему краю по центру</p> <p>Горизонтальное выравнивание рисунка: слева от текста справа от текста</p> <p>Размещение подсказки: <code>txt</code> — текст сообщения</p>
--	---

Гиперссылки

<pre> <body link="цвет"> <body alink="цвет"> <body vlink="цвет"></pre>	<p>Ссылка на страницу, которая открывается в том же окне: <code>imf</code> — имя файла или URL</p> <p>Ссылка на страницу, которая открывается в другом окне: <code>imf</code> — имя файла или URL</p> <p>Создание закладки с именем <code>nm</code></p> <p>Ссылка на закладку с именем <code>nm</code> в том же документе</p> <p>Ссылка на закладку с именем <code>nm</code> в другом документе: <code>imf</code> — имя файла или URL</p> <p>Цвет непосещенной ссылки: <code>цвет</code> — название или код</p> <p>Цвет активной ссылки: <code>цвет</code> — название или код</p> <p>Цвет посещенной ссылки: <code>цвет</code> — название или код</p>
--	---

Списки

	Маркированный список
	Элемент списка
<ul type=circle disc square>	Тип маркера: окружность круг квадрат
	Нумерованный список
<ol type=1 a A>	Способ нумерации: арабские цифры строчные латинские буквы прописные латинские буквы
<ol start=n>	Начало нумерации с номера n

Таблицы

<table>	Таблица
<table width=w>	Ширина таблицы в пикселях или процентах от ширины окна
<table align=left right center >	Горизонтальное выравнивание таблицы относительно соседних объектов (окна): по левому краю по правому краю по центру
<table border=w>	Толщина границ в пикселях
<table cellspacing=w>	Расстояние между ячейками в пикселях
<table cellpadding=w>	Расстояние между содержимым ячейки и ее границей в пикселях
<tr>	Строка таблицы
<td>	Ячейка таблицы
<td width=w>	Ширина ячейки в пикселях или в процентах
<td align=left right center >	Горизонтальное выравнивание содержимого: по левой границе по правой границе по центру ячейки
<td valign=top bottom middle>	Вертикальное выравнивание содержимого: по верхней границе по нижней границе по центру
<td bgcolor="цвет">	Цвет ячейки
<td colspan=n>	Объединение n столбцов
<td rowspan=n>	Объединение n строк

Названия и шестнадцатеричные коды цветов

Название	Код	Цвет
aqua	#00FFFF	
black	#000000	
blue	#0000FF	
blueviolet	#8A2BE2	
brown	#A52A2A	
chocolate	#D2691E	
coral	#FF7F50	
crimson	#DC143C	
cyan	#00FFFF	
darkblue	#00008B	
darkcyan	#008B8B	
darkgray	#A9A9A9	
darkgreen	#006400	
darkmagenta	#8B008B	
darkorange	#FF8C00	
darkred	#8B0000	
gold	#FFD700	
gray	#808080	
green	#008000	
indigo	#4B0082	
ivory	#FFFFFF	
khaki	#F0E68C	
lavender	#E6E6FA	
lightblue	#ADD8E6	
lightcyan	#E0FFFF	
lightgreen	#90EE90	
lightgrey	#D3D3D3	
lightpink	#FFB6C1	

Название	Код	Цвет
lime	#00FF00	
magenta	#FF00FF	
maroon	#800000	
mediumblue	#0000CD	
navy	#000080	
olive	#808000	
orange	#FFA500	
orchid	#DA70D6	
palegreen	#98FB98	
peru	#CD853F	
pink	#FFC0CB	
plum	#DDA0DD	
purple	#800080	
red	#FF0000	
rosybrown	#BC8F8F	
salmon	#FA8072	
seagreen	#2E8B57	
sienna	#A0522D	
silver	#C0C0C0	
skyblue	#87CEEB	
tan	#D2B48C	
teal	#008080	
tomato	#FF6347	
violet	#EE82EE	
wheat	#F5DEB3	
white	#FFFFFF	
yellow	#FFFF00	
yellowgreen	#9ACD32	

Графические примитивы модуля GraphABC

<code>SetPixel(x,y,color: integer);</code>	Закрашивает один пиксель с координатами (x,y) цветом <code>color</code>
<code>MoveTo(x,y: integer);</code>	Передвигает невидимое перо к точке с координатами (x,y) ; работает в паре с функцией <code>LineTo(x,y)</code>
<code>LineTo(x,y: integer);</code>	Рисует отрезок от текущего положения пера до точки (x,y) ; координаты пера при этом становятся равными (x,y)
<code>Line(x1,y1,x2,y2: integer);</code>	Рисует отрезок с началом в точке $(x1,y1)$ и концом в точке $(x2,y2)$
<code>Circle(x,y,r: integer);</code>	Рисует окружность с центром в точке (x,y) и радиусом <code>r</code>
<code>Ellipse(x1,y1,x2,y2: integer);</code>	Рисует эллипс, заданный описанным около него прямоугольником с координатами противоположных вершин $(x1,y1)$ и $(x2,y2)$
<code>Rectangle(x1,y1, x2,y2: integer);</code>	Рисует прямоугольник, заданный координатами противоположных вершин $(x1,y1)$ и $(x2,y2)$
<code>Arc(x,y,r,a1,a2: integer);</code>	Рисует дугу окружности с центром в точке (x,y) и радиусом <code>r</code> , заключенную между двумя лучами, образующими углы <code>a1</code> и <code>a2</code> с осью ОХ (<code>a1</code> и <code>a2</code> — вещественные, задаются в градусах и отчитываются против часовой стрелки)
<code>Pie(x,y,r,a1,a2: integer);</code>	Рисует сектор круга, ограниченный дугой (параметры процедуры имеют тот же смысл, что в процедуре <code>Arc</code>)
<code>FloodFill(x,y,color: integer);</code>	Заливает область цветом <code>color</code> , начиная с точки (x,y)
<code>TextOut(x,y: integer; s: string);</code>	Выводит строку <code>s</code> в позицию (x,y) (точка (x,y) задает верхний левый угол прямоугольника, который будет содержать текст из строки <code>s</code>)

Действия с графическим окном

<code>SetWindowSize(w,h: integer);</code>	Устанавливает ширину и высоту графического окна
<code>SetWindowCaption(s: string);</code>	Устанавливает заголовок графического окна
<code>SaveWindow(fname: string);</code>	Сохраняет содержимое графического окна в файл с именем <code>fname</code>
<code>LoadWindow(fname: string);</code>	Выводит в графическое окно рисунок из файла с именем <code>fname</code>
<code>ClearWindow;</code>	Очищает графическое окно белым цветом
<code>ClearWindow(c: ColorType);</code>	Очищает графическое окно цветом <code>c</code>
<code>Redraw;</code>	Осуществляет перерисовку окна

Константы стандартных цветов

clBlack	черный	clBrown	коричневый
clPurple	фиолетовый	clBlue	синий
clWhite	белый	clSkyBlue	голубой
clMaroon	темно-красный	clYellow	желтый
clRed	красный	clGray	серый
clGreen	зеленый	clSilver	серебряный

Действия с пером и кистью

SetPenColor(color: integer);	Устанавливает цвет пера, задаваемый параметром color
SetPenWidth(w: integer);	Устанавливает ширину пера, равную w пикселям
SetPenStyle(ps: integer);	Устанавливает стиль пера, задаваемый параметром ps. Стили пера задаются именованными константами: psSolid, psClear, psDash, psDot, psDashDot, psDashDotDot
SetBrushColor(color: integer);	Устанавливает цвет кисти, задаваемый параметром color
SetBrushPicture(fname: string);	Устанавливает в качестве образца для закраски кистью изображение, хранящееся в файле fname
SetBrushStyle(bs: integer);	Устанавливает стиль кисти, задаваемый параметром bs. Стили кисти задаются именованными константами: bsSolid, bsHorizontal, bsBDiagonal, bsCross, bsClear, bsDiagCross, bsVertical, bsFDiagonal

Действия с рисунками

LoadPicture(fname: string): integer;	Загружает рисунок из файла с именем fname в оперативную память и возвращает описатель рисунка в целую переменную n (n:=LoadPicture(fname))
DrawPicture(n,x,y: integer);	Выводит рисунок с описателем n в позицию (x,y) графического окна
DrawPicture(n,x,y,w,h: integer);	Выводит рисунок с описателем n в позицию (x,y) графического окна, масштабируя его размеры к ширине w и высоте h
SavePicture(n: integer; fname: string);	Сохраняет рисунок с описателем n в файл с именем fname (форматы BMP, JPG или GIF)
SetPictureSize(n,w,h: integer);	Устанавливает размер рисунка с описателем n, равным w × h пикселей
SetPictureTransparent (n: integer; b: boolean);	Устанавливает (b=True) или отключает (b=False) режим прозрачности при рисовании изображения с описателем n

СОДЕРЖАНИЕ

От авторов	3
Глава 1. Основы веб-конструирования	
§ 1. Представление о веб-конструировании	4
1.1. Инструменты и методы разработки веб-сайтов	—
1.2. Проектирование сайта	6
§ 2. Использование офисных приложений для создания веб-страниц	8
2.1. Создание веб-страниц в редакторе MS Word	—
2.2. Сохранение презентаций PowerPoint в виде веб-страниц	14
§ 3. Основы языка разметки HTML	18
3.1. Создание HTML-документа в редакторе Блокнот	—
3.2. Изображения на веб-страницах	24
3.3. Создание гиперссылок	30
§ 4. Подготовка изображений для Интернета	35
§ 5. Веб-конструирование в редакторе FrontPage	41
5.1. Основные элементы интерфейса	—
5.2. Работа в редакторе FrontPage	43
5.3. Использование таблиц	50
5.4. Разработка веб-сайта	54
5.5. Публикация сайта	61
§ 6. Создание фрагментов сайтов по различным предметным областям	62
Глава 2. Основы алгоритмизации и программирования	
§ 7. Выполнение практических заданий из различных предметных областей	65
7.1. Построение геометрических фигур	—
7.2. Использование растровых изображений	70
7.3. Построение графиков и диаграмм	79
§ 8. Выполнение практических заданий по темам учебных предметов	85
8.1. Астрономия	—
8.2. География	90
8.3. Биология и экология	94
8.4. Физика	98
Глава 3. Обработка информации в системе управления базами данных	
§ 9. Базы данных и системы управления базами данных	104
§ 10. Создание таблицы базы данных	108
10.1. Проектирование базы данных	—
10.2. Создание структуры таблицы	111
§ 11. Связывание таблиц базы данных	117
§ 12. Создание и заполнение формы	121
§ 13. Поиск данных с помощью запросов	125
§ 14. Сортировка записей в таблице	129
§ 15. Создание отчетов	131
Глава 4. Информационные системы и технологии	
§ 16. Информационные системы	135
§ 17. Информационные технологии	140
Приложения	144

Учебное издание

**Заборовский Георгий Александрович
Пупцев Александр Евгеньевич**

ИНФОРМАТИКА

Учебное пособие для 11 класса
общеобразовательных учреждений
с русским языком обучения

Зав. редакцией *В. Г. Бехтина*. Редактор *Н. М. Алганова*. Художественный редактор *Л. А. Дашикевич*. Технический редактор *М. И. Чепловодская*. Компьютерная верстка *Г. А. Дудко*. Корректоры *З. Н. Гришели, Т. Н. Ведерникова, Д. Р. Лосик, В. С. Бабеня, А. В. Алешико*.

Подписано в печать 05.07.2010. Формат 70×90¹/₁₆. Бумага офсетная. Гарнитура литературная. Офсетная печать. Усл. печ. л. 11,12. Уч.-изд. л. 9,1. Тираж 113 800 экз. Заказ .

Издательское республиканское унитарное предприятие «Народная асвета»

Министерства информации Республики Беларусь.

ЛИ № 02330/0494083 от 03.02.2009.

Пр. Победителей, 11, 220004, Минск.

Республиканское унитарное предприятие

«Минская фабрика цветной печати»

ЛП № 02330/0494156 от 03.04.2009.

Ул. Корженевского, 20, 220024, Минск.

(Название и номер школы)

Учебный год	Имя и фамилия ученика	Состояние учебного пособия при получении	Оценка ученику за пользование учебным пособием
20 /			
20 /			
20 /			
20 /			
20 /			