

Г. А. Заборовский А. И. Лапо А. Е. Пупцев

# ИНФОРМАТИКА

Учебное пособие для 9 класса  
общеобразовательных учреждений  
с русским языком обучения

*Допущено*  
*Министерством образования*  
*Республики Беларусь*

Минск  
«Народная асвета»  
2009

Правообладатель Народная асвета

УДК 004(075.3=161.1)  
ББК 32.81я721  
3-41

Авторы:

Г. А. Заборовский (главы 1, 2, 3, 4), А. И. Лапо (глава 2),  
А. Е. Пупцов (главы 1, 4)

Рецензенты:

кафедра информационных технологий в культуре Белорусского государственного университета культуры и искусств (зав. каф., канд. физ.-мат. наук, доцент *П. В. Гляков*); учитель информатики высшей категории СШ № 209 г. Минска *Л. И. Мосткова*

**Заборовский, Г. А.**

**3-41 Информатика : учеб. пособие для 9-го кл. общеобразоват. учреждений с рус. яз. обучения / Г. А. Заборовский, А. И. Лапо, А. Е. Пупцов. — Минск : Нар. асвета, 2009. — 191 с. : ил.**

ISBN 978-985-03-1224-2.

УДК 004(075.3=161.1)  
ББК 32.81я721

ISBN 978-985-03-1224-2

© Заборовский Г. А., Лапо А. И.,  
Пупцов А. Е., 2009

© Оформление. УП «Народная асвета», 2009

Правообладатель Народная асвета

## ОТ АВТОРОВ

*Уважаемые школьники!*

Информатика в современном мире является одной из самых быстро развивающихся наук. Каждый год совершенствуются информационные технологии, появляется новое техническое, программное и информационное обеспечение компьютера. Настоящее учебное пособие поможет Вам продолжить изучение информатики, информационных технологий и программирования.

Первая глава учебного пособия позволит Вам узнать, каким образом хранится информация в компьютере и как измеряется объем информации.

Вторая глава углубит Ваши знания в области программирования. Вы познакомитесь с новым типом данных — массивом — и научитесь решать задачи с использованием этого типа данных.

В третьей главе Вы узнаете об основах анимации, познакомитесь с видами анимации, научитесь создавать анимационные фильмы.

В четвертой главе Вам предстоит ознакомиться с информационными ресурсами глобальной компьютерной сети Интернет. Вы узнаете о службе WWW и о способах поиска информации.

Для закрепления изученного материала после параграфов помещены вопросы ?, для практической работы даны упражнения.

Желаем успехов!

# Глава 1

## ПРЕДСТАВЛЕНИЕ ИНФОРМАЦИИ В КОМПЬЮТЕРЕ

### § 1. Кодирование информации

Для общения друг с другом мы используем естественный язык, например белорусский или русский. В основе естественного языка лежит алфавит, представляющий собой систему графических знаков для передачи звуков устной речи. Алфавит естественного языка является универсальным кодом любой письменной культуры. При разговоре этот код передается звуками, при письме — буквами.

Кроме естественных языков, человек часто использует формальные языки со своими особыми кодами: математические и химические формулы, ноты.

Код — это система условных знаков, каждому из которых ставится в соответствие определенное значение.

Процесс преобразования информации в соответствии с правилами, заданными некоторым кодом, называют **кодированием**. Процесс, обратный кодированию, называют **декодированием**.

Кодировать и передавать информацию можно различными способами: устно, письменно, с помощью жестов, электрических сигналов и т. п.

Например, широко применяется кодирование товаров с помощью штрих-кода (рис. 1.1). На магнитной полосе банковской пластиковой карточки закодирована информация о ее владельце (рис. 1.2).



Рис. 1.1

- 1 — код страны
- 2 — код изготовителя
- 3 — код товара
- 4 — контрольная цифра

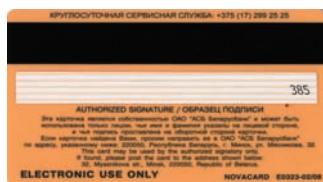


Рис. 1.2

Во время радиосвязи применяют азбуку Морзе. В ней буквы заменяются последовательностями из коротких и длинных сигналов — точек и тире.

<b>А</b>	<b>Б</b>	<b>В</b>	<b>Г</b>	<b>Д</b>	<b>Е</b>	<b>Ж</b>	<b>З</b>	<b>И</b>	<b>К</b>
.-	-...-	-.-.	-.-.	-..	.	....-	-.-..	..	-.-
<b>Л</b>	<b>М</b>	<b>Н</b>	<b>О</b>	<b>П</b>	<b>Р</b>	<b>С</b>	<b>Т</b>	<b>У</b>	<b>Ф</b>
-..	--	-.	-.-	-.-.	-.	...	-	..-	-.-
<b>Х</b>	<b>Ц</b>	<b>Ч</b>	<b>Ш</b>	<b>Щ</b>	<b>ТЬ,Ь</b>	<b>Ы</b>	<b>Э</b>	<b>Ю</b>	<b>Я</b>
....	-.-.	-.-.	----	-.-.	-.-.	-.-.	-.-..	..--	-.-.
<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>0</b>
-....-	...-....	...-....	....-	.....	-....	-.-..	-....	-....	-----

Компьютер может обрабатывать различную информацию (числовую, текстовую, графическую, звуковую) только в цифровой форме.

Цифровая форма представления информации в компьютере реализована в виде двоичного кода, алфавит которого состоит из двух цифр: 0 и 1. Это связано с тем, что наиболее просты технические устройства, обладающие двумя устойчивыми состояниями: включено/выключено, соединено/разъединено, есть ток/нет тока и т. п. Такие состояния могут быть выражены логически: истина (единица), ложь (нуль).

- ? 1. Что называется кодом?  
2. Приведите примеры кодирования информации.  
3. Почему в компьютере используется двоичное кодирование информации?

### *Упражнение*

Закодируйте с помощью азбуки Морзе:

- а) слово **КОД**;
- б) слово **АЗБУКА**;
- в) сигнал бедствия **СОС**.

## **§ 2. Единицы измерения объема информации**

В своей деятельности человек постоянно использует различные единицы измерения. Например, время измеряется в секундах, минутах, часах, расстояние — в метрах, километрах, температура — в градусах и т. д.

Для измерения количества информации тоже существуют свои единицы. Минимальную единицу количества информации называют **битом** (bit).

Слово «бит» произошло от двух английских слов *binary* — двоичный, *digit* — цифра. Бит может принимать только одно из двух значений: 0 или 1.

Для кодирования минимального количества информации достаточно одного двоичного разряда, поэтому в вычислительной технике слово «бит» часто применяют в значении «двоичный разряд».

Устройства компьютера работают с группами битов. Так, первые процессоры за один такт обрабатывали 8 бит или 16 бит. Современные процессоры имеют разрядность 32 бит или 64 бит.

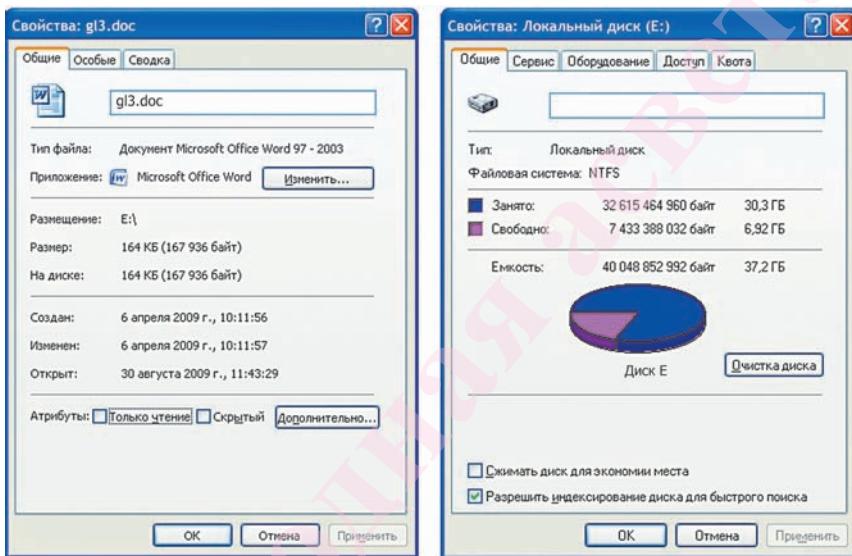
В качестве более крупной единицы измерения информации принят **байт**.

**Байт** — единица измерения количества информации, состоящая из восьми последовательных и взаимосвязанных битов: **1 байт =  $2^3$  бит = 8 бит**.

Для обозначения большого объема информации используются производные единицы измерения, значения которых связаны со степенью числа 2:

- 1 Кбайт (килобайт) = 1024 байт =  $2^{10}$  байт;  
 1 Мбайт (мегабайт) = 1024 Кбайт =  $2^{20}$  байт;  
 1 Гбайт (гигабайт) = 1024 Мбайт =  $2^{30}$  байт;  
 1 Тбайт (терабайт) = 1024 Гбайт =  $2^{40}$  байт.

В этих единицах измеряются объем памяти компьютера, размеры файлов.



а

б

Рис. 1.3

Для определения размера файла, папки или диска при работе в операционной системе Windows необходимо щелкнуть правой кнопкой мыши на требуемом объекте и в открывшемся контекстном меню выбрать пункт **Свойства**. Например, размер файла *gl3.doc* на рисунке 1.3, *а* равен 164 Кбайт, а емкость диска Е: на рисунке 1.3, *б* — 37,2 Гбайт (30,3 Гбайт занято, 6,92 Гбайт свободно).

- ? 1. В каких единицах измеряют количество информации?  
 2. Сколько байтов содержит 1 Кбайт?

3. Сколько битов занимают:
  - а) 20 байт;
  - б) 1000 байт;
  - в) 1 Кбайт?
4. Определите, какой объем информации больше:
  - а) 7 бит или 1 байт;
  - б) 1 Кбайт или 1000 байт;
  - в) 1025 Мбайт или 1 Гбайт.

### **Упражнения**

1. Определите, сколько всего гигабайтов памяти выделено на винчестере Вашего компьютера под диск С:. Сколько гигабайтов свободно?
2. Определите размер папки **Мои документы** Вашего компьютера.

## **§ 3. Понятие системы счисления. Двоичная система счисления**

С древних времен в практической деятельности человека часто возникала потребность счета и измерения. Результаты счета предметов выражались вначале весьма примитивно: зарубки на палочках, узелки на веревках и др. С развитием письменности человек начал отображать с помощью знаков (записывать) информацию о количестве предметов на подручных материалах: глиняных табличках, папирусе, бересте и др. Таким образом, для обозначения чисел стали использоваться знаки.

Способ записи чисел с помощью письменных знаков называют **системой счисления**.

Одной из наиболее древних являлась **египетская иероглифическая** система счисления. В ней числа представлялись в виде отдельных знаков, например:

— один;    — десять;    — сто.

Так, число  означало:

$$100 + 10 + 10 + 1 + 1 + 1 = 123.$$

Существовали системы счисления, в которых для записи чисел использовались буквы алфавита, например *старославянская* система счисления.

Старославянская система счисления										
а	в	г	д	ѓ	з	ѣ	и	и	к	
1	2	3	4	5	6	7	8	10	20	
Десятичная система счисления										

Десятичная система счисления зародилась в Индии приблизительно в V в., затем она появилась в арабских рукописях. Из арабских рукописей эта система пришла в Европу в IX—XII вв. Поэтому современную десятичную систему счисления называют *арабской*.

Системы счисления могут быть *непозиционными* и *позиционными*.

**Непозиционными** называют такие системы счисления, в которых каждый знак (цифра) в записи любого числа имеет одно и то же значение и не зависит от своего расположения в числе.

В непозиционной *римской* системе счисления для обозначения чисел используются следующие знаки:

Римская система счисления											
I	II	III	IV	V	VI	VII	VIII	IX	X	и т. д.	
1	2	3	4	5	6	7	8	9	10		
Десятичная система счисления											

Например, число XXVIII, записанное в римской системе счисления, в десятичной системе счисления означает:  $10 + 10 + 5 + 1 + 1 + 1 = 28$ .

Древнеегипетская и старославянская системы также являются непозиционными.

**Позиционными** называют такие системы счисления, в которых значение каждого знака (цифры) в записи любого числа зависит от расположения (позиции) этого знака в числе. Количество цифр, используемых для записи чисел в позиционной системе счисления, называется ее **основанием**.

Мы используем **позиционную десятичную систему счисления**. Основанием этой системы является число 10. Предполагают, что основание 10 связано с количеством пальцев рук у человека.

Для записи любого числа в десятичной системе счисления используют десять цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Комбинируя эти цифры, можно записывать любые числа.

Например, цифры числа 737 в десятичной системе счисления являются коэффициентами его записи в виде суммы степеней числа 10:

$$737 = 7 \cdot 10^2 + 3 \cdot 10^1 + 7 \cdot 10^0 = 7 \cdot 100 + 3 \cdot 10 + 7 \cdot 1.$$

Из этого примера видно, что цифра 7 в зависимости от своей позиции в этом числе означает и 7 сотен, и 7 единиц, а цифра 3 означает три десятка.

Для кодирования информации в компьютере вместо привычной десятичной системы счисления используется **двоичная система счисления**.

Двоичной системой счисления люди начали пользоваться очень давно. Древние племена Австралии и островов Полинезии использовали эту систему в быту. Так, полинезийцы передавали необходимую

информацию, выполняя два вида ударов по барабану: звонкий и глухой. Это было примитивное представление двоичной системы счисления.

**Двоичная система счисления** — это позиционная система счисления с основанием 2. Для записи чисел в ней используются только две цифры: 0 и 1.

Для обозначения системы счисления, в которой представляется число, используют нижний индекс, указывающий основание системы. Например,  $11011_2$  — число в двоичной системе счисления.

Цифры в двоичном числе являются коэффициентами его представления в виде суммы степеней с основанием 2, например:

$$1011_2 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0.$$

В десятичной системе счисления это число будет выглядеть так:

$$1011_2 = 8 + 0 + 2 + 1 = 11_{10}.$$

Определим максимальное число, которое может разместиться в одном байте. Представим байт (последовательность из 8 бит) в виде двоичного числа:

бит	бит	бит	бит	бит	бит	бит	бит
1	1	1	1	1	1	1	1
байт							

Таким образом, максимальное двоичное число, которое может разместиться в одном байте, равно  $11111111_2$ . Представим его в десятичной системе счисления:

$$\begin{aligned} 11111111_2 &= 1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + 1 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + \\ &\quad + 1 \cdot 2^1 + 1 \cdot 2^0 = \\ &= 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255_{10}. \end{aligned}$$

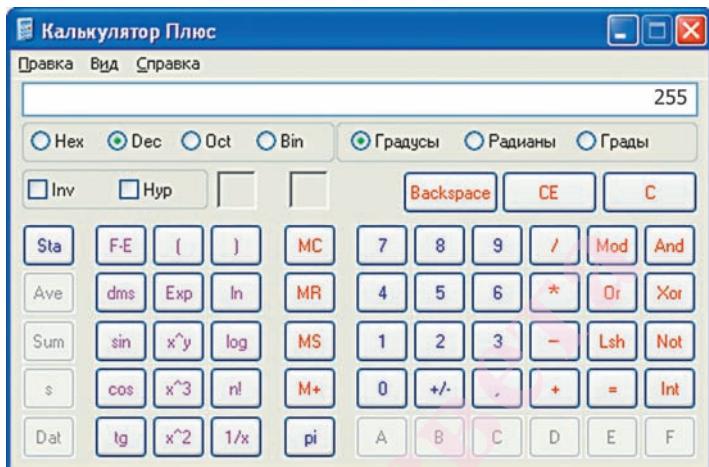


Рис. 1.4

Специалисты-профессионалы, работающие с компьютером, часто используют **восьмеричную и шестнадцатеричную** системы счисления. Это позволяет сократить запись двоичного числа.

При необходимости перевода чисел из одной системы счисления в другую может быть использован калькулятор, имеющий возможность выполнять такие операции. Так, выбрав в меню калькулятора, входящего в состав стандартных приложений операционной системы Windows, **Вид → Инженерный**, можно работать с двоичными, десятичными, шестнадцатеричными числами.

Введем, например, на калькуляторе число 255 в десятичной системе счисления при установленном переключателе **Dec**. Этот переключатель используется для представления числа в десятичной системе счисления (рис. 1.4).

Установим на калькуляторе переключатель **Bin**. Число будет представлено в двоичной системе счисления:  $11111111_2$  (рис. 1.5).

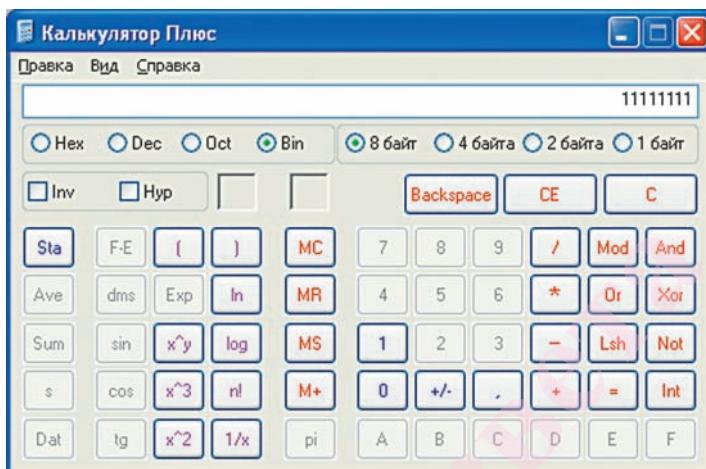


Рис. 1.5

- ? 1. Что называют системой счисления?  
 2. Приведите примеры позиционной и непозиционной систем счисления.  
 3. Какие цифры используют для представления числа в:  
   а) десятичной системе счисления;  
   б) двоичной системе счисления?

### Упражнения

- Представьте целое двоичное число в десятичной системе счисления:
  - $10_2$ ;
  - $111_2$ ;
  - $1111_2$ ;
  - $101010_2$ .
- Прочитайте стихотворение. Какую систему счисления использовал автор этого стихотворения?

Ей было тысяча сто лет.  
 Она в сто первый класс ходила.  
 В портфеле по сто книг носила.  
 Все это правда, а не бред.  
 Когда, пыля десятком ног,  
 Она шагала по дороге,  
 За ней всегда бежал щенок  
 С одним хвостом, зато стоногий.

Она ловила каждый звук  
Своими десятью ушами,  
И десять загорелых рук  
Портфель и поводок держали.  
И десять темно-синих глаз  
Оглядывали мир привычно.  
Но станет все совсем обычным,  
Когда поймете наш рассказ.

## § 4. Представление различных видов информации

### 4.1. Кодирование текстовой информации

Для кодирования текста каждому символу алфавита ставят в соответствие некоторый двоичный код.

Если для кодирования каждого символа использовать 8 бит, или 1 байт, то всего можно закодировать 256 символов.

Совокупность всех символов и соответствующих им кодов обычно представляют в виде таблицы. Такую таблицу называют **кодовой таблицей символов** или **таблицей кодировки**. Нередко для удобства использования вместо двоичного кода в ней записывают его десятичный аналог (Приложение 1).

Отсутствие стандартных таблиц кодировок создавало пользователям большие неудобства, поскольку набранный на одном компьютере текст без перекодирования мог не читаться на другом. Международным стандартом стала таблица ASCII.

Таблица ASCII-кодов состоит из двух частей.

Первая половина таблицы, которую называют стандартной, содержит латинские буквы, цифры и некоторые символы. Они имеют коды от 00000000 до 01111111 (в десятичном представлении — от 0 до 127).

Вторая половина таблицы с кодами от 10000000 до 11111111 (в десятичном представлении — от 128 до

255) используется для кодирования национальных алфавитов. Эта часть имеет несколько вариантов. Для кодирования кириллицы в среде Windows наибольшее распространение получила кодировка **СР 1251**.

### Пример 1. Закодировать слово КОД.

Для кодирования трех букв требуется три байта. По таблице (см. Приложение 1) определим десятичные коды букв. Получим: 138 142 132.

В настоящее время разработана универсальная кодировка Юникод (Unicode), содержащая символы многих национальных алфавитов. Для кодирования символа в ней используется 16 бит (2 байт). Всего можно закодировать 65 536 символов.

Для просмотра кодовых таблиц на Вашем компьютере необходимо выполнить команды: Пуск → → Программы → Стандартные → Служебные → Таблицы символов. Выбрав в таблице символ, можно получить его код (рис. 1.6).

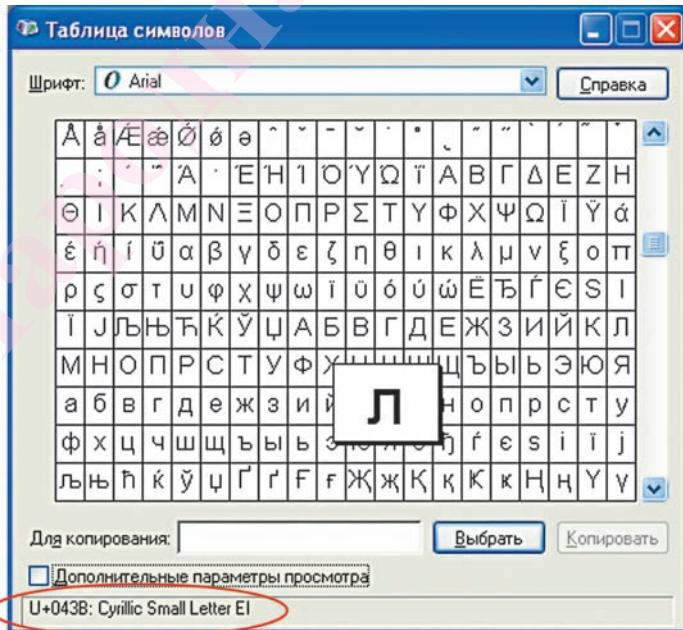


Рис. 1.6

## 4.2. Кодирование графической информации

Изображение на экране монитора формируется из точек (пикселей).

В простейшем случае (черно-белое изображение) каждая точка может иметь лишь два состояния: черный или белый цвет. Для кодирования этой информации требуется 1 бит. Изображение размером  $8 \times 8$  точек занимает 64 бит, или 8 байт.

Любой цвет можно получить смешением в определенных количествах трех основных: красного (R — red), зеленого (G — green) и синего (B — blue). Такая цветовая модель носит название RGB. Чем больше оттенков цвета в изображении, тем больше битов понадобится для кодирования.

Количество битов, используемых для кодирования цвета одного пикселя, называют **глубиной цвета**. Так, при глубине цвета 24 бит изображение размером  $8 \times 8$  точек будет занимать 1536 бит, или 192 байт. При этом на каждую из трех составляющих цвета приходится 8 бит. Возможное количество оттенков равно

$$2^{24} = 2^8 \cdot 2^8 \cdot 2^8 = 256 \cdot 256 \cdot 256 = 16\,777\,216.$$

Для того чтобы на экране монитора формировалось изображение, информация о каждой его точке (цвет точки) должна храниться в видеопамяти компьютера.

**Пример 2.** Рассчитать информационный объем изображения размером  $800 \times 600$  точек при глубине цвета 24 бит.

Для расчета воспользуемся стандартным калькулятором.

Определим количество точек в изображении:

$$800 \cdot 600 = 480\,000.$$

Для кодирования цвета каждой точки требуется  $24 \text{ бит} = 3 \text{ байт}$ . Информационный объем изображения равен:  $3 \cdot 480\,000 = 1\,440\,000 \text{ байт} = 1406 \text{ Кбайт}$ .

В таблице приведен информационный объем (в килобайтах) изображения, занимающего весь экран монитора, в зависимости от параметров мониторов.

Размер изображения	Глубина цвета, бит	
	16	24
	Количество цветов	
	65 536	16 777 216
$1024 \times 768$	1536	2304
$1280 \times 1024$	2560	3840

Информация обо всех точках изображения может быть сохранена на диске в формате **.bmp** (bit map picture — битовая карта изображения). В этом формате по умолчанию сохраняются изображения в редакторе **Paint**. Размер файла даже для простой картинки исключительно велик (см. пример 2).

Для уменьшения объема файла с изображением его битовую карту необходимо подвергнуть дальнейшему кодированию (сжатию).

Можно, например, ограничить палитру цветов, уменьшив их количество. Так, графический формат **.gif** позволяет сохранять информацию не более чем о 256 цветах.

В широко распространенном формате **.jpeg** (**.jpg**) информация о каждом отдельном пикселе заменяется информацией о группах пикселей. Это позволяет уменьшить размер файла (в среднем в 5—6 раз без существенной потери качества изображения).

Следует заметить, что такое кодирование является необратимым. Декодировать сжатый файл и получить изображение исходного качества нельзя.

### 4.3. Кодирование звуковой информации

Компьютер, имеющий звуковую плату, микрофон и акустическую систему, позволяет кодировать (оцифровывать), сохранять и воспроизводить звуковую информацию. С помощью специальных программных средств (звуковых редакторов) звуковые файлы можно редактировать: добавлять голоса или музыкальные инструменты, а также разнообразные эффекты. Компьютер превращается в звуковую студию. Сегодня начинают использоваться программы распознавания речи, появляется возможность управления компьютером при помощи голоса.

Звук — это волна с изменяющейся амплитудой и частотой в диапазоне от 20 Гц до 20 кГц. Чем больше амплитуда, тем громче звук, чем больше частота, тем выше тон.

Микрофон превращает звуковую волну в электрический сигнал, а звуковая плата кодирует его, превращая в последовательность нулей и единиц. Точность преобразования определяется **разрешающей способностью** преобразователя (8 бит — 256 уровней, 16 бит — 65 536 уровней, 24 бит — 16 777 216 уровней) и числом преобразований (выборок) за 1 с — **частотой дискретизации**. Так, для оцифровки сигнала, изображенного на рисунке 1.7, используется 8 уровней (3 бит).

При частоте 8 кГц качество оцифрованного звука соответствует радиотрансляции, а при частоте 44,1 кГц — звучанию аудио-CD. Студийное качество достигается при 96 или 192 кГц.



Рис. 1.7

**Пример 3.** Рассчитать объем звукового фрагмента длительностью звучания 2 с при частоте дискретизации 20 кГц и разрешении 16 бит.

Воспользуемся стандартным калькулятором.

Разрешение умножим на число выборок за 1 с и на время:

$$\begin{aligned} 16 \cdot 20\,000 \cdot 2 &= 640\,000 \text{ бит} = \\ &= 80\,000 \text{ байт} = 78 \text{ Кбайт.} \end{aligned}$$

Закодированный таким образом звуковой фрагмент может быть сохранен в формате .wav.

В таблице приведены размеры звуковых файлов длительностью звучания 1 с (в килобайтах) при различных разрешениях звуковой карты и частотах дискретизации. Для стереозвука размер файла удваивается.

Частота дискретизации, кГц	Разрешение		
	8 бит	16 бит	24 бит
	256	65 536	16 777 216
8	7,813	15,625	23,438
24	23,438	46,875	70,313
44,1	43,066	86,133	129,199
48	46,875	93,750	140,625
96	93,750	187,500	281,250

В формате **.wav** сохраняются звуки с помощью стандартного приложения **Звукозапись**. Размер такого файла исключительно велик (более 20 Мбайт для стереомузыки CD-качества длительностью 1 мин). Для уменьшения объема звукового файла его необходимо подвергнуть дальнейшему кодированию (сжатию). Наиболее популярным сжатым форматом в настоящее время является **.mp3**.

- ? 1. Сколько битов необходимо для кодирования символа с помощью таблицы ASCII?
2. От каких параметров зависит качество двоичного кодирования изображения?
3. От каких параметров зависит качество двоичного кодирования звука?

### **Упражнения**

1. Используя кодовую таблицу ASCII, закодируйте слово:

а) Bit;      б) Байт;      в) ДОМ;      г) Текст.

2. Используя кодовую таблицу ASCII, декодируйте текст: 172 174 164 165 168.

3. Рассчитайте информационный объем изображения размером  $10 \times 10$  точек при глубине цвета 8 бит.

4. Рассчитайте объем монофонического аудиофайла длительностью 10 с при 16-битном кодировании и частоте дискретизации 44,1 кГц.

## Глава 2

# ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

### § 5. Структурированный тип данных: массив

При решении многих задач приходится обрабатывать большое количество однотипных данных, например при расчетах узлов машин и конструкций зданий, при поиске справочной информации и составлении прогноза погоды, наконец, при накоплении результатов и подведении итогов тестирования. Для хранения этих данных пришлось бы вводить большое количество переменных, что привело бы к громоздкости программ их обработки.

Эта проблема решается путем введения специального типа данных, который называется **массивом**.

**Массив** — это обозначаемая одним именем последовательность однотипных элементов.

Место каждого элемента в этой последовательности определяется его **индексом**.

Массиву присваивается имя, посредством которого можно ссылаться на него, как на единое целое. Доступ к отдельному элементу осуществляется по имени массива с указанием индекса, который записывается после имени в квадратных скобках, например

$A[5]$ ,  $time[7]$ .

Массив является структурированным (составным) типом данных. Это означает, что величина, описанная как массив, состоит из конечного числа других величин. Так, например, можно создать массив из 20 целых или 1000 вещественных чисел.

Элементы массива в программах могут использоваться так же, как обычные переменные, например:

```
A[3]:=25; S:=(T[1]+T[31])/2;
A[k]:=B[k]*2; Sum:=Sum+A[i].
```

Массивы, обращение к элементам которых осуществляется при помощи только одного индекса, называют **одномерными**. Элементы такого массива располагаются в памяти компьютера цепочкой друг за другом.

A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]
------	------	------	------	------	------	------

Индексов может быть несколько, тогда массив называют **многомерным**. Пример двумерного массива — таблица. Заметим, что ее можно рассматривать, как совокупность одномерных массивов (столбцов и строк).

Массив в языке Pascal ABC описывается так:

имя: array[инд1..индN] of тип \_ элементов;

Здесь имя — имя массива (идентификатор) задается по тем же правилам, что и имена переменных других типов.

Служебное слово `array` означает массив.

В квадратных скобках задается диапазон индексов элементов массива: сначала указывают индекс первого элемента `инд1`, затем ставят две точки, после указывают индекс последнего элемента `индN`.

Диапазон индексов определяет максимально возможное количество элементов в массиве — **размер массива**.

Тип элементов может быть любым, например: числа, символы, строки, массивы. Мы будем рас-

сматривать массивы с элементами числовых типов `integer` (целый) и `real` (вещественный).

Напомним, что в языке Pascal ABC данные типа `integer` занимают 4 байт и находятся в диапазоне от  $-2\ 147\ 483\ 648$  до  $+2\ 147\ 483\ 647$ . Данные типа `real` занимают 8 байт и по модулю не превосходят  $1,7 \cdot 10^{308}$ .

Индексы мы будем задавать целыми числами: положительными или отрицательными.

Рассмотрим примеры описания массивов.

**Пример 1.** В классе 25 учеников. Известен рост каждого в сантиметрах. Для хранения значений роста можно использовать массив `A`, состоящий из 25 целых чисел. Индекс каждого элемента — порядковый номер ученика по списку в классном журнале. Например, элемент `A[5]` хранит рост ученика под пятым номером.

Описание этого массива может выглядеть так:

```
Var A: array[1..25] of integer;
```

Этот массив можно описать и так (задав диапазон константами `n1` и `n2`):

```
Const n1=1; n2=25;
```

```
Var A: array[n1..n2] of integer;
```

**Пример 2.** Каждый день с 20 по 31 декабря измеряли температуру воздуха. Для хранения значений температуры можно использовать массив `T`, состоящий из 12 вещественных чисел. Индекс элемента — дата (номер дня в декабре). Например, элемент `T[23]` содержит температуру воздуха 23 декабря.

Описание этого массива может выглядеть так:

```
Var T: array[20..31] of real;
```

этаж 3	$i = 3$
этаж 2	$i = 2$
этаж 1	$i = 1$
цоколь	$i = 0$
подвал	$i = -1$
подвал	$i = -2$

Рис. 2.1

**Пример 3.** В здании торгового центра 6 этажей: два подземных, цокольный и три верхних (рис. 2.1). Элементы массива Komnaty содержат количество комнат на каждом этаже. Индекс — обозначение этажа от  $-2$  до  $3$ . Размер этого массива 6 элементов.

Описание массива может выглядеть так:

```
Var Komnaty: array[-2..3] of integer;
```

Наиболее распространенная ошибка при работе с массивами — выход за границы изменения индекса. В Pascal ABC эта ошибка приводит к завершению работы программы.

- ? 1. Что называют массивом?  
 2. Что понимают под индексом элемента массива?  
 3. Как описываются массивы в языке Pascal?

### Упражнения

1. Заполните пропуски в описании массива из 80 элементов, содержащего целые числа. Числа пронумерованы от 1 до 80.

```
var a: array[1..__] of integer;
var a: array[__..80] of integer;
var a: array[1..80] of _____;
var a: array[__..__] of _____;
```

2. Опишите массив из 50 действительных чисел.

## § 6. Ввод и вывод элементов массива

Для работы с массивом необходимо присвоить значения его элементам. Сделать это можно несколькими способами.

Значения элементов, которые не изменяются при работе программы (констант), можно задавать в разделе описаний. Например, массив из 8 простых чисел может быть задан так:

```
Const A: array[1..8] of integer =  
  (2, 3, 5, 7, 11, 13, 17, 19);
```

Для присвоения значения элементу указывают имя массива и индекс этого элемента, например

```
A[4]:=7;
```

Для работы с несколькими элементами используют циклические алгоритмы. В тех случаях, когда заранее известно количество элементов, используют цикл `for`. Важно помнить, что значение параметра цикла (счетчик) `i` не может выходить за границы диапазона, заданного в описании массива.

Для ввода значений с клавиатуры используют стандартные процедуры `read` и `readln`. Например, следующая программа присваивает значения, вводимые с клавиатуры, пяти элементам массива.

```
Var A: array[1..5] of integer;  
  i: integer;  
Begin  
  for i:=1 to 5 do  
    read(A[i]);           {Ввод значений}  
End.
```

При использовании процедуры `read(A[i])` в строке ввода можно набрать значения нескольких

или сразу всех элементов массива, отделяя их друг от друга пробелами, и нажать Enter. При этом каждый набор отображается в окне вывода в одну строку (рис. 2.2, а). Если в строке ввода набрать больше пяти чисел, то элементам массива будут присвоены только первые пять значений. Если набрать меньше пяти чисел, то программа будет ожидать продолжения ввода.

При использовании процедуры `readln(A[i])` значения вводят по одному и после каждого нажимают Enter. Эти значения отображаются в окне вывода в столбец (рис. 2.2, б).

The image shows two side-by-side screenshots of the Delphi IDE. Both windows have a title bar with the Delphi logo and a status bar at the bottom.

**Window A (Left):**

```
var A:array[1..5] of integer;
i:integer;
begin
  for i:=1 to 5 do
    read(A[i]);
end.
```

Output window:

```
7 9 6
8 5
```

Input window (labeled "Ввод данных"):

```
8 5
```

**Window B (Right):**

```
var A:array[1..5] of integer;
i:integer;
begin
  for i:=1 to 5 do
    readln(A[i]);
end.
```

Output window:

```
7
9
6
8
5
```

Input window (labeled "Ввод данных"):

```
5
```

а

б

Рис. 2.2

Для вывода используют процедуры `write` и `writeln`. Процедура `write` выводит значения элементов массива в строку. При этом выводимые значения необходимо отделять пробелами или иными символами (например, запятой, точкой с запятой), иначе все они будут напечатаны слитно. Для вывода значений элементов в столбец используют процедуру `writeln`. Для вывода значений элементов в обратном порядке используют цикл `for ... downto...`.

Во многих случаях удобно использовать форматный вывод. Напомним, как задается формат вывода:

для целых чисел — X:m,

для вещественных чисел — X:m:n,

где m — ширина поля вывода, а n — количество знаков после десятичной точки.

Пусть, например, массив описан так:

```
const A: array[1..5] of integer =
  = (2, 3, 5, 7, 9);
```

Приведем некоторые примеры применения процедур write и writeln.

Команда	Вывод
for i:=1 to 5 do write (A[i]);	23579
for i:=1 to 5 do write (A[i], ',');	2, 3, 5, 7, 9,
for i:= 2 to 4 do write (A[i], ';');	3; 5; 7;
for i:= 5 downto 1 do write(A[i], ';');	9; 7; 5; 3; 2;
for i:=1 to 4 do writeln(A[i]);	2 3 5 7
for i :=1 to 4 do write (A[i]:5);	 позиций

Для удобства при вводе/выводе значений элементов массива можно использовать текстовые подсказки и указывать индекс, например: 'введите значение  $i$ -го элемента' или ' $i$ -й результат равен'.

Нередко значения элементов массивов не вводятся с клавиатуры, а вычисляются с помощью арифметических выражений. Например, массив из 9 нечетных чисел можно сформировать так:

```
for i:=1 to 9 do A[i]:= 2*i - 1;
```

Вычисление значений функции на заданном отрезке с определенным шагом называется *табулированием* функции. В 8-м классе с этой целью Вы использовали циклы. При этом значения функций не сохранялись — они вновь вычислялись на каждом шаге цикла.

Для хранения таблиц значений функций используют массивы. Аргументами в простейших случаях могут служить индексы элементов (либо арифметические выражения от индексов).

**Пример 1.** Составить программу, которая осуществляет табулирование функции  $y = x^2$ , сохранение результатов в массиве и вывод значения элементов массива в строку. Аргумент изменяется от  $-6$  до  $6$  с шагом  $1$ .

```
Program Primer6_1;
Var y: array[-6..6] of integer;
    i: integer;
Begin
    for i:=-6 to 6 do y[i]:=sqr(i);
                    {Формирование массива}
    for i:=-6 to 6 do write(y[i]:4);
                    {Форматный вывод в строку}
End.
```

Результат работы программы будет выглядеть так:

36	25	16	9	4	1	0	1	4	9	16	25	36
----	----	----	---	---	---	---	---	---	---	----	----	----

При решении многих практических задач используют числа, которые задаются случайным образом. Часто их применяют при тестировании и отладке программ, чтобы не набирать вручную исходные данные, например значения элементов массива.

Случайные значения элементов массива задаются с помощью стандартной функции `random`.

Функция `random` без аргумента генерирует случайные вещественные числа на промежутке  $[0, 1)$ .

Если случайные вещественные числа должны принадлежать иному промежутку, например  $[3, 4)$ , то значение элемента задается выражением  $x[i] := random + 3$ ; (рис. 2.3).

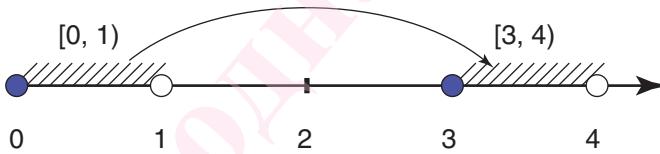


Рис. 2.3

**Пример 2.** Составить программу, которая формирует массив из семи случайных вещественных чисел, принадлежащих промежутку  $[3; 4)$ , и выводит пять первых чисел массива с двумя десятичными цифрами в столбец шириной 8 знаков.

```
Program Primer6_2;
Var X: array[1..7] of real;
    i: integer;
Begin
    for i:=1 to 7 do {Формирование массива}
```

```

X[i]:=random+3;
for i:=1 to 5 do
  writeln(X[i]:8:2);
  { Вывод значений первых пяти
    элементов в столбец шириной
    8 знаков, два десятичных }
End.

```

Результат может выглядеть так:

3.46
3.59
3.83
3.39
3.27

Целые случайные числа из промежутка  $[0; n]$  генерирует функция `random(n+1)`. Если целочисленные значения элементов массива должны принадлежать промежутку  $[a; b]$ , то их вычисляют с помощью выражения:

$$X[i] := \text{random}(b - a + 1) + a;$$

Например, массив из 10 случайных целых чисел, принадлежащих промежутку  $[20; 50]$ , можно сформировать так:

```

for i:= 1 to 10 do X[i]:= random(31) + 20;
                           { 50 - 20 + 1 = 31 }.

```

Элементы массива в программах могут использоваться так же, как обычные переменные. Покажем это на примере вывода графических объектов.

Напомним, что рисование производится в графическом окне и требует подключения модуля `GraphAbc`.

**Пример 3.** Составить программу, рисующую 9 концентрических окружностей (рис. 2.4), радиусы которых

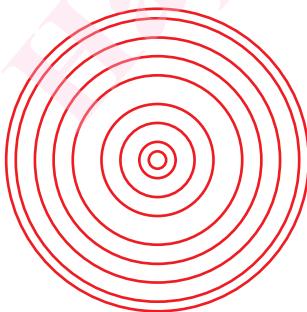


Рис. 2.4

хранятся в массиве случайных целых чисел, принадлежащих промежутку [20; 200].

Программа может быть такой:

```
Program Primer6_3;
Uses GraphABC;
    {Подключение графического модуля}
Var R: array[1..9] of integer;
    i: integer;
Begin
    for i:=1 to 9 do
        R[i]:=random(181) + 20;
            {Формирование массива из 9
             случайных целых чисел от 20 до 200}
    SetBrushStyle(bsClear);
        {Задание прозрачного фона}
    SetPenColor(clRed);
        {Задание красного цвета пера}
    for i:=1 to 9 do
        Circle(240, 240, R[i]);
            {Рисование окружностей}
End.
```

- ? 1. Какие процедуры используются для ввода значений элементов массива?
- 2. Какие процедуры используются для вывода значений элементов массива?
- 3. Какая функция используется для формирования массива случайных чисел?

### Упражнения

1. Заполните пропуски в программе ввода и вывода 10 целочисленных элементов массива, заданных случайным образом на промежутке [20; 40]:

```
Var A: array[1..____] of integer;
    i: integer;
```

```
Begin
    for i:=1 to __ do
        begin
            A[i]:=random(____) + ____;
            write(____);
        end;
End.
```

2. Составьте программу для ввода с клавиатуры в массив 7 целых чисел и вывода этих чисел в обратном порядке.

3. Составьте программу для ввода и вывода температуры за неделю.

Результат может быть таким:

```
1 сентября температура была 17
2 сентября температура была 15
3 сентября температура была 16
4 сентября температура была 19
```

4. Составьте программу, которая формирует массив из двенадцати случайных целых чисел, принадлежащих промежутку  $[-5; 5]$ , и выводит эти числа в столбец.

5. Составьте программу, которая формирует массив из десяти случайных вещественных чисел, принадлежащих промежутку  $[4; 5)$ , и выводит числа с 3-го по 7-й в столбец с двумя десятичными цифрами.

## § 7. Арифметические действия над элементами массива

Элементы массива в программах используются так же, как обычные переменные. Над ними можно выполнять операции, допустимые для перемен-

ных соответствующего типа. Элементам типа `integer` можно присваивать результат выполнения операций `+`, `-`, `*`, `div`, `mod` (участвовать в качестве operandов они могут и в операции `/`). Для элементов типа `real` возможны операции `+`, `-`, `*`, `/`.

Рассмотрим примеры выполнения арифметических действий над элементами одномерных числовых массивов.

**Пример 1.** В массиве хранятся данные о количестве осадков (в мм), выпадавших ежемесячно на протяжении года. Составить программу, которая вычисляет суммарное количество осадков за год, а также среднемесячное количество осадков.

Массив данных о количестве осадков за 12 месяцев зададим в виде целочисленных констант:

```
const h: array[1..12] of integer =(43, 40,  
42, 46, 61, 82, 90, 81, 60, 50, 53, 50);
```

Вычисленную сумму будем помещать в переменную `Sum` целого типа, а среднее значение — в переменную `Sred` вещественного типа.

Последовательность действий при вычислении суммы и среднего арифметического такова:

- Присвоим начальное значение суммы `Sum:= 0`.

- Вычисление суммы будем производить в цикле `for`. Значение параметра (счетчика) цикла `i` изменяется от 1 до 12. На каждом шаге цикла к значению переменной `Sum` будем прибавлять значение очередного элемента `Sum:=Sum+h[i]`.

- По завершении цикла вычислим среднее значение `Sred:=Sum/12` и выведем результаты.

Программа имеет вид:

```
Program Primer7_1;
Const h: array[1..12] of integer=(43, 40,
        42, 46, 61, 82, 90, 81, 60, 50, 53, 50);
Var Sum, i: integer;
    Sred: real;
Begin
    Sum:=0;      {Начальное значение суммы}
    for i:=1 to 12 do Sum:=Sum+h[i];
                    {Вычисление суммы}
    Sred:=Sum/12; {Вычисление среднего}
    write ('Суммарное количество = ', Sum,
    ' Среднемесячное количество = ',
    Sred:5:2);
End.
```

Протестируем программу. В окне вывода получим:

Суммарное количество = 698 Среднемесячное количество = 58.17
--

Алгоритм вычисления произведения элементов числового массива отличается лишь тем, что начальное значение  $P:=1$ . На каждом шаге цикла выполняется умножение  $P$  на значение очередного элемента  $P:=P*A[i]$ .

В некоторых задачах требуется выполнять арифметические действия не над всеми элементами массива, а лишь над удовлетворяющими определенным условиям, например положительными или отрицательными, четными или нечетными числами, значениями, кратными заданному числу.

**Пример 2.** Составить программу, которая формирует массив из 12 случайных целых чисел, принадлежащих промежутку  $[-40; 40]$ , и вычисляет сумму положительных чисел.

Сформируем массив случайных чисел на указанном промежутке.

Вычисленную сумму будем помещать в переменную S. Присвоим начальное значение S:=0.

Суммирование будет производиться в цикле for при выполнении условия A[i]>0.

Программа может выглядеть так:

```
Program Primer7_2;
Var A: array[1..12] of integer;
    i, S: integer;
Begin
    for i:=1 to 12 do A[i]:=random(81)-40;
                    {Формирование массива}
    S:=0;
    for i:=1 to 12 do
        if A[i]>0 then S:=S+A[i];
    {Вычисление суммы положительных чисел}
    for i:=1 to 12 do write(A[i]:4);
                    {Форматный вывод чисел в строку}
    writeln;      {Переход на новую строку}
    writeln('Сумма положительных чисел =
          ',S);           {Вывод суммы}
End.
```

Результат работы может выглядеть так:

```
-21   -7   35   -9   10   36   22   -19   1   -34   15   37
Сумма положительных чисел = 156
```

Аналогичным способом выполняются арифметические действия над элементами массива, значения которых удовлетворяют, например, условиям:

```
if A[i] mod 2 = 0 then... {четные числа}
if A[i] mod 2 = 1 then... {нечетные числа}
if A[i] mod N = 0 then... {числа, кратные N}
```

**Пример 3.** Составить программу, которая формирует массив из 7 случайных целых чисел, принадлежащих промежутку [1; 5], и вычисляет произведение нечетных чисел.

Сформируем массив случайных чисел на указанном промежутке.

Вычисленное произведение будем помещать в переменную P. Начальное значение P:=1.

Умножать P на значение очередного элемента будем только на тех шагах цикла for, где выполняется условие  $A[i] \bmod 2 = 1$ .

Программа может выглядеть так:

```
Program Primer7_3;
Var A: array[1..7] of integer;
    i, P: integer;
Begin
    for i:=1 to 7 do A[i]:=random(5)+1;
                    {Формирование массива}
    P:=1;
    for i:=1 to 7 do
        if A[i] mod 2 = 1 then P:=P*A[i];
                    {Вычисление произведения}
    for i:=1 to 7 do write(A[i]:4);
                    {Форматный вывод чисел в строку}
    writeln;
    writeln('Произведение нечетных чисел = ',P); {Вывод произведения}
End.
```

Результат работы может выглядеть так:

2	1	4	5	4	5	3
Произведение нечетных чисел = 75						

**Пример 4.** В массиве A хранятся данные о расходе топлива на 100 км пробега автомобилей восьми

моделей. При замене топлива на новую марку расход уменьшается на 10 %. Составить программу, которая вычисляет расход нового топлива и выводит в таблицу номер модели автомобиля, расход старого и нового топлива с двумя десятичными цифрами.

Пусть данные о расходе топлива хранятся в массиве констант вещественного типа:

```
const A: array[1..8] of real = (7.4, 7.0,  
6.8, 5.9, 6.4, 9.1, 7.6, 8.2);
```

Расход нового топлива будем вычислять непосредственно в процедуре вывода `writeln`. Это позволит обойтись без создания второго массива. Для помещения данных в таблицу используем форматный вывод.

Программа может выглядеть так:

```
Program Primer7_4;  
Const A: array[1..8] of real = (7.4, 7.0,  
6.8, 5.9, 6.4, 9.1, 7.6, 8.2);  
Var i: integer;  
Begin  
    writeln('Модель Расход-стар Расход-нов');  
    {Заголовок таблицы}  
    for i:=1 to 8 do  
        writeln(i:4,A[i]:10:2, 0.9*A[i]:10:2);  
        {Вычисление и вывод}  
End.
```

Результат работы будет выглядеть так:

Модель	Расход-стар	Расход-нов
1	7.40	6.66
2	7.00	6.30
3	6.80	6.12
4	5.90	5.31
5	6.40	5.76
6	9.10	8.19
7	7.60	6.84
8	8.20	7.38

- ? 1. Какие операции допустимы для элементов массива целых чисел?
2. Какие операции применимы для элементов массива вещественных чисел?

### Упражнения

1. Составьте программу, которая вычисляет среднюю плотность раствора (в кг/м<sup>3</sup>) по результатам измерения плотности 7 проб (задаются в массиве констант: 1205, 1350, 1170, 1320, 1220, 1310, 1180).

2. Составьте программу, которая осуществляет ввод в массив с клавиатуры размеров 6 файлов в байтах (от 100 до 2000) и вычисляет:

- суммарный размер файлов в килобайтах;
- средний размер файла.

3. Составьте программу, которая формирует массив из 40 случайных целых чисел от 0 до 20 и вычисляет:

- среднее арифметическое нечетных чисел;
- сумму чисел, не превышающих 10;
- сумму чисел, кратных 3.

На экран выводятся исходные значения и результаты вычислений.

## § 8. Преобразование элементов массива

При работе с массивами нередко приходится выполнять различные преобразования их элементов, например преобразовывать единицы измерения, масштабы и координаты, изменять значения элементов в зависимости от выполнения некоторых условий, менять местами элементы в массиве.

Рассмотрим простейшие преобразования элементов массива на примерах.

**Пример 1.** Массив содержит значения диагонали 8 мониторов в дюймах: 12, 14, 15, 17, 19, 20, 22, 24. Составить программу перевода этих значений в сантиметры и вывода с округлением до целых.

Сформируем исходный массив, присвоив значения каждому элементу. Использовать массив констант нельзя, поскольку в этом случае значения элементов нельзя будет изменить.

Преобразования единиц измерения будем производить в цикле `for`, учитывая, что 1 дюйм = 2,54 см. Функция округления `round()` возвращает значения целого типа, поэтому возможен тип массива `integer`.

Программа может выглядеть так:

```
Program Primer8_1;
Var D: array[1..8] of integer;
    i: integer;
Begin
    {Формирование массива}
    D[1]:=12; D[2]:=14; D[3]:=15; D[4]:=17;
    D[5]:=19; D[6]:=20; D[7]:=22; D[8]:=24;
    writeln('Диагональ монитора в дюймах');
    for i:=1 to 8 do write(D[i]:4);
    {Вывод значений в дюймах}
    writeln;
    writeln('Диагональ монитора в сантиметрах');
    for i:=1 to 8 do
        D[i]:=round(D[i]*2.54);
    {Преобразование дюймов в сантиметры}
    for i:=1 to 8 do write(D[i]:4);
    {Вывод значений в сантиметрах}
End.
```

Результат работы может выглядеть так:

Диагональ монитора в дюймах
12 14 15 17 19 20 22 24
Диагональ монитора в сантиметрах
30 36 38 43 48 51 56 61

Аналогичным способом выполняются преобразования единиц измерения, например: скорости из м/с в км/ч, температуры из градусов Цельсия ( $T_C$ ) в Кельвины ( $T_K = T_C - 273$ ) и т. п.

Рассмотрим замену абсолютных значений величин процентами.

**Пример 2.** В опытах по исследованию растворимости в 40 г воды растворяли 3, 12, 17, 22, 27, 32 и 38 г вещества. Составить программу, которая осуществляет ввод в массив масс растворенного вещества и замену их массовыми долями в процентах.

Организуем ввод исходных данных (масс растворенного вещества в граммах) в целочисленный массив  $M$ . Замену масс  $M[i]$  их массовыми долями в растворе  $100 * M[i] / (40 + M[i])$  будем производить в цикле `for`.

Функция округления `round()` возвращает значения целого типа, поэтому и после замены возможен тип массива `integer`.

Программа будет выглядеть так:

```
Program Primer8_2;
Var M: array[1..7] of integer;
    i: integer;
Begin
    writeln('Введите значения массы');
    for i:=1 to 7 do read(M[i]);
    {Ввод масс}
```

```
for i:=1 to 7 do
    M[i]:=round(100*M[i]/(40+M[i]));
    {Замена процентами}
    writeln('Массовая доля в %');
    for i:=1 to 7 do write(M[i], ' ');
End.
```

Результат работы может выглядеть так:

```
Введите значения массы
3 12 17 22 27 32 38
Массовая доля в %
7 23 30 35 40 44 49
```

В некоторых задачах требуется выполнять преобразования не всех элементов массива, а лишь удовлетворяющих определенным условиям, например с отрицательными, четными значениями или значениями, кратными заданному числу.

**Пример 3.** Составить программу, которая формирует массив из 12 случайных целых чисел, принадлежащих промежутку  $[-20, 20]$ , заменяет отрицательные числа на  $-1$  и выводит значения элементов массива до и после замены.

Организуем формирование массива из 12 случайных целых чисел, принадлежащих промежутку  $[-20, 20]$ :  $A[i]:=random(41)-20;$

Преобразования элементов будем производить в цикле `for` при выполнении условия:

```
if A[i]<0 then A[i]:=-1;
```

Программа может выглядеть так:

```
Program Primer8_3;
Var A: array[1..12] of integer;
    i: integer;
Begin
```

```

for i:=1 to 12 do A[i]:=random(41)-20;
                {Формирование массива}
for i:=1 to 12 do write (A[i]:4);
                {Вывод элементов до замены}
writeln;
for i:=1 to 12 do
    if A[i]<0 then A[i]:=-1;
    {Замена отрицательных чисел на -1}
for i:=1 to 12 do write (A[i]:4);
                {Вывод элементов после замены}
End.

```

Результат работы программы может выглядеть так:

-4	6	10	-15	17	0	-14	-9	19	2	1	16
-1	6	10	-1	17	0	-1	-1	19	2	1	16

Аналогичным способом производится замена элементов массива по иным условиям, например:

- замена всех четных значений элементов массива квадратами их номеров:

```
if A[i] mod 2 = 0 then A[i]:=sqr(i);
```

- замена всех отрицательных чисел их квадратами, а неотрицательных — значениями квадратного корня:

```
if A[i]<0 then A[i]:=sqr(A[i])
else A[i]:=sqrt(A[i]);
```

Заметим, что при использовании функции `sqrt()` требуется тип массива `real` (вещественный).

Перестановка (обмен местами) двух элементов массива сводится к обмену их значениями. Для этого используют дополнительную переменную (которую называют буфером обмена).

Пусть, например, требуется поменять местами 2-й и 5-й элементы массива (рис. 2.5).

Алгоритм обмена заключается в том, что в буфер помещают значение заменяемого элемента массива  $B := A[2]$ . Затем этому элементу присваивают значение второго (заменяющего его) элемента  $A[2] := A[5]$ , которому, в свою очередь, присваивают значение буфера обмена  $A[5] := B$  (см. рис. 2.5).

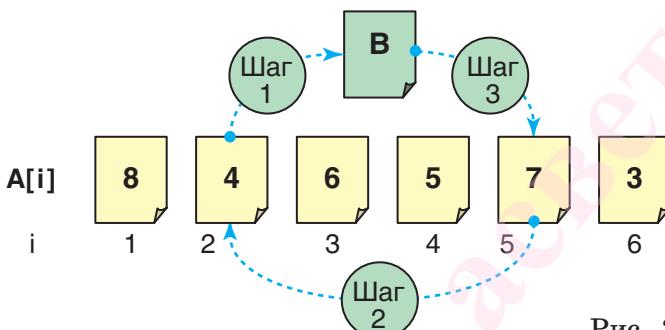


Рис. 2.5

В общем виде последовательность операций такова:

$$B := A[i]; \quad A[i] := A[k]; \quad A[k] := B;$$

Если попытаться поменять местами элементы без буфера, например так:  $A[i] := A[k]; \quad A[k] := A[i]$ , то мы потеряли первоначальное значение элемента  $A[i]$  и получим два элемента со значением  $A[k]$ .

**Пример 4.** Составить программу, которая формирует массив из 15 случайных целых чисел от 0 до 20, меняет местами 3-й и 12-й элементы и выводит значения элементов массива на экран до и после обмена.

Сформируем массив  $A$ : `for i:=1 to 15 do A[i]:=random(21)`. Номера элементов соответствуют индексам массива.

Поменяем местами элементы:

$$B := A[3]; \quad A[3] := A[12]; \quad A[12] := B;$$

Программа может выглядеть так:

```
Program Primer8_4;
Var A: array[1..15] of integer;
    B, i: integer;
Begin
    for i:=1 to 15 do A[i]:=random(21);
                    {Формирование массива}
    for i:=1 to 15 do write (A[i]:4);
                    {Вывод элементов до обмена}
    writeln;
    B:=A[3]; A[3]:=A[12]; A[12]:=B;
                    {Обмен значениями}
    for i:=1 to 15 do write (A[i]:4);
                    {Вывод элементов после обмена}
End.
```

Результат работы программы может выглядеть так:

6	15	2	13	17	0	0	2	4	11	4	18	17	15	17
6	15	18	13	17	0	0	2	4	11	4	2	17	15	17

- ? 1. Какие преобразования элементов массива возможны?  
 2. С какой целью используют дополнительную переменную (буфер) при обмене местами элементов массива?

### Упражнения

1. Составьте программу, которая осуществляет преобразование единиц измерения физических величин:
  - значения скорости, измеренные в км/ч, преобразуются в значения, измеренные в м/с ( $1 \text{ м/с} = 3,6 \text{ км/ч}$ );
  - значения мощности, измеренные в лошадиных силах, преобразуются в значения, измеренные в киловаттах ( $1 \text{ л. с.} = 0,735 \text{ кВт}$ );

в) значения давления, измеренные в миллиметрах ртутного столба, преобразуются в значения, измеренные в килопаскалях (1 мм. рт. ст. = = 0,133 кПа).

Исходные данные вводятся в массив с клавиатуры, результат выводится на экран.

2. Составьте программу, которая осуществляет ввод в массив с клавиатуры размеров 5 файлов в байтах (от 100 до 9000 байт) и преобразует значения, превышающие 1024 байт, в килобайты. Результат выводится в строку.

3. Составьте программу, которая формирует массив из пятнадцати случайных целых чисел от -20 до 20 и:

а) четные числа делит на два, а нечетные — умножает на три;

б) отрицательные числа возводит в квадрат, а из неотрицательных извлекает квадратный корень;

в) увеличивает в 2 раза числа, кратные 5, а остальные уменьшает в 2 раза;

г) меняет местами значения первого и последнего элементов массива.

На экран выводятся исходные и преобразованные значения.

## § 9. Поиск элементов с заданными свойствами

При обработке информации постоянно приходится сталкиваться с задачами поиска данных. Эти задачи весьма разнообразны: от поиска телефонного номера или справочных данных до проверки правильного ответа в тестах или угадывания числа в

играх. Алгоритмы поиска являются одними из наиболее часто выполняемых алгоритмов.

Пусть, например, в массиве  $h$  хранится рост учеников 9-го класса (рис. 2.6).

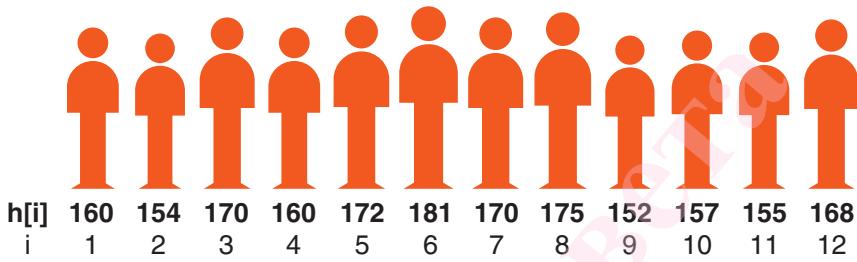


Рис. 2.6

Для этого массива можно, например, сформулировать такие задачи поиска:

- Определить, есть ли хотя бы один ученик, рост которого равен 172 см (или больше 180 см).
- Найти номер по списку (индекс) ученика, рост которого 172 см.
- Определить, сколько учеников в классе имеют рост 170 см (или меньше роста первого по списку ученика).
- Найти номера (индексы) всех учеников ростом 170 см (или больше 175 см).
- Найти номер ученика максимального (минимального) роста.

Обобщая эти задачи, можно сказать, что цель поиска заключается в нахождении значений, индексов и количества элементов массива, удовлетворяющих заданным условиям.

В качестве простых условий поиска чаще всего используется сравнение значений элементов массива  $X[i]$  с заданным числом  $B$  (например, равенство  $X[i]=B$  или неравенства  $X[i]<B$  или  $X[i]>=B$ ).

Известны различные алгоритмы поиска, каждый из которых эффективен для решения определенного круга задач.

Самый простой способ поиска элементов массива с заданными свойствами — это последовательный просмотр всех элементов и проверка выполнения условий поиска. Такой алгоритм поиска называется **линейным** или **последовательным**.

Рассмотрим примеры решения некоторых задач поиска.

**Пример 1.** В массиве хранится рост 12 учеников 9-го класса (см. рис. 2.6). Составить программу, которая определяет, есть ли в классе хотя бы один ученик ростом 172 см, и выводит его номер по списку.

Это простейшая задача поиска элемента массива, значение которого равно заданному. Алгоритм ее решения сводится к последовательному сравнению значений всех элементов массива с заданным числом.

Массив данных о росте учеников зададим в виде целочисленных констант:

```
const h: array[1..12] of integer=
  = (160, 154, 170, 160, 172, 181,
    170, 175, 152, 157, 155, 168);
```

Результаты поиска будем помещать в переменную *k*. Если условие не выполнено ни разу (элемент не найден), *k*:=0. Если условие выполнено (элемент найден), то переменной *k* присваивается индекс найденного элемента *k*:=*i*.

Рассмотрим алгоритм поиска.

- Присвоим начальное значение *k*:=0.
- Проверять выполнение условий поиска (сравнение значений) будем в цикле *for*. На каждом

шаге цикла от 1 до 12 будем сравнивать значение очередного элемента  $h[i]$  с числом 172.

На первых четырех шагах условие  $h[i]=172$  не выполняется, значение переменной  $k$  остается равным нулю. Это условие будет выполнено на пятом шаге. Элемент найден, переменной  $k$  присваивается значение его индекса  $k:=5$ . Далее на шагах 6—12 это значение  $k$  сохраняется, поскольку в данном примере искомый элемент единственный.

- По завершении цикла выведем индекс элемента (номер ученика по списку) либо сообщение 'ученика с таким ростом нет'.

Программа выглядит так:

```
Program Primer9_1;
Const h: array[1..12] of integer = (160, 154,
170, 160, 172, 181, 170, 175, 152, 157,
155, 168);
Var i, k: integer;
Begin
    k:=0; {Начальное значение k}
    for i:=1 to 12 do {Просмотр массива}
        if h[i]=172 then k:=i;
    if k>0 then writeln('Номер ученика
по списку = ', k) {Вывод}
    else writeln('Ученика с таким
ростом нет');
End.
```

Результат работы будет выглядеть так:

Номер ученика по списку = 5
-----------------------------

В рассмотренном примере искомый элемент был единственным. Если элементов, удовлетворяющих

условиям поиска, несколько, то после нахождения первого из них возможны два варианта действий:

1) продолжить выполнение цикла до конца, тогда в переменную  $k$  будет помещен индекс последнего из найденных элементов;

2) прервать цикл командой `break`, тогда в  $k$  останется индекс первого из найденных элементов.

**Пример 2.** В массиве хранится рост учеников 9-го класса (см. рис. 2.6). Составить программу, которая подсчитывает, сколько учеников имеют рост больше 170 см.

Массив данных о росте учеников зададим в виде целочисленных констант.

Условием поиска в этой задаче является неравенство  $A[i] > 170$ . Результат поиска будем помещать в переменную  $k$  (счетчик найденных элементов). При выполнении условия поиска значение этой переменной увеличивается на 1.

Рассмотрим алгоритм поиска.

- Присвоим начальное значение  $k := 0$ .
- Проверять выполнение условий поиска будем в цикле `for`. На каждом шаге цикла от 1 до 12 будем сравнивать значение очередного элемента  $A[i]$  с числом 170.

На четырех шагах условие  $A[i] > 170$  не выполняется, значение переменной  $k$  остается равным нулю. Это условие будет выполнено на пятом шаге. Первый искомый элемент обнаружен, значение счетчика стало  $k := 1$ . В дальнейшем условия поиска выполняются еще дважды: на 6-м и на 8-м шагах. В результате  $k := 3$ . Далее это значение остается неизменным.

- По завершении цикла выведем значение  $k$ .

Программа будет выглядеть так:

```
Program Primer9_2;
Const h: array[1..12] of integer = (160,
    154, 170, 160, 172, 181, 170, 175,
    152, 157, 155, 168);
Var i, k: integer;
Begin
    k:=0; {Начальное значение k}
    for i:=1 to 12 do {Просмотр массива}
        if h[i]>170 then k:=k+1; {Подсчет k}
    writeln('Таких учеников ', k);
End.
```

Протестируем программу. В окне вывода получим сообщение:

Таких учеников 3

Приведем примеры некоторых условий, которые часто используют при поиске элементов:

Значение элемента	Условие
Положительное число	$A[i] > 0;$
Отрицательное число	$A[i] < 0;$
Число, не равное нулю	$A[i] \neq 0;$
Четное число	$A[i] \bmod 2 = 0;$
Четное число, исключая нуль	$(A[i] \bmod 2 = 0) \text{ and } (A[i] \neq 0);$
Нечетное число	$X[i] \bmod 2 = 1;$
Число, кратное целому числу В	$A[i] \bmod B = 0;$

**Пример 3.** Составить программу, которая формирует массив из 15 случайных целых чисел от 0 до 50 и выводит на экран нечетные элементы массива и их индексы.

Сформируем массив случайных чисел

$A[i]:=random(51)$ .

В цикле зададим условие нечетности значения элемента  $if A[i] \bmod 2 = 1$  (остаток от целочисленного деления на 2 равен 1).

Организуем вывод нечетных значений элементов и их индексов:

```
writeln('Найден элемент A[', i, '] = ', A[i]);
```

Если элемент не найден, никакое сообщение не выводится.

Для проверки выведем также все значения сформированного массива.

Программа будет выглядеть так:

```
Program Primer9_3;
Var A: array[1..15] of integer;
    i: integer;
Begin
    for i:=1 to 15 do A[i]:=random(51);
                    {Формирование массива}
    for i:=1 to 15 do {Просмотр массива}
        if A[i] mod 2 = 1 then
            {Проверка условия}
            writeln('Найден элемент A[', i,
' ] = ', A[i]);
                    {Вывод нечетных чисел}
    for i:=1 to 15 do write(a[i]:4);
                    {Вывод всех чисел}
End.
```

Результат работы может выглядеть так:

```
Найден элемент A[3] = 21
Найден элемент A[6] = 37
Найден элемент A[8] = 25
Найден элемент A[12] = 23
Найден элемент A[14] = 49
 20 10 21 48 32 37 24 25 34 24 6 23 30 49 18
```

Очень часто для решения задач требуется находить максимальный (наибольший) или минимальный (наименьший) элемент массива. Например, определить рост самого высокого ученика в классе, определить, какого числа был самый холодный день в декабре.

**Пример 4.** Составить программу, которая формирует массив из 7 случайных целых чисел от 0 до 50 и осуществляет поиск максимального элемента.

Рассмотрим алгоритм поиска максимального элемента на примере сформированного случайным образом массива A из 7 элементов (рис. 2.7).



Рис. 2.7

Значение максимального элемента будем хранить в переменной `max`, а его индекс — в переменной `k`.

Вначале полагаем, что максимальным является первый элемент:

$$\text{max} := \text{A}[1], \quad k := 1.$$

Простейший алгоритм поиска сводится к последовательному сравнению со значением `max` всех элементов массива `A[i]`, начиная со второго.

Организуем цикл `for` с параметром `i`, изменяющимся от 2 до 7. На каждом шаге цикла будем проверять условие `A[i] > max`. Если оно выполнится (очередной элемент больше `max`), то значение этого элемента примем в качестве максимального `max := A[i]`, а его индекс присвоим переменной `k := i`. В нашем примере новое значение

ние max присваивается на третьем шаге ( $A[3] > 15$ ,  $max := 45$ ,  $k := 3$ ) и на четвертом шаге ( $A[4] > 45$ ,  $max := 48$ ,  $k := 4$ ).

После перебора всех элементов массива переменная max будет содержать значение максимального элемента массива, а переменная k — его индекс.

Программа может выглядеть так:

```
Program Primer9_4;
Var A: array[1..7] of integer;
    max, k, i: integer;
Begin
    for i:=1 to 7 do A[i]:=random(51);
                    {Формирование массива}
    max:=A[1]; k:=1;
                    {Начальные значения max и k}
    for i:=2 to 7 do {Просмотр массива}
        if A[i]>max then
            {Условие поиска максимума}
            begin
                max:=A[i]; k:=i;
                {Присвоение значений max и k}
            end;
    writeln('Максимальный элемент A[ ', k,
    ' ] = ', max);           {Вывод максимума}
    for i:=1 to 7 do write(a[i]:4);
                    {Вывод всех чисел}
End.
```

Результат работы может выглядеть так:

```
Максимальный элемент A[4] = 48
15 13 45 48 18 7 41
```

Если в массиве несколько элементов имеют максимальное значение, то в переменной  $k$  будет запоминаться индекс первого из них. Для запоминания индекса последнего из максимальных элементов необходимо использовать условие  $A[i] >= \max$ .

Для поиска минимального элемента необходимо заменить знак « $>$ » в условии оператора ветвления на знак « $<$ ».

Заметим, что в алгоритме нахождения максимального или минимального элемента можно искать индекс этого элемента и не использовать промежуточную переменную.

**Пример 5.** В массив с клавиатуры вводятся результаты соревнований по бегу двенадцати спортсменов (значения времени от 50 до 100 с). Составить программу, которая находит номер и время победителя.

Результаты участников соревнований будем вводить с клавиатуры в массив  $A[i]$ . Номер победителя (индекс участника, показавшего минимальное время) будем хранить в переменной  $k$ . Тип всех переменных `integer`.

Вначале полагаем, что минимальное время показал участник номер 1. Присвоим переменной  $k$  его индекс  $k := 1$ .

В цикле зададим условие поиска индекса участника с минимальным временем:

```
if A[i] < A[k] then k := i;
```

По окончании цикла выведем результаты всех участников, номер и время победителя.

Программа будет выглядеть так:

```
Program Primer9_5;
Var A: array[1..12] of integer;
    i, k: integer;
```

```
Begin
    for i:=1 to 12 do read(A[i]);
                    {Ввод результатов}
    k:=1;           {Начальное значение k}
    for i:=2 to 12 do {Просмотр массива}
        if A[i]<A[k] then k:=i;
                    {Индекс минимального элемента}
    writeln('Победил участник номер ', k,
           ' с результатом ', A[k]);
    {Вывод номера и результата победителя}
End.
```

Результат работы может выглядеть так:

```
68 74 93 67 87 84 79 65 72 92 75 82
Победил участник номер 8 с результатом 65
```

- ? 1. Какой алгоритм поиска называют последовательным?  
2. На елочном базаре случайным образом берут несколько елок. Их высота в метрах заносится в массив вещественных чисел. Какие задачи поиска могут быть сформулированы для этих данных?

### Упражнения

1. Составьте программу, которая формирует массив из шестнадцати случайных целых чисел от  $-20$  до  $20$  и:
  - находит элементы с нулевыми значениями;
  - находит индексы четных элементов;
  - подсчитывает количество отрицательных элементов;
  - определяет, есть ли в этом массиве хотя бы одно число, кратное трем;
  - находит в этом массиве индексы элементов, значения которых кратны трем.

На экран выводятся исходный массив и найденные элементы.

2. В массиве хранятся сведения о количестве осадков (в мм), выпадавших ежедневно в сентябре. Составьте программу, которая подсчитывает количество дождливых дней ( $h[i] > 0$ ).

3. В массиве хранится информация о среднедневной температуре декабря. Составьте программу, которая подсчитывает, сколько в декабре было дней с нулевой, отрицательной и положительной температурой.

4. В массиве хранится информация о стоимости товаров. Составьте программу, которая определяет стоимость самого дешевого (дорогого) товара и его индекс.

### § 10. Решение задач с использованием массивов

Рассмотрим алгоритмы решения задач с использованием одномерных числовых массивов на примерах из различных предметных областей.

Решение любой задачи с помощью компьютера содержит несколько этапов:

- I. Определение исходных данных (что дано?).
- II. Определение результатов (что требуется найти?).
- III. Описание переменных (определение типов данных).
- IV. Составление алгоритма решения задачи.
- V. Написание и отладка программы.
- VI. Тестирование программы.

**Пример 1.** Составить программу, которая осуществляет ввод в массив значений сопротивления  $n$  резисторов и расчет общего сопротивления электрической цепи при последовательном соединении этих резисторов (рис. 2.8).



Рис. 2.8

I. Исходными данными являются количество резисторов и сопротивление каждого из них. Будем вводить эти данные с клавиатуры и помещать соответственно в переменную *n* и массив *R*. Индексом массива будет номер резистора от 1 до *n*.

II. Требуется найти общее сопротивление цепи. Будем хранить его в переменной *Rs*.

III. Пусть максимальное количество соединяемых резисторов (размер массива) равно 20, а их сопротивления измеряются в омах и имеют целочисленные значения. Тогда все переменные имеют тип *integer* и могут быть описаны так:

```

var R: array[1..20] of integer;
    n, i, Rs: integer;
  
```

IV. Алгоритм решения задачи основан на использовании формулы: общее сопротивление цепи при последовательном соединении равно сумме сопротивлений всех резисторов  $Rs = R_1 + R_2 + \dots + R_n$ .

Вычисление суммы будем производить в цикле *for*. В качестве начального значения возьмем сопротивление первого резистора  $Rs := R[1]$  (подключен один резистор). Тогда значение счетчика цикла *i* изменяется от 2 до *n*. На каждом шаге цикла к цепи подключается еще один резистор (к значению *Rs* прибавляется сопротивление очередного резистора  $R[i]$ ).

V. Программа имеет вид:

```

Program Primer10_1;
Var R: array[1..20] of integer;
    n, i, Rs: integer;
  
```

```

Begin
    write( 'Введите количество резисторов
(<=20)' );
    readln(n);
    writeln('Введите их сопротивления в омах ');
    for i:=1 to n do read(R[i]);
                                {Ввод сопротивлений}
    Rs:=R[1];   {Задание начального значения}
    for i:=2 to n do Rs:=Rs + R[i];
                    {Вычисление сопротивления цепи}
    writeln('Общее сопротивление цепи = ',
    Rs, ' Ом');           {Вывод}
End.

```

VI. Протестируем программу. Результат может выглядеть так:

```

Введите количество резисторов (<=20) 7
Введите их сопротивления в омах
47 120 51 100 47 75 91
Общее сопротивление цепи = 531 Ом

```

**Пример 2.** Составить программу, которая осуществляет ввод в массив значений сопротивления  $n$  резисторов и рассчитывает общее сопротивление электрической цепи при параллельном соединении этих резисторов (рис. 2.9).

Решение этой задачи аналогично решению предыдущей задачи. Отличие в том, что алгоритм решения основан на использовании формулы расчета

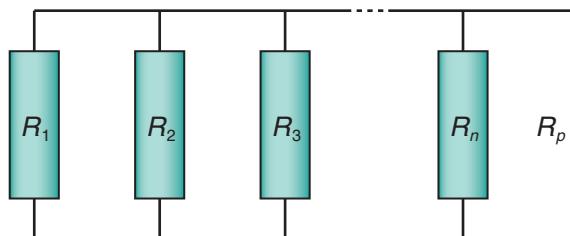


Рис. 2.9

общего сопротивления цепи при параллельном соединении резисторов (суммируются обратные величины):  $1/R_p = 1/R_1+1/R_2+\dots+1/R_n$ .

Переменная  $R_p$  должна иметь вещественный тип (либо преобразовываться в целый тип одной из функций округления). Описание переменных принимает вид:

```
var R: array[1..20] of integer;
    n, i: integer;
    Rp: real;
```

Вычисления будем производить в цикле `for`. В качестве начального значения возьмем сопротивление первого резистора  $R_p:=R[1]$ . Значение счетчика цикла  $i$  будет изменяться от 2 до  $n$ . На каждом шаге цикла к цепи подключается еще один резистор (к значению  $1/R_p$  прибавляется величина  $1/R[i]$ , обратная сопротивлению очередного резистора).

Программа имеет вид:

```
Program Primer10_2;
Var R: array[1..20] of integer;
    n, i: integer;
    Rp: real;
Begin
    write('Введите количество резисторов (<=20)');
    readln(n);
    writeln('Введите их сопротивления в омах');
    for i:=1 to n do read(R[i]);
                                {Ввод сопротивлений}
    Rp:=R[1]; {Задание начального значения}
    for i:=2 to n do
```

```

Rp:=1/(1/Rp + 1/R[i]);
{Вычисление сопротивления цепи}
writeln('Общее сопротивление цепи =',
Rp:6:2, ' Ом'); {Вывод}
End.

```

Протестируем программу. Результат может выглядеть так:

```

Ведите количество резисторов (<=20) 7
Ведите их сопротивления в омах
47 120 51 100 47 75 91
Общее сопротивление цепи = 9.54 Ом

```

**Пример 3.** В лабораторной работе измеряют массу  $m$  и ребро  $a$  пяти образцов кубической формы (рис. 2.10). Составить программу для расчета плотности вещества. Данные измерений вводятся с клавиатуры. Результаты измерений и вычислений выводятся в таблицу.

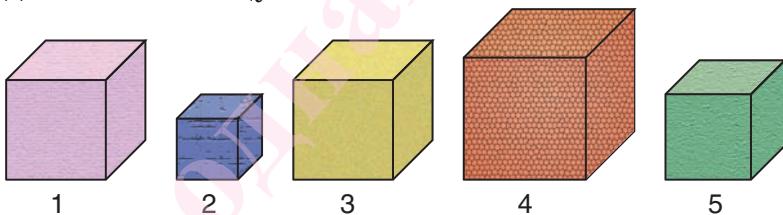


Рис. 2.10

Исходными данными являются масса и размер образцов. Будем вводить эти данные с клавиатуры и помещать соответственно в массивы  $m$  и  $a$ . Индексом массивов будет номер образца от 1 до 5.

Требуется вычислить и вывести в таблицу плотность вещества каждого образца. Чтобы не создавать еще один массив, будем производить вычисления непосредственно при выводе в процедуре `writeln()`.

Масса измеряется в граммах, а размер — в сантиметрах с точностью до одного десятичного знака

(вещественный тип). Таким образом, переменные могут быть описаны так:

```
Var m, a: array[1..5] of real;
    i: integer;
```

Алгоритм решения задачи основан на использовании формулы плотности вещества  $d = m/v = m/a^3$ .

Вычисление и вывод результатов в таблицу будут производиться в цикле for.

Программа имеет вид:

```
Program Primer10_3;
Var m, a: array[1..5] of real;
    i: integer;
Begin
    writeln('Попарно вводите массу (в г)
и ребро (в см) образцов');
    for i:=1 to 5 do readln(m[i], a[i]);
                                {Ввод m и a}
    writeln('Образец : Масса : Ребро :
Плотность');           {Заголовок таблицы}
    for i:=1 to 5 do
        writeln(i:4, m[i]:10:2, a[i]:9:2,
m[i]/(a[i]*a[i]*a[i]):10:2);
                                {Вывод}
End.
```

Протестируем программу. Результат может выглядеть так:

Попарно вводите массу (в г) и ребро (в см) образцов
67 2.3
36 1.6
85 2.2
45 4.1
52 1.8
Образец : Масса : Ребро : Плотность
1 67.00 2.30 5.51
2 36.00 1.60 8.79
3 85.00 2.20 7.98
4 45.00 4.10 0.65
5 52.00 1.80 8.92

Особый интерес представляет решение задач с использованием числовых массивов и графических объектов. Графические элементы улучшают наглядность и выразительность представляемой информации, а также могут иметь самостоятельное значение, например в программах с элементами геометрии, рисования, черчения.

**Пример 4.** Задан массив из 7 простых чисел. Составить программу, которая нарисует числовую ось и отметит на ней эти числа кружками.

Исходными данными являются простые числа, заданные в массиве:

```
const A: array[1..7] of integer =
  = (2, 3, 5, 7, 11, 13, 17);
```

Требуется нарисовать числовую ось и отметить на ней эти числа кружками.

Напомним, что начало отсчета находится в левом верхнем углу экрана, ось абсцисс направлена вправо, а ось ординат — вниз.

Пусть горизонтальная числовая ось проведена из точки с координатами (20, 200) и имеет 18 делений через 40 пикселей. Нетрудно вычислить координату ее конца  $20 + 18 \cdot 40 = 740$ . Деления представляют собой вертикальные отрезки длиной 20 пикселей ( $200 - 180$ ) с абсциссами  $20 + 40 \cdot i$ .

Центры кружков лежат на числовой оси (ордината 200) и имеют абсциссы  $20 + 40 \cdot A[i]$ . Пусть их радиус 8 пикселей, а цвет красный (clRed).

Программа может выглядеть так:

```
Program Primer10_4;
Uses GraphAbc;
  {Подключение графического модуля}
Const A: array[1..7] of integer = (2, 3, 5,
7, 11, 13, 17);
```

```
Var i: integer;  
Begin  
    line(20,200,740,200); {Рисование оси}  
    for i:=0 to 18 do {Рисование делений}  
        line(20+40*i,180, 20+40*i,200);  
    setBrushColor(clRed);  
        {Задание цвета кисти}  
    for i:=1 to 7 do circle(20+40*A[i], 200,8);  
        {Рисование кружков}  
End.
```

Результат работы будет выглядеть так:



**Пример 5.** В массив вводят результаты соревнований по прыжкам в высоту. Составить программу для отображения их в виде линейной диаграммы.

Исходными данными являются количество участников соревнований (не более 20) и результат каждого из них (высота прыжка в сантиметрах — целое число до 150). Будем вводить эти данные с клавиатуры и помещать соответственно в переменную n и массив h.

Линейная диаграмма представляет собой вертикальные отрезки, высота которых в пикселях равна высоте прыжков h[i] в сантиметрах. Нижние их концы лежат на горизонтальной оси

и имеют координаты  $(40*i, 200)$ , а верхние —  $(40*i, 200 - h[i])$ .

При работе с графическим окном системы PascalABC удобно все данные вводить и выводить в этом окне. Совмещать работу с текстом и графикой в одном окне можно, подключив модули CRT и GraphABC одновременно.

Программа может выглядеть так:

```
Program Primer10_5;
Uses CRT, GraphAbc;
Var h: array[1..20] of integer;
    n, i: integer;
Begin
    write('Введите количество участников
(от 5 до 20)');
    readln(n);
    writeln('Введите их результаты (до 150 см)');
    for i:=1 to n do read(h[i]);
    { Ввод результатов }
    line(20, 200, 740, 200);
    { Рисование горизонтальной оси }
    setPenColor(clRed);
    { Задание цвета пера }
    setPenWidth(6);
    { Задание толщины линии }
    for i:=1 to n do
        line(40*i, 200, 40*i, 200-h[i]);
    { Рисование вертикальных линий }
End.
```

Результат работы представлен на рисунке 2.11.

Несложно изменить эту программу так, чтобы результаты меньше, например, 100 см отображались

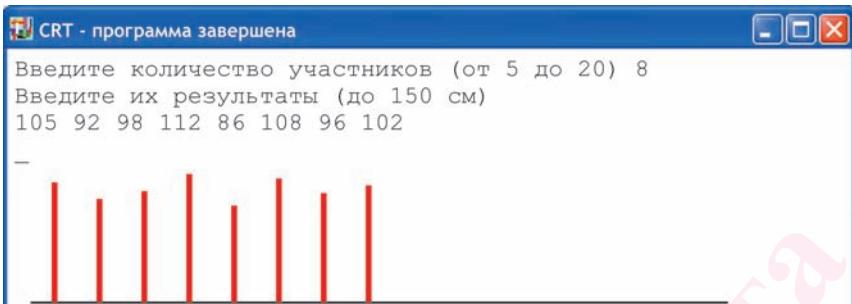


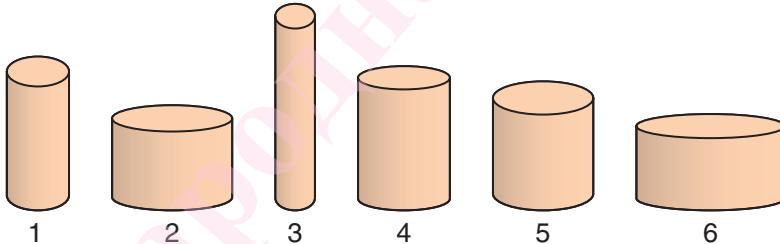
Рис. 2.11

синим цветом. Изменения заключаются в выборе цвета пера в цикле рисования линий по условию:

```
if h[i]<100 then setPenColor(clBlue)
else setPenColor(clRed);
```

### Упражнения

1. Сосуды цилиндрической формы заполнены до краев жидкостью плотностью  $d = 1200 \text{ кг}/\text{м}^3$ .



Составьте программу, которая вычисляет:

- давление столба жидкости на дно сосудов  $p = dgh$  ( $g = 9,8 \text{ м}/\text{с}^2$ );
- массу жидкости в каждом сосуде  $m = dV = dhS$  ( $S = 3,14 R^2$ );
- суммарную массу жидкости.

Требуемые исходные данные (радиус сосуда  $R$ , высота столба жидкости  $h$ ) вводятся в массив с клавиатуры, результат выводится на экран.

**2.** В лабораторной работе по определению теплоемкости вещества измеряют приращения температуры ( $\Delta T$ ) при сообщении одинакового количества теплоты  $Q = 420$  Дж шести образцам одинаковой массы  $m = 0,12$  кг, изготовленным из разных материалов. Составьте программу, которая вычисляет теплоемкость веществ  $c = \frac{Q}{m\Delta T}$ . Данные измерений  $\Delta T$  вводятся в массив с клавиатуры. Результаты измерений и вычислений выводятся в таблицу.

## Глава 3

### ОСНОВЫ АНИМАЦИИ

#### § 11. Основные понятия. Виды анимации

**Анимация** (от лат. *animare* — оживить) — имитация движения или изменения формы статических объектов. Анимацией называют также вид искусства, произведения которого создаются путем по-кадровой съемки отдельных рисунков или сцен. Помимо термина «анимация» широко употребляется также и термин «мультипликация» (от лат. *multiplicatio* — умножение, размножение).

**Кадры** — это нарисованные или сфотографированные изображения последовательных фаз движения объектов или их частей (рис. 3.1). При просмотре последовательности кадров возникает иллюзия оживления изображенных на них статичных персонажей. Для создания эффекта плавного изменения формы и положения объектов частота смены кадров, исходя из особенностей человеческого восприятия, должна быть не менее 12 — 16 кадров в секунду. В кино используется частота 24 кадра в секунду, в телевидении — 25 или 30 кадров в секунду.

Принцип анимации был найден задолго до изобретения кинематографа. Еще в начале XIX в. бельгийский физик Жозеф Плато и другие ученые и



Рис. 3.1



Рис. 3.2

изобретатели использовали для воспроизведения на экране движущихся изображений вращающийся диск или ленту с рисунками, систему зеркал и источник света — фонарь (рис. 3.2).

Рисованная анимация возникла в конце XIX в. В 1900—1907 гг. американец Джеймс Стюарт Блектон снял анимационные фильмы «Волшебные рисунки»,

«Комические выражения смешного лица», «Отель с привидениями». В России первые мультфильмы были созданы в 1911—1913 гг. В Беларуси первый мультфильм «Октябрь и буржуазный мир» создан в 1927 г.

Прорисовка всех фаз движения (кадров) в первых мультфильмах требовала огромных трудозатрат. Так, для мультфильма продолжительностью 5 мин при частоте 24 кадра в секунду необходимо 7200 рисунков. При этом многие кадры содержат повторяющиеся фрагменты, которые приходилось многократно перерисовывать. Поэтому с 20-х гг. XX в. начали применять упрощенную технологию анимации: на статичный, неизменный рисунок стали накладывать прозрачные целлулоидные пленки с изменяющимися подвижными элементами. Это был первый шаг в механизации труда художника-аниматора, который получил развитие в компьютерных технологиях.

В компьютерной анимации рисуются лишь некоторые опорные кадры (их называют **ключевыми**),

а промежуточные генерируются (рассчитываются) компьютерными программами. Независимая анимация отдельных элементов изображения обеспечивается созданием графических объектов для каждого персонажа и размещением их на разных слоях (подобно прозрачным пленкам в классической анимации).

Основные виды компьютерной анимации: **покадровая анимация** и **автоматическая (движения и формы)**.

Покадровая анимация (мультипликация) состоит в прорисовке всех фаз движения. Все кадры при этом являются ключевыми.

Автоматическая анимация заключается в рисовании ключевых кадров, соответствующих основным фазам или этапам движения, и последующем автозаполнении промежуточных кадров.

В основе любой анимации лежит фиксация фаз движения объектов — определение в каждый момент времени их положения, формы, размеров и иных свойств, например цвета. Эту операцию называют фазовкой или таймингом.

Чтобы уменьшить трудозатраты и избежать ошибок при работе на компьютере, полезно сначала наметить фазы на бумаге. При анимации движения неодушевленных объектов можно ограничиться указанием траектории движения и фиксацией объектов в наиболее важных положениях. При этом следует учитывать, что механическое движение любых объектов подчиняется законам физики.

**Пример 1.** Брошенный в горизонтальном направлении мяч трижды ударяется о пол. Зарисовать несколько фаз движения.

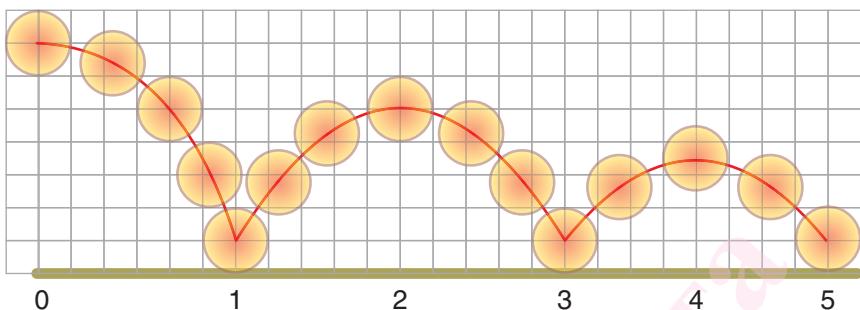


Рис. 3.3

Нарисуем траекторию движения центра мяча (рис. 3.3). Изобразим положения мяча в моменты удара о пол (1, 3, 5) и наибольшего подъема (2, 4). Добавим несколько промежуточных положений.

**Пример 2.** Брошенный под углом к горизонту молот вращается вокруг оси, проходящей через точку крепления рукоятки. Зарисовать несколько фаз движения.

Изобразим траекторию движения и несколько положений молота (рис. 3.4).

При определении фаз анимации одушевленных объектов необходимо учитывать не только физические законы, но и характеры персонажей и даже их эмоциональное состояние. Число фаз должно

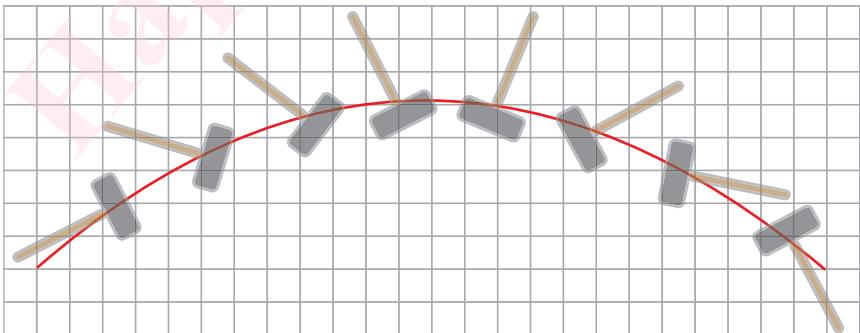


Рис. 3.4

обеспечивать плавность движения. Так, для шагающего человечка требуется не менее шести — восьми фаз.

**Пример 3.** Зарисовать фазы движения шагающего и бегущего человечка.

Изобразим фазы движения, как показано на рисунке 3.5.

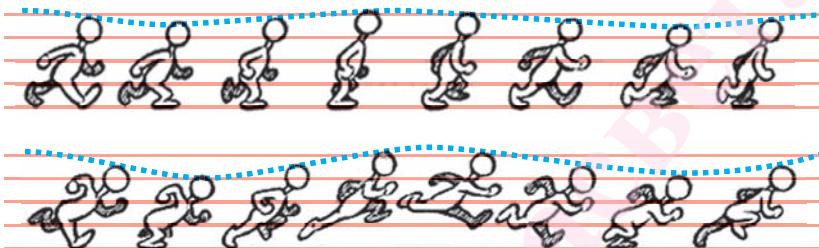


Рис. 3.5

- ? 1. В чем заключается покадровая анимация?  
2. Что представляет собой кадр в анимации?  
3. Какие кадры называют ключевыми?

### **Упражнение**

Зарисуйте основные фазы для анимации:

- а) воздушный шарик поднимается вверх и уменьшается;  
б) лодка качается на волнах;  
в) маятник совершает колебания;  
г) колесо скатывается по наклонной плоскости;  
д) из крана в стакан падают капли воды.

## **§ 12. Работа в редакторе Flash**

### **12.1. Основные элементы интерфейса**

Одной из универсальных систем векторной графики и компьютерной анимации является редактор Macromedia Flash. Кроме стандартных инструментов

тов рисования он содержит язык программирования Action Script, что позволяет создавать как простые анимации, так и интерактивные, т. е. управляемые пользователем.

После запуска редактора открывается окно, основные элементы которого представлены на рисунке 3.6.

В центре окна находится **Рабочее поле (Stage)**, на котором размещаются изображения. В верхней части окна находится строка меню, под которой располагается горизонтальная панель, содержащая инструменты управления **слоями (Layers)** и **шкалой времени (Timeline)**, обеспечивающую управле-

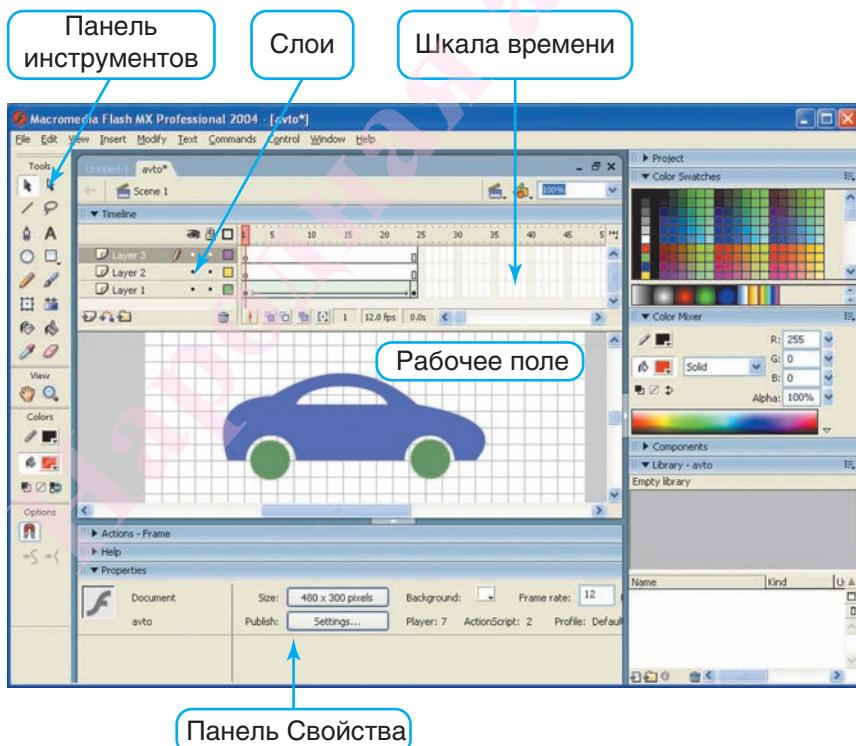


Рис. 3.6

ние кадрами (**Frames**). Вдоль левой стороны окна обычно располагают панель инструментов (**Tools**), в правой и нижней частях окна принято располагать так называемые плавающие панели, например Смеситель цветов (**Color Mixer**), Свойства (**Properties**), Библиотека (**Library**) и др. Чтобы освободить место для рисования и просмотра изображений, панели можно перемещать, удалять с экрана и возвращать по мере надобности. Для этого используют меню **Окно** (**Window**).

Панель инструментов рисования и редактирования содержит четыре раздела (рис. 3.7).

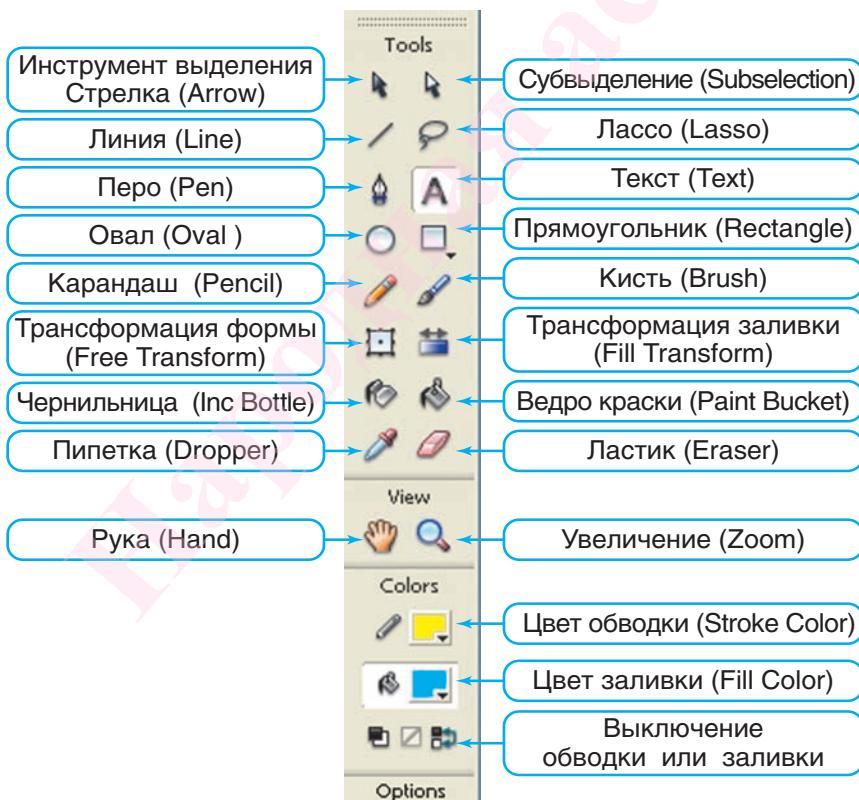


Рис. 3.7

В разделе **Tools** собраны сами инструменты.

Затем следуют разделы **View** (способ просмотра), **Color** (выбор цвета обводки и заливки) и, наконец, контекстно-зависимая область настройки инструментов **Options** (параметры).

## 12.2. Рисование в редакторе Flash

Любая анимация содержит последовательность статичных изображений, поэтому сначала надо научиться их создавать. Вы уже умеете рисовать в графическом редакторе Paint, а также работать с векторной графикой в текстовом редакторе Word. Некоторые инструменты Вам знакомы, например **Карандаш (Pencil)**, **Кисть (Brush)**, **Линия (Line)**, **Овал (Oval)**, **Прямоугольник (Rectangle)**, **Ластик (Eraser)**.

Рассмотрим, как создаются изображения в редакторе Flash.

В векторной графике все изображения и их фрагменты представляют собой **объекты**. Они компонуются из простейших объектов (графических примитивов), например линий (прямых или кривых), овалов, прямоугольников. Графический объект является совокупностью контура и внутренней области. Контур может быть обрамлен **обводкой (Stroke)** — линией, имеющей определенный цвет, толщину и другие особенности, т. е. **стиль**. Внутренняя область, ограниченная контуром, может иметь **заливку (Fill)**. Она также имеет свой стиль. Для задания цвета обводки и заливки можно использовать панель инструментов, а также панели **Набор цветов (Color Swatches)** и **Смеситель цветов (Color Mixer)**. Графический объект не обязательно

должен содержать одновременно и обводку, и заливку. Можно нарисовать фигуру без обводки или заливки, отключив их на панели инструментов. Можно также удалить обводку или заливку впоследствии.

Каждый объект можно трансформировать, т. е. преобразовывать, изменяя его свойства (например, форму, размер, положение, цвет, прозрачность).

С самого начала важно приобрести навыки выделения объектов и их частей, а также изменения их свойств. Основной инструмент выделения — Стрелка (Arrow). В версии Flash 8 этот инструмент называется Выделение (Selection).

**Пример 1.** Нарисовать закрашенный прямоугольник (рис. 3.8, *а*), выделить его и скопировать в буфер обмена. Вставить на рабочее поле четыре его копии и с помощью инструмента выделения Стрелка трансформировать их, как показано на рисунках 3.8, *в*, *г*, *д*, *е*.

Прежде всего, с помощью команд **Вид → Сетка** (**View → Grid**) или комбинации клавиш **Ctrl + Э** для удобства рисования на рабочем поле отобразим сетку. Она используется только при редактировании и не видна при просмотре.

Подберем цвет заливки и обводки. Возьмем инструмент **Прямоугольник** и нарисуем исходное изображение (см. рис 3.8, *а*).

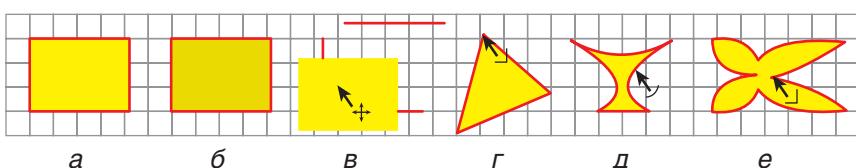


Рис. 3.8

Возьмем инструмент выделения **Стрелка**, при этом указатель мыши примет вид черной стрелки. Выделим весь нарисованный объект (т. е. и обводку и заливку) двойным щелчком мыши или заключив в прямоугольник при нажатой левой кнопке мыши (этот способ более универсален, поскольку позволяет выделять сразу несколько объектов). Индикатором выделения объектов служит появление на их изображении мелкой сеточки (рис. 3.8, б).

Комбинацией клавиш **Ctrl + С** скопируем выделенное изображение в буфер обмена. Затем с помощью клавиш **Ctrl + В** вставим на рабочее поле четыре копии изображения.

В первой копии подведем **Стрелку** к внутренней области фигуры (при этом ниже стрелки появится крестик) и перетащим вниз ее заливку, удерживая нажатой левую кнопку мыши. Теперь подведем стрелку к верхней стороне обводки, щелчком левой кнопкой мыши выделим ее и перетащим вправо. Наконец, выделим правую сторону и нажатием клавиши **Del** удалим ее (см. рис. 3.8, в).

В второй копии будем подводить **Стрелку** к вершинам прямоугольника (при этом у стрелки появляется уголок) и перетаскивать их. Перетащим две вершины прямоугольника до соединения — получим треугольник (см. рис. 3.8, г).

В третьей и четвертой копиях будем подводить **Стрелку** к сторонам прямоугольника (при этом у стрелки появляется дуга) и искривлять их, протаскивая с нажатой левой кнопкой мыши (см. рис. 3.8, д). Если делать это, удерживая нажатой клавишу **Ctrl**, то получаются новые вершины или изломы линий (см. рис. 3.8, е).

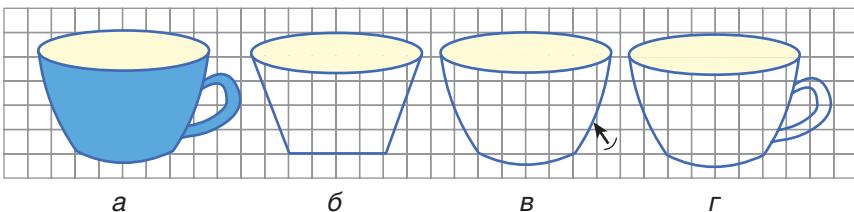


Рис. 3.9

**Пример 2.** Нарисовать голубую чашку (рис. 3.9, а).

Возьмем инструмент **Овал**, подберем цвет обводки и заливки и изобразим овал. Инструментом **Линия** нарисуем боковые стенки и дно чашки (рис. 3.9, б).

Инструментом выделения **Стрелка** придадим дну и стенкам требуемую кривизну (рис. 3.9, в).

Возьмем инструмент **Карандаш**, в области **Параметры (Options)** панели инструментов **Tools** установим параметр линии **Сглаженная (Smooth)** (рис. 3.10) и нарисуем ручку (рис. 3.9, г).

Осталось залить поверхности чашки и ручки с помощью инструмента **Ведро краски (Paint Bucket)**.

Параметры выделенного объекта отображаются и задаются на панели **Свойства (Properties)**. Ее вид и содержание **контекстно-зависимы**, т. е. соответствуют выделенному в данный момент объекту и выбранным инструментам. Так, после выбора инструмента рисования можно задать параметры обводки (**Stroke**) и заливки (**Fill**), настроить стиль линий (цвет, толщину, тип штриха и другие параметры). Тип линии выбирается

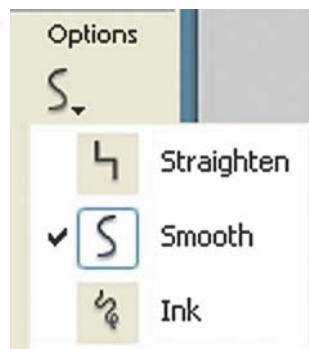


Рис. 3.10

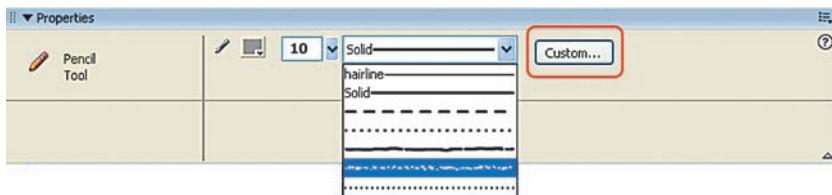


Рис. 3.11

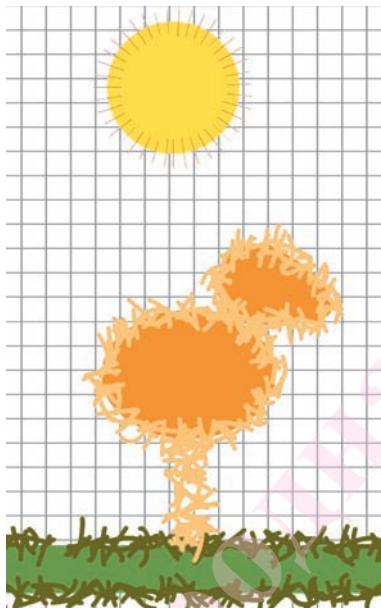


Рис. 3.12

из предлагаемых образцов: сплошная (**solid**), штриховая, пунктирная и т. д. (рис. 3.11) — и может настраиваться пользователем (кнопка **Custom**).

**Пример 3.** Создать изображение (рис. 3.12), используя инструменты **Прямоугольник** и **Овал** и подбирая настройки стиля обводки.

Для изображения солнца с лучами настроим стиль обводки (**Stroke Style**) с помощью кнопки **Custom**. Установим толщину (**Thickness**) обводки, равную длине ее лучей (10) (рис. 3.13, а). Нарисуем круг.

Для изображения травы и цыпленка настроим стиль обводки по-другому.

Для травинок зададим следующие параметры: длина (**Length**) — **Random** (случайная), повороты (**Rotate**) — **Free** (произвольные) и изгибы (**Curve**) — **Very Curved** (сильно искривленные) (рис. 3.13, б). Нарисуем зеленый прямоугольник.

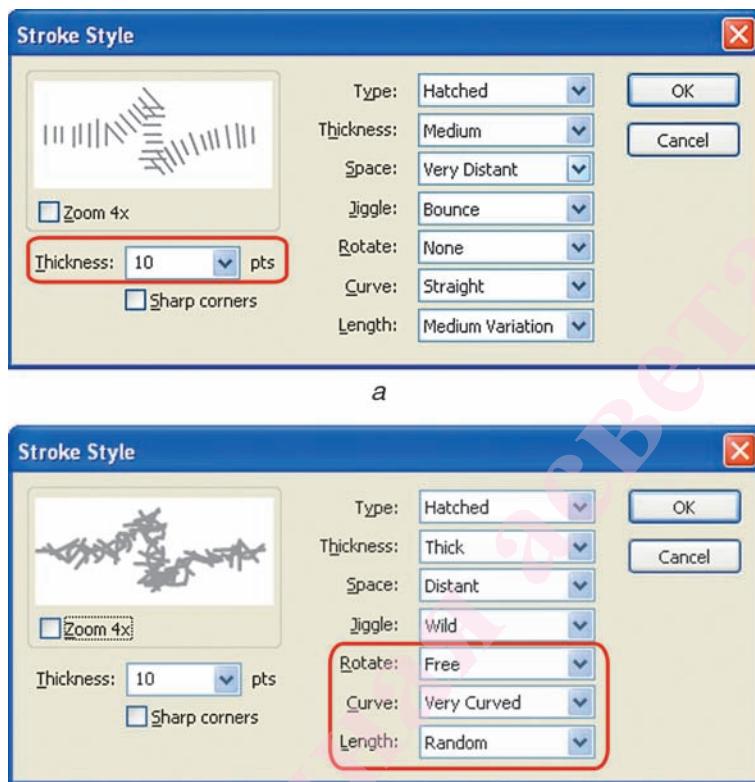


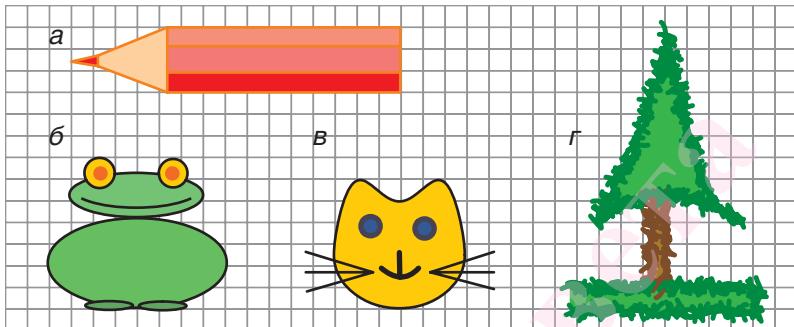
Рис. 3.13

Изменив только цвет, нарисуем два желтых овала и прямоугольник. С помощью инструмента **Стрелка** придадим им подобие цыпленка.

- ? 1. Перечислите основные инструменты рисования.  
 2. Для чего используют инструмент выделения **Стрелка**?  
 3. Как выделить обводку? Заливку?  
 4. Как настроить стиль обводки? Заливки?

## Упражнение

Создайте изображение по образцу:



## § 13. Преобразование объектов

Применение инструментов преобразования графических объектов исключительно полезно для анимации, поскольку позволяет сократить число новых рисунков. Анимируемые объекты можно получать в результате трансформации объектов из предыдущих кадров.

### 13.1. Преобразование формы

Графические объекты можно подвергать различным преобразованиям с помощью меню **Модифицировать** → **Трансформировать** (**Modify** → **Transform**) или инструмента **Трансформация** (**Free Transform**). Вид преобразования выбирается в меню или на панели инструментов в разделе **Параметры** (**Options**) (рис. 3.14).

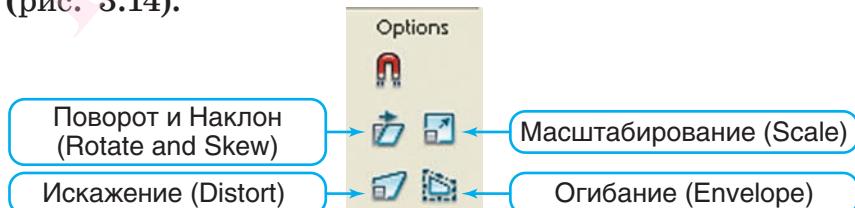


Рис. 3.14

**Пример 1.** Нарисовать фигуру, изображенную на рисунке 3.15, а, и трансформировать ее в фигуры, изображенные на рисунках 3.15, б, в, г.

С помощью инструментов **Прямоугольник**, **Овал** и **Стрелка** создадим исходное изображение (см. рис. 3.15, а). Выделим его, скопируем в буфер обмена, затем трижды вставим на рабочее поле.

Возьмем инструмент **Трансформация** (*Free Transform*). Вид указателя мыши при этом зависит от типа преобразования. Перемещая появившиеся маркеры мышью, будем трансформировать копии изображения. Для первой копии совершим преобразование **Поворот** (*Rotate*) (см. рис. 3.15, б), для второй — **Наклон** (*Skew*) (см. рис. 3.15, в), а для третьей — **Огибание** (*Envelope*) (см. рис. 3.15, г).

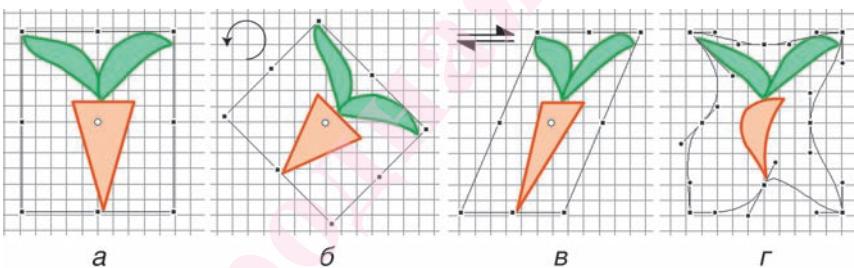


Рис. 3.15

**Пример 2.** Нарисовать елочку, размножить изображение и с помощью преобразования **Искажение** (*Distort*) создать эффект перспективы.

Нарисуем елочку и с помощью буфера обмена добавим две ее копии (рис. 3.16, а).

Выделим все елочки. Возьмем инструмент **Трансформация** и применим преобразование **Искажение** (*Distort*), достигая эффекта перспективы (рис. 3.16, б).

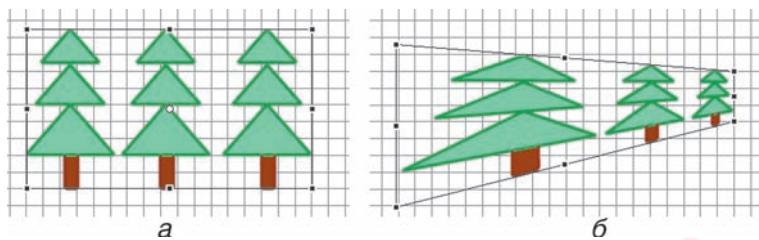


Рис. 3.16

Интересные эффекты наблюдаются при наложении фигур друг на друга. Верхняя фигура замещает собой перекрываемую ею область нижней фигуры. Если две фигуры имеют заливку одинакового цвета, то при удалении границы пересечения они объединяются в один объект. Инструменты **Карандаш** (**Pencil**) и **Линия** (**Line**) могут действовать как нож, когда создаваемые ими линии пересекают фигуры. Все это позволяет объединять объекты и вырезать их части.

**Пример 3.** Получить изображения, представленные на рисунке 3.17, путем объединения и вырезания фигур.

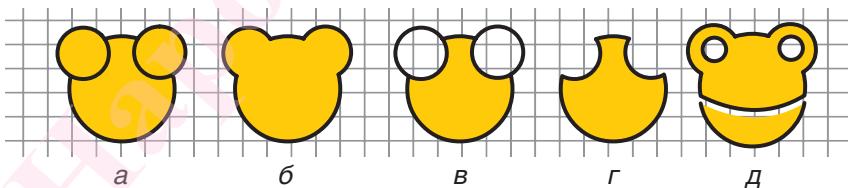


Рис. 3.17

Нарисуем круг, а на нем — два меньших круга такого же цвета (см. рис. 3.17, а).

Выделим все изображение (обводки и заливки), скопируем в буфер обмена, а затем вставим на рабочее поле четырех копии.

В первой копии выделим границы пересечения и клавишей Del удалим их. Получим объединенный объект (см. рис. 3.17, б).

В второй копии выделим и удалим заливку меньших кругов (см. рис. 3.17, в).

В третьей копии удалим заливку меньших кругов и внешние участки обводки (см. рис. 3.17, г).

На четвертой копии нарисуем еще два круга и удалим их заливку. Карандашом нарисуем дугу, пересекающую фигуру. Выделим заливку и обводку ниже дуги и сдвинем их вниз. Получим разрезанную на две части фигуру (см. рис. 3.17, д).

### 13.2. Преобразование заливки

Заливка может быть однотонной (solid), с плавным переходом цветов — градиентом, а также с заполнением растровым изображением (bitmap). Цветовые палитры и шаблоны градиентов представлены на панели **Набор цветов (Color Swatches)**. Для настройки параметров заливки используют панель **Смеситель цветов (Color Mixer)**.

**Пример 4.** Изобразить прямоугольник с заливкой спектром (красный, оранжевый, желтый, зеленый, голубой, синий, фиолетовый) с линейным градиентом.

На панели **Набор цветов (Color Swatches)** выберем подходящий шаблон заливки (рис. 3.18).

На панели **Смеситель цветов (Color Mixer)** выберем линейный градиент (**Linear**) и настроим набор цветов.

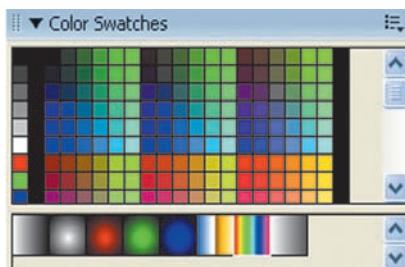


Рис. 3.18

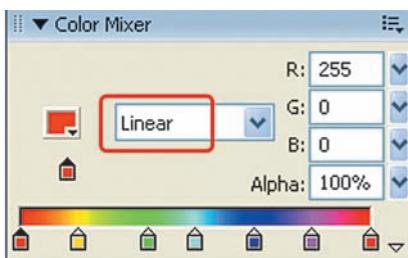


Рис. 3.19

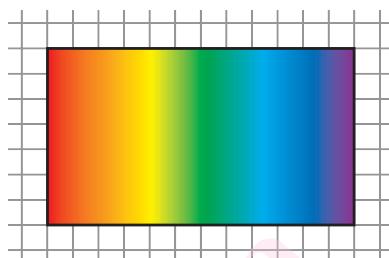


Рис. 3.20

С этой целью, щелкая мышью по полоске цветов (спектру), будем устанавливать маркеры и задавать для каждого из них требуемый цвет (рис. 3.19).

Нарисуем прямоугольник с заливкой спектром (рис. 3.20).

Для преобразования градиентной заливки служит инструмент **Трансформация заливки** (*Fill Transform*).

**Пример 5.** Нарисовать овалы с радиальной и линейной градиентными заливками (рис. 3.21, а, в). Трансформировать заливки в соответствии с рисунками 3.21, б, г.

Нарисуем овал с радиальной градиентной заливкой (см. рис. 3.21, а).

Выберем инструмент **Трансформация заливки** (*Fill Transform*) и щелкнем мышью по фигуре. Вокруг появится граница с маркерами редактирования. Каждый маркер имеет свое назначение. Переместив центральный маркер, изменим положение светового блика, а с помощью двух других маркеров вытянем и повернем блик (рис. 3.21, б).

Нарисуем овал с линейной градиентной заливкой (рис. 3.21, в). Переместив маркеры, изменим параметры заливки в соответствии с рисунком 3.21, г.

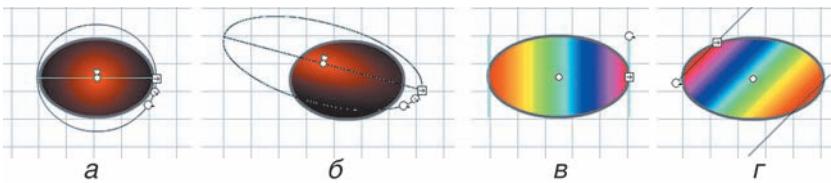


Рис. 3.21

Для копирования стилей обводки и заливки одного объекта на другой предназначен инструмент **Пипетка (Dropper)**.

**Пример 6.** Нарисовать четыре розовых квадрата с красной обводкой, как на рисунке 3.22, *а*, и круг с заливкой радиальным градиентом и «мохнатой» обводкой (рис. 3.22, *д*). Скопировать стили обводки и заливки круга на квадраты.

Нарисуем заданные фигуры.

Возьмем инструмент **Пипетка** и щелкнем по обводке круга. Активизируется инструмент **Чернильница**. Щелкая мышью на квадратах, придадим им стиль обводки круга (рис. 3.22, *б*, *в*, *г*).

Снова возьмем инструмент **Пипетка** и щелкнем на внутренней области (заливке) круга. Активизируется инструмент **Ведро краски**. Щелкая мышью на квадратах, придадим им стиль заливки круга. Если при этом выключена **Блокировка заливки (Lock Fill)** (изображение замка в секции **Options** панели инструментов), то каждый объект начинает

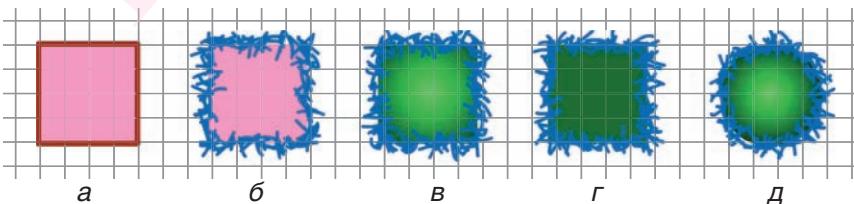


Рис. 3.22

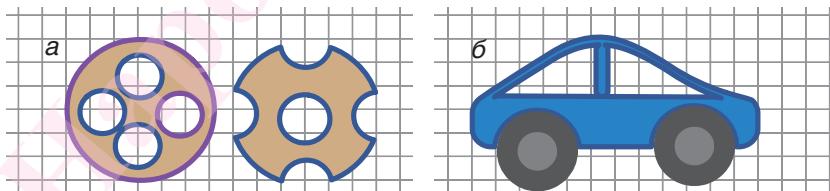
заливаться подобно исходному (сравните рис. 3.22, *в* и *д*). При включенной блокировке заливка каждого объекта является продолжением предыдущей (в нашем примере цвет исходного круга изменяется от светлого до темно-зеленого, последним цветом и заливает квадрат на рисунке 3.22, *г*).

Фигуры можно заливать не только сплошным цветом или градиентом, но и растровым изображением из файла. Эту заливку также можно подвергать трансформациям: изменять масштаб и поворачивать изображения.

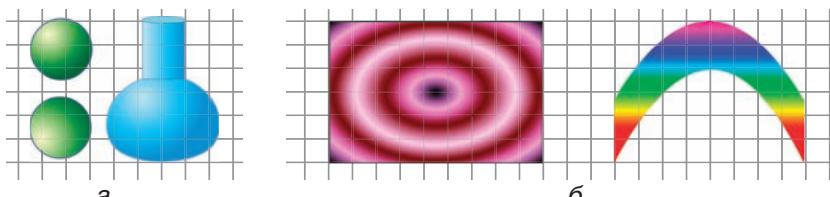
- ? 1.** Какие преобразования можно совершить инструментом **Трансформация**?
- 2.** Какие преобразования можно совершить инструментом **Трансформация заливки**?
- 3.** С какой целью используют инструмент **Пипетка**?

### Упражнения

**1.** Используя инструменты **Овал** и **Прямоугольник**, а также инструменты выделения и преобразования, нарисуйте и трансформируйте фигуру в соответствии с рисунком:



**2.** Создайте фигуры с заливкой по образцу:



## § 14. Использование слоев и библиотеки объектов

### 14.1. Создание и использование слоев

При создании композиций из нескольких изображений используются **слои (layers)**. Слои можно воспринимать как прозрачные пленки (кальки) с изображениями. Слои — важнейший элемент анимации. Они обеспечивают независимые движения и преобразования нескольких объектов. Кроме того, слои используются для размещения траекторий движения, звуков, видеофрагментов, а также программ на языке Action Script.

Список слоев (рис. 3.23) находится в левой части шкалы времени. При создании нового фильма в этом списке всего один слой с именем **Layer 1 (Слой 1)**. Рекомендуется сразу же изменить это имя на более соответствующее содержанию слоя. Для этого достаточно дважды щелкнуть левой кнопкой мыши на предлагаемом имени слоя и ввести новое. Слои можно помещать в папки. Папки позволяют группировать слои со сходными объектами.

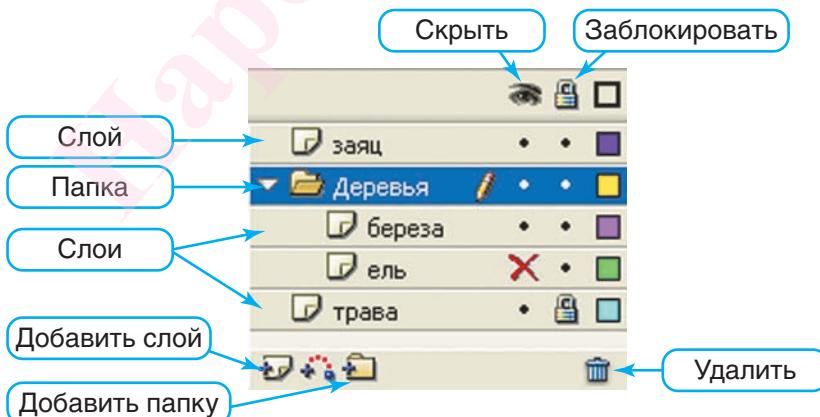


Рис. 3.23

Слой, содержимое которого можно редактировать в данный момент, является **активным**. Его имя выделяется в списке темным цветом и отмечается изображением карандаша. Слой можно **заблокировать**, т. е. запретить редактирование расположенных на нем объектов. Это полезно делать, чтобы нечаянно не поместить на «чужой» слой вновь создаваемые объекты, а также чтобы не исказить и не переместить уже существующие. Чтобы заблокировать слой, достаточно щелкнуть мышью по точке, расположенной в столбце под изображением замка.

Слой со всеми объектами на нем можно **скрыть** (сделать невидимым). Для этого достаточно щелкнуть мышью по точке под изображением глаза. Второй щелчок вернет видимость. Если щелкнуть на изображении глаза над списком слоев, то все они станут невидимыми. Для удаления слоя необходимо выделить в списке его имя и щелкнуть на кнопке с изображением корзины под списком.

**Пример 1.** Изобразить траву, березу, ель и зайца на разных слоях. Поместить слои «береза» и «ель» в папку «Деревья». Исследовать влияние порядка следования и видимости слоев на изображение.

На единственном имеющемся слое нарисуем зеленый прямоугольник, настроив стиль обводки. Назовем слой «трава». Заблокируем его.

Добавим еще три слоя. Для этого трижды щелкнем мышью на кнопке **Добавить слой (Insert Layer)**. Изменим предлагаемые имена слоев на имена: «береза», «ель», «заяц».

Нарисуем деревья и зайца на соответствующих слоях (рис. 3.24, а).

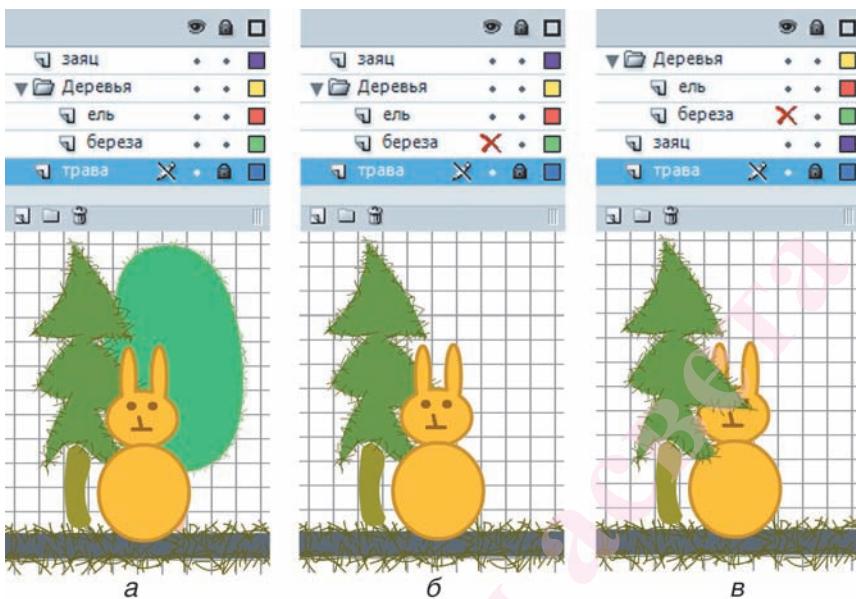


Рис. 3.24

Для создания папки щелкнем на кнопке **Добавить папку** (*Insert Folder*). Изменим предлагаемое имя **Folder 1** (**Папка 1**) на «Деревья».

Перетащим в эту папку слои «береза» и «ель».

Скроем слой «береза» (рис. 3.24, *б*).

Порядок слоев в списке задает порядок расположения объектов: слой, чье имя находится выше, располагается над слоем (ближе к зрителю), имя которого стоит ниже. Так, ель находится перед березой, а заяц перед елью, если слои расположены в соответствии с рисунком 3.24, *а*. Для изменения порядка слоев достаточно перетащить их мышью на требуемый уровень списка. Перетащим слой «заяц» ниже слоев деревьев — заяц окажется за елью (рис. 3.24, *в*).

## 14.2. Создание и использование библиотечных образцов

Для хранения объектов многократного использования предназначена **библиотека (Library)**. Объект из библиотеки можно размещать на рабочем поле в нескольких экземплярах, подвергая каждый из них нужным трансформациям.

Чтобы поместить объект в библиотеку, необходимо преобразовать его в **библиотечный образец**, называемый в терминологии Flash **символом (symbol)**. Возможны три типа символов: **MovieClip** (фрагмент фильма или **Клип**), **Graphic** (Графика) и **Button** (Кнопка). Клип может содержать как статичное изображение, так и анимацию, звук и видеофрагменты. Он может управляться программой на языке Action Script. Тип Графика также может содержать анимацию и аудио-, видеофрагменты, однако им нельзя управлять программно. Кнопка используется для интерактивного управления анимацией.

**Пример 2.** Нарисовать зеленое яблоко и превратить его в библиотечный образец типа Графика. Поместить на рабочее поле три экземпляра яблока из библиотеки, трансформировать их в соответствии с рисунком 3.25.

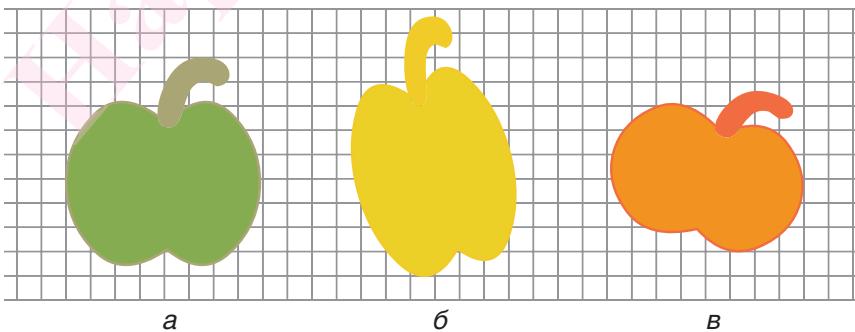


Рис. 3.25

Возьмем инструмент **Овал** и нарисуем зеленый круг. С помощью инструментов выделения придадим ему форму яблока. Инструментом **Кисть (Brush)** додадим веточку (см. рис. 3.25, а).

Для превращения объекта в библиотечный образец выделим его и нажмем клавишу F8 или выполним команду меню **Вставка → Преобразовать в символ (Insert → Convert to Symbol)**. В появившемся диалоговом окне введем имя «яблоко», выберем тип **Графика (Graphic)** и нажмем кнопку **OK** (рис. 3.26).

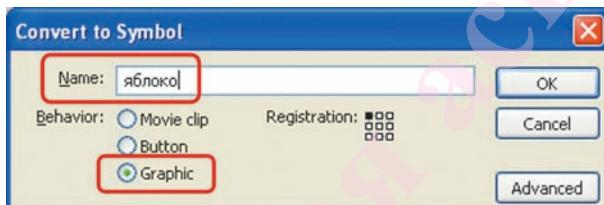


Рис. 3.26

Просмотрим содержимое библиотеки, открыв ее окно с помощью меню **Окно → Библиотека (Window → Library)** или комбинации клавиш Ctrl + L. В ней находится один только что созданный символ «яблоко».

В верхней части окна библиотеки расположена область просмотра символов, а в нижней — список со значками, обозначающими тип символа (рис. 3.27).

Символы можно размещать в папках. На нижней рамке окна библиотеки расположены кнопки создания и удаления символов и папок. Ширина окна библиотеки, а также сортировка символов в списке регулируются кнопками на правой рамке окна.

Поместим на рабочем поле три экземпляра яблока. С этой целью захватим мышью либо название

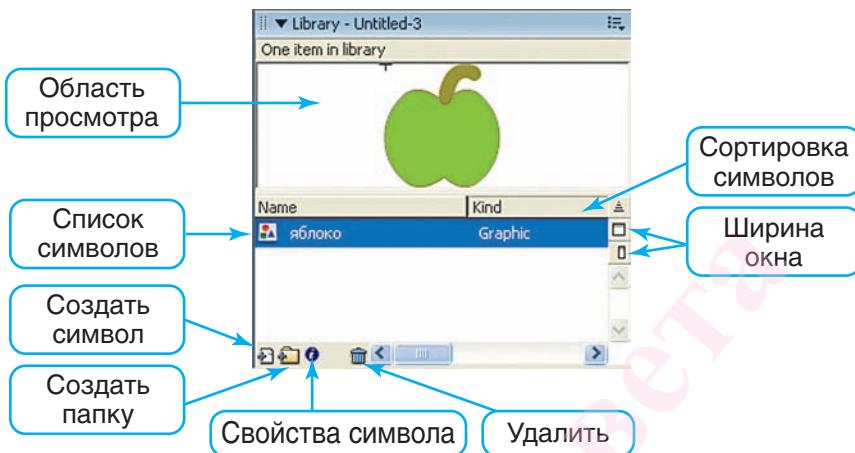


Рис. 3.27

символа «яблоко» либо изображение в области просмотра и перетащим его на рабочее поле. Повторим эту операцию дважды.

Оказавшийся на рабочем поле объект является экземпляром (*instance*) символа (прототипа объекта). Фактически экземпляр представляет собой лишь ссылку на библиотечный образец. Параметры экземпляра отображаются на панели **Свойства** (*Properties*) и могут быть изменены с помощью меню, инструментов трансформации и панели **Свойства**.

Инструментом **Трансформация** изменим форму, размеры и положение экземпляров в соответствии с рисунками 3.25, б, в. Для изменения цвета используем панель **Свойства**, на которой подберем нужный **Оттенок** (*Tint*) (рис. 3.28).



Рис. 3.28

**Пример 3.** Превратить яблоки из примера 2 в груши.

Для превращения сразу всех яблок достаточно изменить их библиотечный образец. С этой целью перейдем на его рабочее поле, дважды щелкнув мышью по изображению в области просмотра библиотеки либо по значку в списке символов. С помощью инструментов выделения трансформируем яблоко в грушу.

Вернемся на основную сцену, щелкнув по надписи **Сцена 1 (Scene 1)** на рамке шкалы времени. Все экземпляры символа «яблоко» превратились в груши (рис. 3.29).



Рис. 3.29

Если в библиотечном образце вырезать отверстие, то все его экземпляры унаследуют это изменение (рис. 3.30).

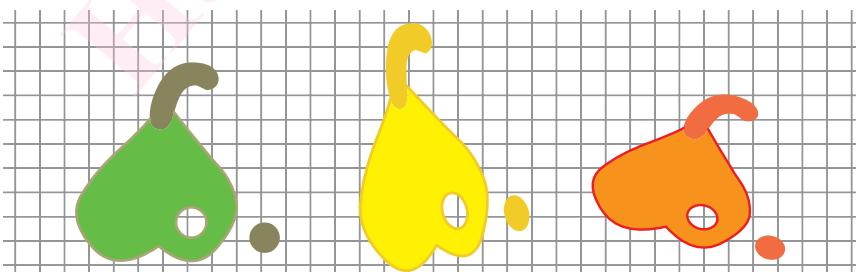


Рис. 3.30

Таким образом, преобразования экземпляра (изменения расположения, размера или цвета) не отражаются на библиотечном образце (родительском символе). В то же время изменения библиотечного образца наследуются всеми его экземплярами.

В документ Flash можно импортировать изображения, аудио- и видеофрагменты из файлов других графических и мультимедийных форматов, например .bmp, .jpeg, .gif, .wav, .mp3, .avi, .mpeg. Импортированное изображение сразу же попадает в библиотеку. Оно используется в документе Flash как единый объект, который можно подвергать некоторым трансформациям: масштабировать, перемещать, вращать и наклонять.

Импортированное изображение можно преобразовать в символ. Экземпляры символов допускают больше преобразований (например, изменение цвета, прозрачности).

- !** Если созданное или импортированное изображение предназначено для размножения или анимации движения, его необходимо преобразовать в символ.



Рис. 3.31

**Пример 4.** Нарисовать шар. Импортировать изображение снежинки. Превратить их в библиотечные образцы. Нарисовать елку и развесить на ней разноцветные шары и снежинки (рис. 3.31).

На единственном имеющемся слое нарисуем елку, настроив стиль обводки. Назовем слой «елка» и заблоки-

руем его во избежание случайных изменений.

Добавим еще один слой. Назовем его «игрушки».

На слое «игрушки» нарисуем шар, выбрав заливку радиальным градиентом. Выделим его и нажатием клавиши F8 преобразуем в символ типа **Графика (Graphic)**, присвоив имя «шар». Он попадет в библиотеку.

На этот же слой с помощью команд **Файл → → Импорт (File → Import)** импортируем изображение снежинки из файла *Flake.gif*. Оно попадет на рабочее поле и в библиотеку. Нажатием клавиши F8 преобразуем его в символ типа **Графика**, присвоив имя «снежинка». В библиотеке теперь и исходное растровое изображение, и символ (рис. 3.32).

Развесим на елке шары и снежинки. Для этого будем перетаскивать экземпляры объектов «шар» и «снежинка» из библиотеки на слой «игрушки», изменяя при этом их цвет и размер в соответствии с рисунком 3.31.

- ?
- 1. С какой целью используются слои и папки слоев?
- 2. Каково назначение библиотеки объектов?
- 3. Что называют символом?
- 4. Какие операции необходимо произвести для преобразования изображения в символ?

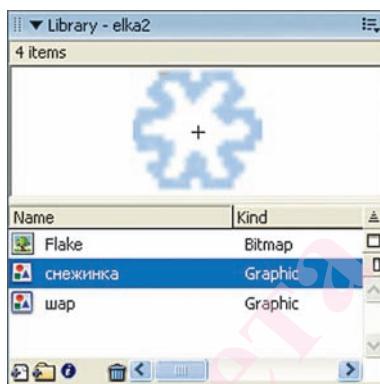


Рис. 3.32

### *Упражнение*

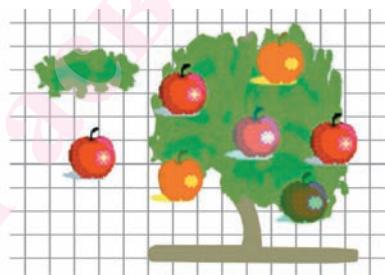
Создайте композицию по образцу:

а) импортируйте изображения: звездного неба *sky.gif*, Земли *zem.gif* и Луны *luna.gif*. Преобразуйте их в символы. Разместите экземпляры на разных слоях;

б) на слое «дерево» изобразите ствол. Создайте символ «ветка», импортируйте изображение яблока *apple.gif*. Разместите экземпляры веток на слое «дерево», а яблок — на слое «яблоки».



а



б

## **§ 15. Основы Flash-анимации**

### **15.1. Основные понятия**

На предыдущих уроках мы рассматривали Flash как векторный графический редактор. Однако его главное назначение состоит в создании анимаций, которые могут содержать звуковое сопровождение, видеофрагменты и интерактивные элементы. В настоящее время Flash-анимация превратилась в мощную технологию, особенностями которой являются: векторная графика; реализация нескольких видов анимации; импорт и экспорт векторных и растровых изображений, аудио- и видеофрагментов; поддержка интерактивных элементов интерфейса; встроенный язык программирования. С помощью этой техноло-

гии создаются рекламные ролики, фильмы, мультимедийные приложения, модели, интерактивные web-страницы и даже игры. Благодаря исключительно малому размеру они широко распространены в сети Интернет.

Процесс разработки анимации в среде Flash состоит в следующем. Сначала создается исходный документ (файл с расширением **.fla**). Этот документ можно просматривать и редактировать в редакторе Flash. Затем он преобразуется в **swf**-файл, который уже можно просмотреть во Flash-проигрывателе и web-браузере. Этот процесс называют **публикацией** фильма. Кроме того, можно экспорттировать результаты работы в файлы других форматов, например **.avi**, **.mov**, анимированный **.gif**.

Основной инструмент при работе с анимацией — **шкала времени**. На ней отображается информация о слоях и кадрах: типы кадров, их содержимое, наличие звуков и действий (рис. 3.33). Ячейки сетки на шкале времени соответствуют кадрам. Выделенный красным цветом **указатель** (или **маркер кадра**) указывает на кадр, содержимое которого отображается в рабочей области.

Пустые кадры не окрашены. Ключевые кадры обозначаются кружками (заполненные — черными,

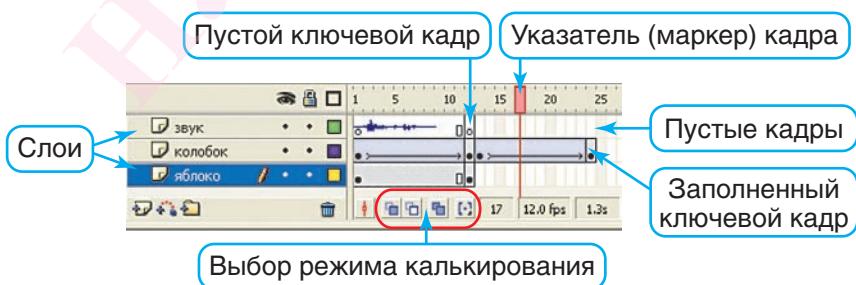


Рис. 3.33

а незаполненные — белыми). Неизменяемые промежуточные кадры, которые дублируют ключевые, окрашены серым цветом. Сиреневая или светло-зеленая окраска шкал времени говорит о том, что кадры сгенерированы автоматически.

Любая анимация состоит из последовательности кадров (**frames**). Кадры создаются вручную или генерируются автоматически. Важнейшее понятие анимации — **ключевые кадры (keyframes)** — это кадры, которые не изменяются при просмотре фильма, а используются в качестве опорных для генерации промежуточных кадров. При покадровой анимации все кадры являются ключевыми и создаются вручную.

Различают два вида автоматической анимации, которым соответствуют два способа генерации промежуточных кадров:

- изменение формы объектов (**shape tween**);
- изменение положения и трансформация экземпляров символов (**motion tween**).

## 15.2. Покадровая анимация

**Покадровая анимация (мультипликация)** реализуется последовательностью **ключевых кадров**, каждый из которых содержит новое или измененное вручную изображение (фазу анимации).

**Пример 1.** Создать мультипликацию, состоящую из четырех кадров (рис. 3.34).

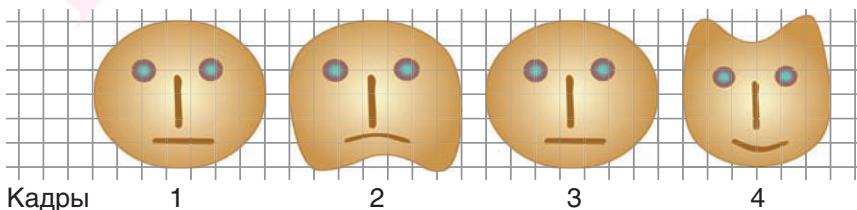


Рис. 3.34

Находясь в кадре 1 единственного слоя, возьмем инструмент **Овал**. На панели **Набор цветов** подберем радиальную градиентную заливку и нарисуем исходное изображение. Кистью дорисуем детали (см. рис. 3.34, кадр 1). Первый ключевой кадр готов.

Продублируем его содержимое еще на три ключевых кадра. Для этого трехкратным нажатием клавиши F6 или командой **Вставка → Ключевой кадр (Insert → Keyframe)** вставим три ключевых кадра с заполнением. Перемещая мышью указатель кадра по шкале времени (рис. 3.35), убедимся, что во всех четырех кадрах одинаковый рисунок.

Дублирование кадров можно выполнить и так: выделить все четыре кадра при нажатой клавише Shift и превратить их в ключевые с заполнением нажатием клавиши F6.

В кадрах 2 и 4 с помощью инструментов выделения трансформируем изображения в соответствии с рисунком 3.34.

Просмотрим последовательность кадров в статике (перемещая мышью указатель кадров по шкале времени) и в динамике (нажатием клавиши Enter). Для тестирования созданной анимации удобен режим многократной демонстрации, который вызывается с помощью меню **Control → Test Movie** или комбинации клавиш Ctrl + Enter.

Сохраним результаты работы под именем **mult.fla** (исходный Flash-документ).

Опубликуем созданную мультипликацию в формате **.swf**, т. е. сохраним Flash-фильм под именем **mult.swf**. Публикация фильма осуществляется с по-

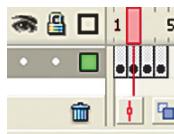


Рис. 3.35

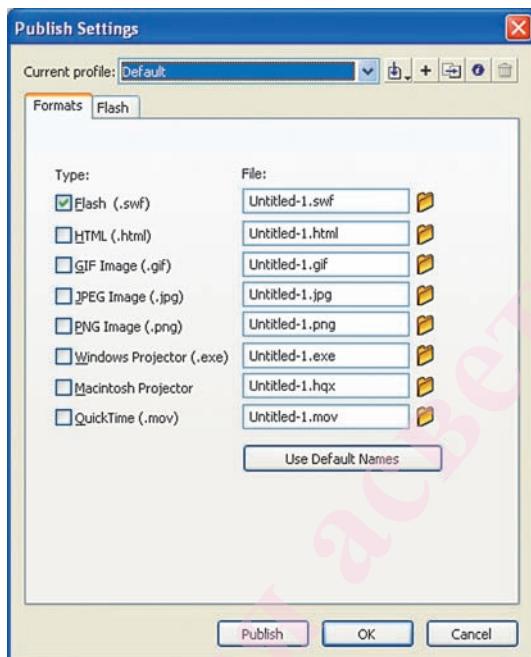


Рис. 3.36

мощью меню **Файл** → **Опубликовать** (File → Publish). Форматы и параметры файлов устанавливаются в окне **Установки публикации** (Publish Settings) (рис. 3.36).

**!** Если исходный Flash-документ уже сохранен в формате .fla, то тестирование приводит к его автоматическому сохранению в формате .swf в той же папке.

**Пример 2.** Создать мультиплексию из трех кадров: шары на елке переливаются разными цветами (рис. 3.37).

Находясь в кадре 1 единственного слоя, нарисуем елку, подобрав подходящий стиль обводки. Назовем слой «елка». Заблокируем его.

Добавим еще один слой, назвав его «шары». Нарисуем шар с заливкой радиальным градиентом.

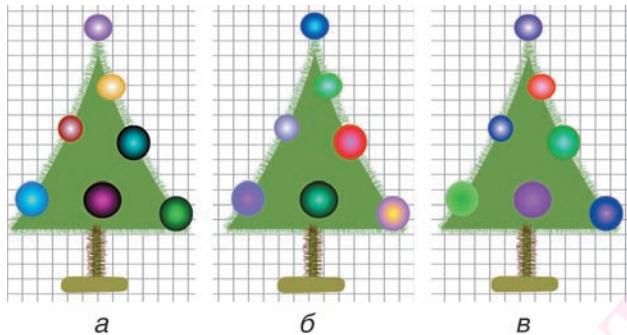


Рис. 3.37

Преобразуем в символ типа Графика (Graphic), присвоив имя «шар».

Развесим шары на елке, перетаскивая экземпляры символа на слой «шары» и устанавливая их цвет и размер в соответствии с рисунком 3.37, а.

Продублируем содержимое обоих слоев первого кадра, вставив два ключевых кадра с заполнением (рис. 3.38).

В кадрах 2 и 3 с помощью панели Свойства изменим цвет шаров (рис. 3.37, б, в).



Рис. 3.38

Просмотрим последовательность кадров в статике.

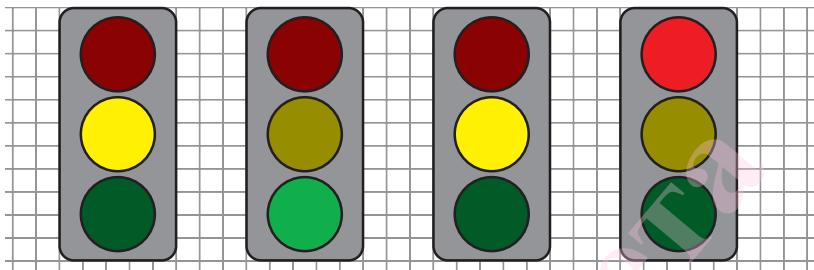
Сохраним результаты работы под именем elka.fla. Протестируем анимацию.

- ?
- 1. Какую анимацию называют покадровой?
- 2. Какие кадры называют ключевыми?
- 3. Для чего дублируют ключевые кадры?

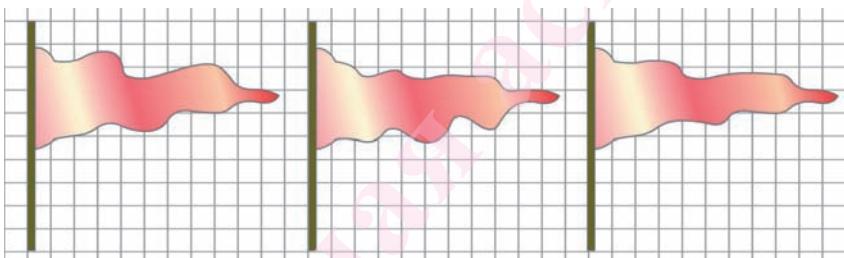
### *Упражнение*

Создайте покадровую анимацию (мультипликацию) по образцу:

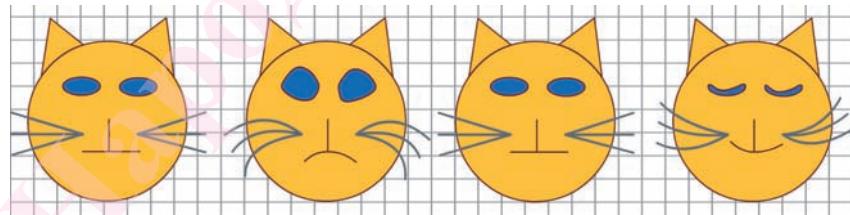
а) светофор (4 кадра: желтый, зеленый, желтый, красный свет);



б) развевающийся вымпел (один слой, 3 кадра, заливка линейным градиентом);



в) моргающий кот (4 кадра, усы расположены на отдельном слое).



## § 16. Анимация движения

**Анимация движения** производится путем автоматической генерации промежуточных кадров (автозаполнения). При этом ключевыми являются лишь первый и последний кадры, которые содержат начальную и конечную фазы анимации. Промежу-

точные кадры создаются путем трансформации экземпляров символов и изменения их положения. Генерацию промежуточных фаз обеспечивает инструмент **Create Motion Tween** (создание движения путем изменения промежуточных кадров).

**Пример 1.** Создать анимацию движения шара на фоне деревьев (рис. 3.39).

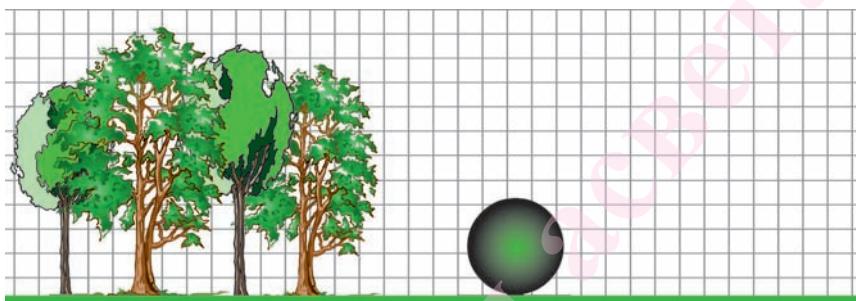


Рис. 3.39

Импортируем рисунок `trees.gif`. Он попадет на единственный имеющийся слой (**Layer 1**) и в библиотеку. Поскольку этот рисунок служит неподвижным фоном, не будем превращать его в символ. Откорректируем его положение и размеры инструментом **Трансформация**. Слой, на котором он находится, назовем «лес». Во избежание случайных изменений заблокируем его.

Добавим еще один слой и назовем его «шар» (рис. 3.40).

В кадре 1 этого слоя, нарисуем шар, выбрав заливку радиальным градиентом. Преобразуем его в символ типа **Графика (Graphic)**, присвоив имя «шар». Созданный символ попадет в библио-

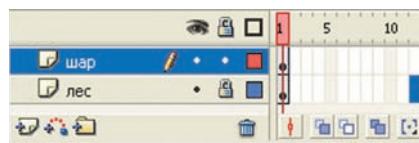


Рис. 3.40

теку (вызывается клавишами **Ctrl + L**), а на слое «шар» теперь находится его экземпляр.

Поместим экземпляр символа «шар» слева перед деревьями.

Выделим на шкале времени 12-й кадр в обоих слоях и преобразуем в ключевой, нажав клавишу **F6**. Все кадры заполняются содержанием, дублирующим ключевые кадры своего слоя (на это укажет серая окраска этих кадров на шкале времени).

На самом деле изображения находятся только в ключевых кадрах 1 и 12, а промежуточные кадры 2—11 содержат ссылки на кадр 1. Поэтому Flash-фильм (файл **.swf**) имеет очень маленький размер.

Кадры слоя «лес» оставим без изменения, поскольку фон неподвижен.

В 12-м кадре слоя «шар» переместим шар в конечное положение, уменьшив его размеры с помощью инструмента трансформации.

Произведем автозаполнение промежуточных кадров. Для этого выделим первый или любой промежуточный кадр этого слоя (например, кадр 7). С помощью меню **Вставка → Создать анимацию движения** (**Insert → Create Motion Tween**), правой кнопки мыши или панели **Свойства** сгенерируем промежуточные кадры. Сиреневая окраска и стрелка от 1-го кадра к 12-му указывает на то, что промежуточные кадры сгенерированы (рис. 3.41). (В случае ошибки стрелка рисуется пунктиром.)



Рис. 3.41

Просмотрим последовательность фаз движения, перемещая указатель кадров вдоль шкалы времени.

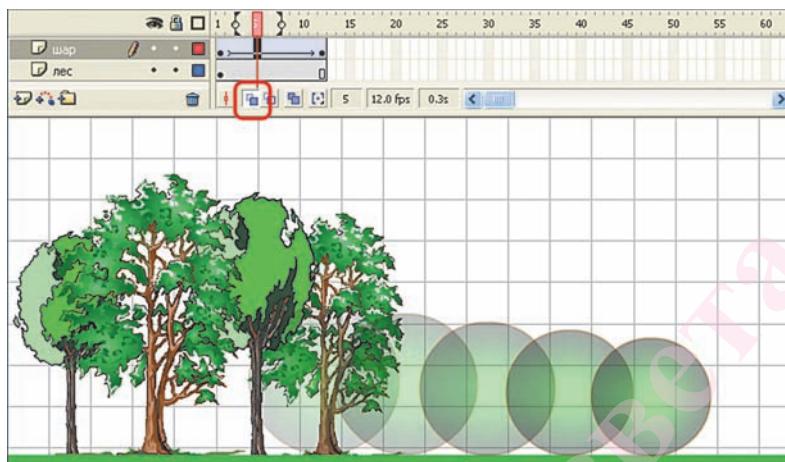


Рис. 3.42

Одновременно несколько фаз анимации удобно наблюдать в режиме **калькирования** (Onion Skin), который включается кнопками на нижней рамке шкалы времени. В этом режиме отображаются контуры объектов в соседних кадрах или шлейф движущегося изображения (рис. 3.42).

Сохраним работу под именем **dvi.fla**. Протестируем анимацию. При этом она будет сохранена в той же папке под именем **dvi.swf**.

**!** Анимация движения применима только для экземпляров библиотечных образцов (символов) или сгруппированных объектов.

Плавность анимации повышается с увеличением числа фаз движения и частоты смены кадров. По умолчанию установлена частота 12 кадров в секунду (12 fps), поэтому длительность созданной анимации 1 с.

Необходимое число кадров вычисляется умножением частоты кадров  $f$  на длительность анимации  $t$  в секундах:  $N = f \cdot t$ .

Промежуточные кадры вставляются с помощью меню **Вставка** → **Кадр** (**Insert** → **Frame**) или клавиши F5, а удаляются с помощью меню **Вставка** → → **Удалить кадры** (**Insert** → **Remove Frames**) или комбинации клавиш Shift + F5.

**Пример 2.** Увеличить длительность созданной в примере 1 анимации до 3 с. Исследовать влияние количества и частоты кадров на длительность и плавность анимации.

Увеличим число кадров до  $N = 12 \cdot 3 = 36$ , вставив промежуточные кадры клавишей F5 или протаскиванием выделенных концевых ключевых кадров с нажатой левой кнопкой мыши. Протестируем анимацию. Ее длительность 3 с.

На панели **Свойства** увеличим частоту до 30 кадров в секунду. Протестируем анимацию. Ее длительность стала  $36 : 30 = 1,2$  с.

Восстановим частоту 12 кадров в секунду.

Анимация движения, состоящего из нескольких этапов, реализуется заданием ключевых кадров в моменты начала и окончания этих этапов и коррекцией свойств объектов (их положения, размеров, цвета, прозрачности) в этих кадрах.

Изменение скорости движения задается на панели **Свойства** параметром **Замедление** (**Ease**). По умолчанию движение равномерное (параметр = 0), при установке положительного значения движение замедленное, отрицательного — ускоренное.

**Пример 3.** Отредактировать созданную анимацию так, чтобы 1,5 с шар двигался с замедлением вправо и уменьшался, останавливался, а затем 1,5 с двигался с ускорением влево и увеличивался.

Выделим 18-й кадр слоя «шар» и преобразуем его в ключевой. Установим шар в крайнее правое

положение и уменьшим его инструментом **Трансформация**.

Выделим последний 36-й кадр и установим шар в крайнее левое положение. Восстановим его размер.

Зададим на первом этапе параметр **Замедление (Ease)** равным 80, а на втором — равным -80.

Сохраним окончательный вариант под именем **dvi1.fla**. Протестируем анимацию.

**Пример 4.** Создать анимацию: пропеллер вентилятора совершает 1 оборот за 1 с.

Находясь в первом кадре единственного слоя, нарисуем овал (рис. 3.43, *а*).

С помощью инструментов выделения трансформируем его в пропеллер (рис. 3.43, *б*) и превратим в символ типа **Графика**.

Центр вращения полученного объекта должен совпадать с центром пропеллера. При необходимости откорректируем положение центра инструментом **Трансформация** (рис. 3.43, *в*).

Превратим 12-й кадр в ключевой.

Произведем автозаполнение промежуточных кадров, выполнив команду **Вставка → Создать анимацию движения (Insert → Create Motion Tween)**.

Выделим любой промежуточный кадр и на панели **Свойства** установим: **Rotate — CW** (Поворот по часовой стрелке), **1 times** (1 оборот) (рис. 3.44).

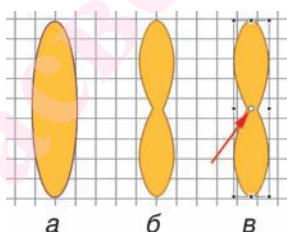


Рис. 3.43

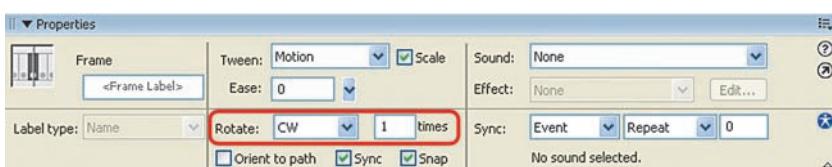


Рис. 3.44

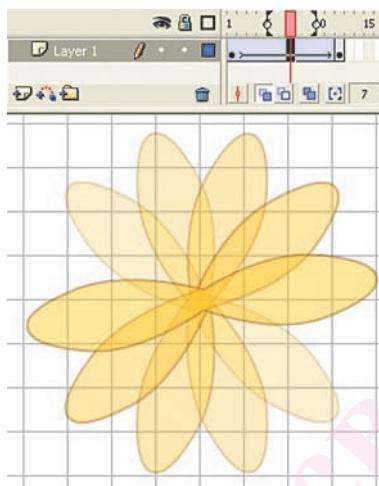


Рис. 3.45

Просмотрим последовательность фаз в режиме калькирования (рис. 3.45).

Сохраним работу под именем **vent.fla**. Протестируем анимацию.

- ?** 1. Какую анимацию называют анимацией движения?
- 2. Как производится автозаполнение промежуточных кадров?
- 3. По какой формуле рассчитывают число кадров для заданной длительности анимации?

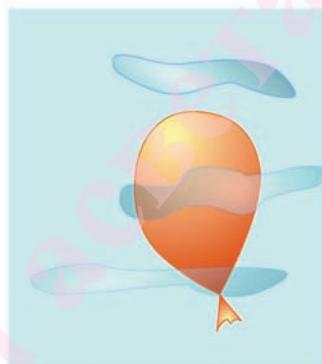
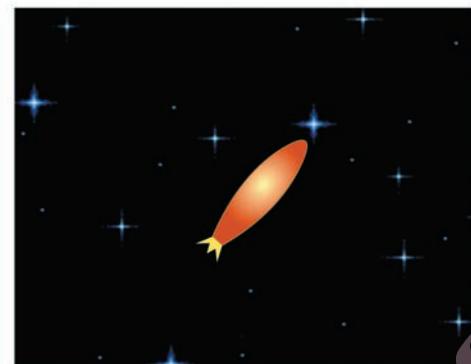
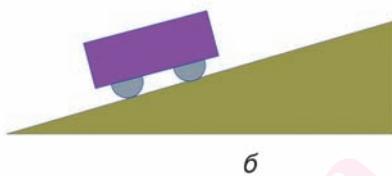
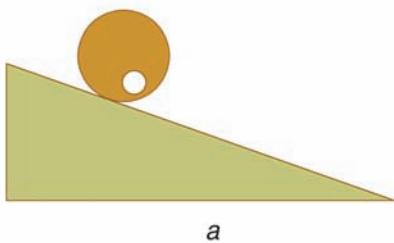
### Упражнение

Создайте анимацию движения по образцу:

а) диск 2 с скатывается по наклонной плоскости и делает 2 оборота (библиотечный образец диск, два слоя: диск и плоскость);

б) тележка 2 с поднимается по наклонной плоскости замедленно, затем 3 с опускается ускоренно (библиотечный образец тележка, два слоя: тележка и плоскость);

в) ракета 3 с движется на фоне звездного неба (библиотечный образец ракета, два слоя: ракета и небо **sky.gif**);



в

г

г) воздушный шар 2 с поднимается вертикально вверх и уменьшается в размерах, затем 2 с опускается и увеличивается (библиотечный образец шар, два слоя: шар и облака).

## § 17. Движение нескольких объектов. Звуковое сопровождение

Анимация движения нескольких объектов создается в несколько этапов:

I. **Расстановка декораций**, т. е. размещение на отдельных слоях объектов, которые должны двигаться независимо.

II. **Создание сценария**, т. е. выделение интервалов (этапов) движения всех объектов и уточнение свойств этих объектов в начале и конце каждого ин-

тервала (полезна запись сценария в словесной форме или в виде схемы).

**III. Реализация сценария**, т. е. расстановка ключевых кадров, задание параметров объектов в этих кадрах, генерация промежуточных кадров на каждом из выделенных интервалов.

**Пример 1.** Создать анимацию движения робота и мяча длительностью 2 с. Первую секунду робот движется из крайнего левого положения вправо до удара по мячу, который покоится посередине поля (рис. 3.46). После удара робот движется в обратном направлении, а мяч 0,5 с летит до удара о стену, затем 0,5 с летит в обратном направлении.



Рис. 3.46

- Разработку анимации начнем с расстановки декораций, т. е. с добавления слоев и размещения объектов.

Импортируем рисунок **trees.gif** на единственный слой (Layer 1), который назовем «лес». Дорисуем на этом же слое наклонную стену. Заблокируем этот слой.

Добавим еще два слоя и назовем их «робот» и «мяч» (рис. 3.47, а). Импортируем на них соответствующие изображения **robot.gif** и **ball.gif**, преобразовав их в символы с именами «робот» и «мяч».

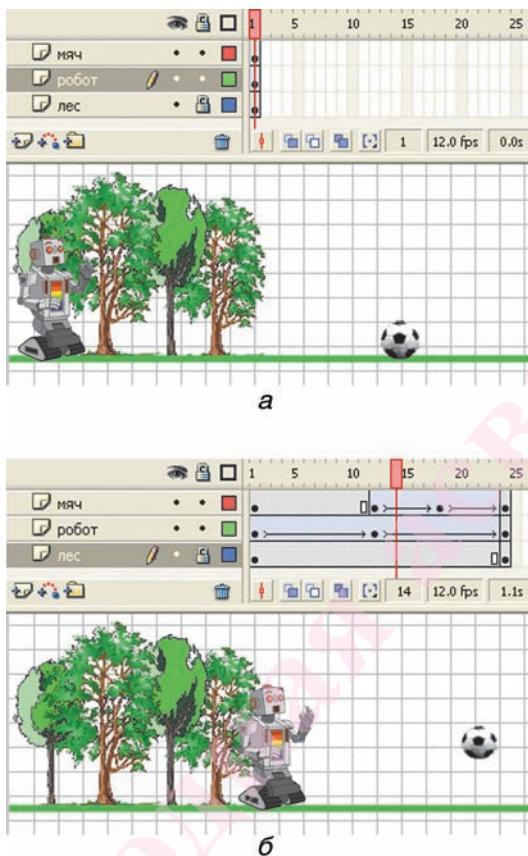


Рис. 3.47

- Теперь займемся расстановкой ключевых кадров.

Поскольку лес неподвижен на протяжении всех 2 с, превратим 24-й кадр слоя «лес» в ключевой.

Робот имеет два этапа движения: первую секунду он движется вправо до удара по мячу в 12-м кадре, а затем — в обратном направлении. Поэтому в слое «робот» вставим ключевые кадры 12 и 24, в которых поместим робота в соответствующие положения. Выделим промежуточные кадры каждого этапа, например кадры 7 и 15, и произведем их автозаполнение.

У мяча три этапа движения: 1) первую секунду (с 1-го кадра по 12-й) он покойится; 2) затем после удара робота 0,5 с он летит к стене; 3) в 18-м кадре мяч ударяется о стену и 0,5 с летит в обратном направлении.

Поэтому в слое «мяч» вставим ключевые кадры 12, 18 и 24, в которых установим мяч в соответствующие положения.

Выделим промежуточные кадры каждого этапа, например кадры 7, 15 и 20, и произведем их автозаполнение (рис. 3.47, б).

- В заключение протестируем созданную анимацию и при необходимости произведем коррекцию. Сохраним работу под именем **robot.fla**.

Flash-анимация может содержать звуковое сопровождение. Звуки импортируются из файлов форматов **.wav** и **.mp3**. Они хранятся в библиотеке и вставляются в нужные места фильма.

**Пример 2.** Вставить в анимацию **robot.fla** звук удара робота по мячу.

С помощью команд **Файл → Импорт (File → Import)** импортируем звук удара из файла **udar.wav** в библиотеку (рис. 3.48).

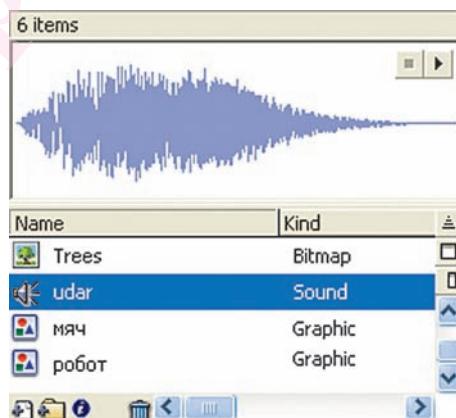


Рис. 3.48

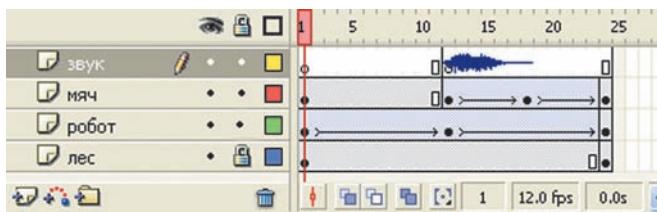


Рис. 3.49

Добавим слой, который назовем «звук». Превратим 12-й кадр этого слоя в ключевой и перетащим звук из библиотеки на рабочее поле (рис. 3.49).

Сохраним работу под именем **robot2.fla**. Протестируем созданную анимацию.

- ? 1. Перечислите этапы создания анимации нескольких движений.  
2. Какой командой помещают звук в библиотеку?

### Упражнение

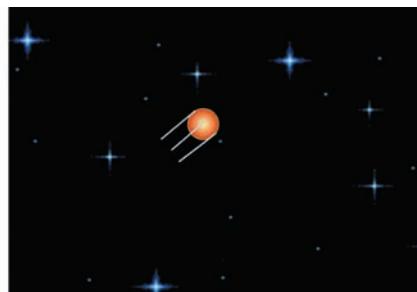
Создайте анимацию по образцу:

а) воздушный шар поднимается вверх. Облака движутся горизонтально (библиотечные образцы шар и облако, два слоя: шар и облака);

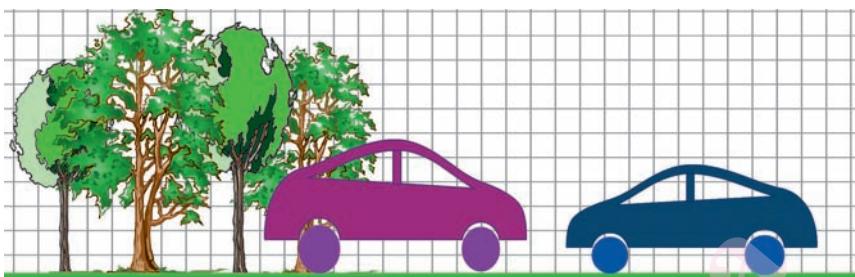
б) спутник движется на фоне звездного неба и посылает звуковые сигналы (библиотечный образец спутник, три слоя: спутник, небо **sky.gif** и звук **bip.mp3**);



а



б



в) два автомобиля движутся навстречу друг другу на фоне неподвижных деревьев (библиотечный образец автомобиль, три слоя: деревья, синий и фиолетовый автомобили).

### § 18. Движение по траектории

Анимация движения по заданной траектории осуществляется с помощью специального направляющего слоя (**Motion Guide**). Его располагают непосредственно над слоем с движущимся объектом, который становится ведомым.

**Пример 1.** Создать анимацию падения яблока с башни по криволинейной траектории (рис. 3.50).



Рис. 3.50

Импортируем изображение **tower.jpg** в 1-й кадр слоя, который назовем «башня».

Добавим слой с именем «яблоко». Импортируем на него рисунок **apple.gif**, превратив его в символ «яблоко».

Нажатием кнопки  под списком слоев добавим слой типа **Направляющий (Motion Guide)** непосредственно над слоем «яблоко» (рис. 3.51).

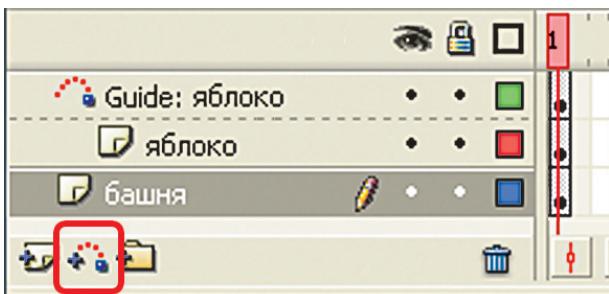


Рис. 3.51

На направляющем слое нарисуем линию от вершины до основания башни. Цвет линии не имеет значения, поскольку она не будет видна при демонстрации фильма. С помощью инструмента выделения **Стрелка** трансформируем ее в требуемую кривую (траекторию движения). Во избежание случайного смещения траектории заблокируем этот слой.

Инструментом **Стрелка** выделим экземпляр символа «яблоко» и переместим его с нажатой левой кнопкой мыши к начальной точке траектории. При отпусканье кнопки мыши объект связывается с началом траектории (рис. 3.52, а).

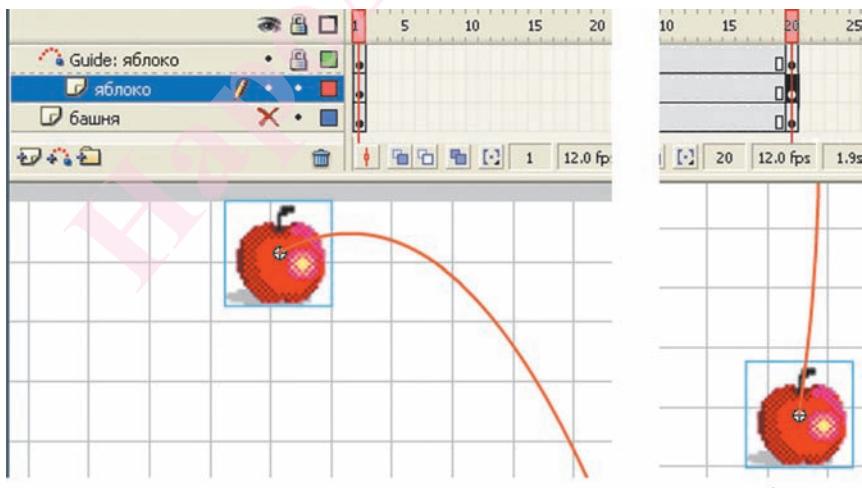


Рис. 3.52

Выделим 20-й кадр в трех слоях и превратим в ключевой клавишей F6.

Свяжем «яблоко» в 20-м кадре с концом траектории (рис. 3.52, б).

Заметим, что для привязки к траектории модификатор **Защелка (Snap)** должен быть включен (магнит в области **Options** панели инструментов).

Выделим любой промежуточный кадр слоя «яблоко» и произведем автозаполнение кадров.

Просмотрим последовательность кадров.

Для ориентации движущегося объекта относительно траектории можно установить флагок **Orient to path** на панели **Свойства**. В этом случае яблоко будет поворачиваться при движении.

Сохраним работу под именем **tower.fla**. Протестируем анимацию.

**Пример 2.** Создать анимацию движения Луны вокруг Земли с периодом 3 с (рис. 3.53).

Импортируем изображения звездного неба (**sky.gif**), Земли (**zem.gif**) и Луны (**luna.gif**) на слои с именами «Небо», «Земля», «Луна».

Превратим изображение Луны в символ.

Над слоем «Луна» добавим направляющий слой, на котором изобразим траекторию (oval без заливки). Инструментом **Ластик (Eraser)** удалим небольшой фрагмент замкнутой орбиты, чтобы обеспечить привязку к началу и концу траектории.



Рис. 3.53

Выделим 36-й кадр во всех слоях и превратим его в ключевой.

Привяжем Луну в 1-м кадре к началу траектории, а в 36-м — к концу.

Произведем автозаполнение кадров в слое «Луна».

Сохраним работу под именем **orbita.fla**. Протестируем анимацию.

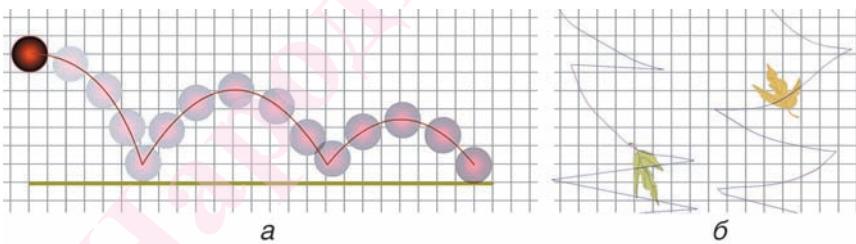
- ? 1. Что понимают под анимацией движения по траектории?
2. Перечислите этапы создания анимации движения по траектории.

### Упражнение

Создайте анимацию по образцу:

а) брошенный в горизонтальном направлении мяч трижды ударяется о пол (библиотечный символ мяч, два обычных слоя: пол и мяч — и один направляющий слой с траекторией мяча);

б) листья падают по криволинейным траекториям и ориентированы вдоль них (библиотечный символ лист, два обычных слоя и два направляющих слоя с криволинейными траекториями).



## § 19. Использование маски

**Маской** называют слой, который скрывает изображения на расположенному под ним **маскируемом** слое. Если на слое-маске поместить объект с заливкой, то через него, как через окно, будет просматриваться участок маскируемого слоя.

**Пример 1.** Создать анимацию движения окна.

В первый кадр слоя с именем «Фон» импортируем изображение *jagoda.jpg*.

Выше добавим второй слой и назовем его «Окно». Нарисуем на нем прямоугольник, залитый любым цветом, выделим его (заливку и обводку) и преобразуем в символ с именем «Окно» (рис. 3.54, *a*).

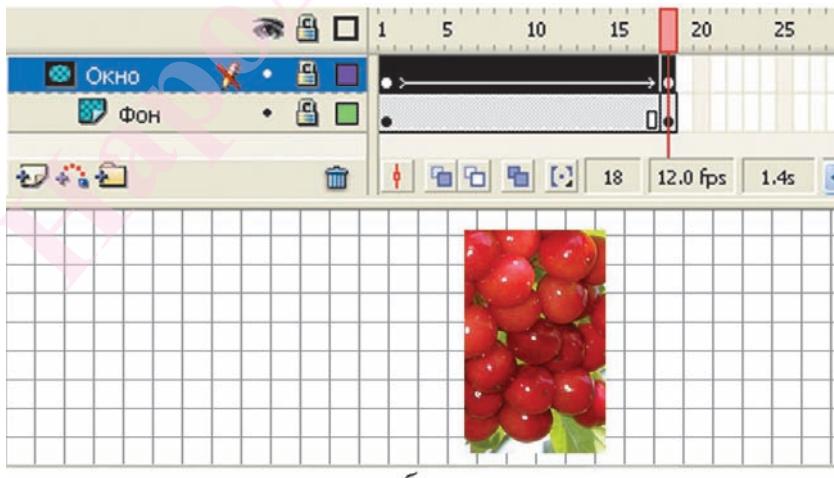
*а**б*

Рис. 3.54

Создадим анимацию движения окна из левого нижнего угла (1-й кадр) в правый верхний (18-й кадр). Для этого выделим 18-й кадр в двух слоях и превратим в ключевой. Выделим промежуточный кадр в слое «Окно» и произведем автозаполнение кадров.

Просмотрим анимацию.

Щелкнем правой кнопкой мыши по слою «Окно» и в открывшемся меню установим тип слоя **Маска (Mask)**. Большая часть фонового изображения станет невидимой, а непрозрачный прямоугольник превратится в «окно». Через него будет виден участок маскируемого слоя «Фон» (рис. 3.54, б).

Сохраним работу под именем **okno.fla**. Протестируем анимацию.

**Пример 2.** Создать анимацию движения панорамы города в телевизоре (рис. 3.55).

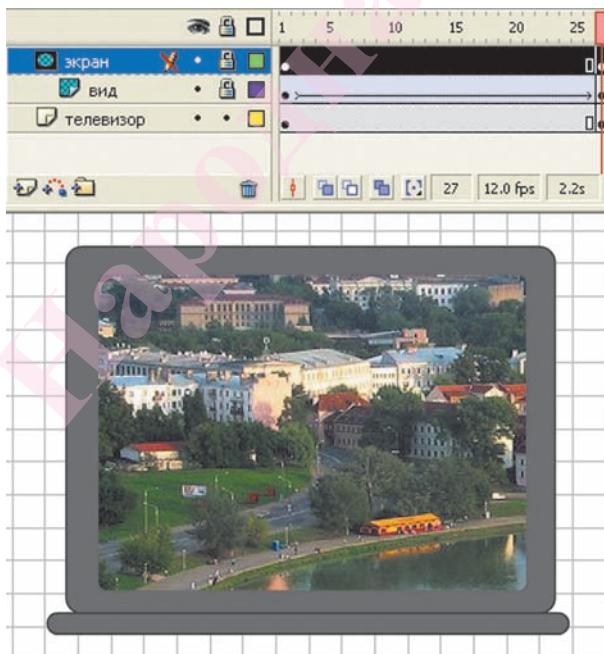


Рис. 3.55

В 1-м кадре слоя «Телевизор» изобразим телевизор (серые прямоугольники со скругленными углами).

Выше добавим слой с именем «Экран». Нарисуем на нем экран телевизора (закрашенный прямоугольник) и преобразуем в символ с именем «Экран». Между этими слоями добавим слой с именем «Вид», на него импортируем панораму города *minsk.jpg*.

На слое «Вид» создадим анимацию движения панорамы города справа (1-й кадр) налево (24-й кадр).

Щелкнем правой кнопкой мыши по слою «Экран» и установим тип **Маска (Mask)**. Через него будет просматриваться участок маскируемого слоя с движущейся панорамой города.

Сохраним работу под именем *tv.fla*. Протестируем анимацию.

Интересные эффекты получаются, если ниже маскируемого слоя поместить еще один слой с полупрозрачным или уменьшенным изображением.

**Пример 3.** Создать анимацию движения линзы над панорамой города.

В первый кадр слоя «Вид» импортируем изображение панорамы города *minsk.jpg* и преобразуем его в символ. С помощью панели **Свойства** установим прозрачность **alpha = 40 %**.

Выше добавим слой с именем «Увеличение». Из библиотеки поместим на него изображение панорамы и увеличим его.

Еще выше добавим третий слой и назовем его «Линза». Нарисуем на нем эллипс с заливкой и преобразуем его в символ с именем «Линза».



Рис. 3.56

В этом слое создадим анимацию движения линзы из левого нижнего угла (1-й кадр) в правый верхний угол (24-й кадр).

Щелкнем правой кнопкой мыши по слою «Линза» и установим тип Маска. Через него будет проецироваться участок маскируемого слоя с увеличенным изображением (рис. 3.56).

Сохраним работу под именем **panorama.fla**. Протестируем анимацию.

- ?
- 1. Какой слой называют маской?
- 2. С какой целью используют маску?

### ***Упражнение***

Создайте анимацию по образцу:

- а) движение лупы над шкалой термометра (рисунок **scale.jpg**);
- б) рыбы проплывают перед иллюминатором подводной лодки (рисунок **fish.jpg**);
- в) наблюдение поверхности Луны через телескоп (рисунок **luna2.jpg**);



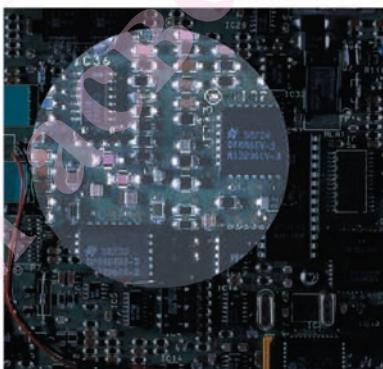
а



б



в



г

г) изучение электронной схемы с помощью микроскопа (рисунок *is.jpg*): 2 с — увеличение и 2 с — возврат к первоначальному размеру.

## § 20. Анимация формы

Рассмотрим еще один вид компьютерной анимации, который заключается в автоматической генерации промежуточных кадров путем **изменения формы** фигур (*Tween Shape*).

**Пример 1.** Создать анимацию превращения зеленого яблока в желтую грушу за 2 с.

В первом кадре единственного слоя нарисуем зеленый овал без обводки. С помощью ин-

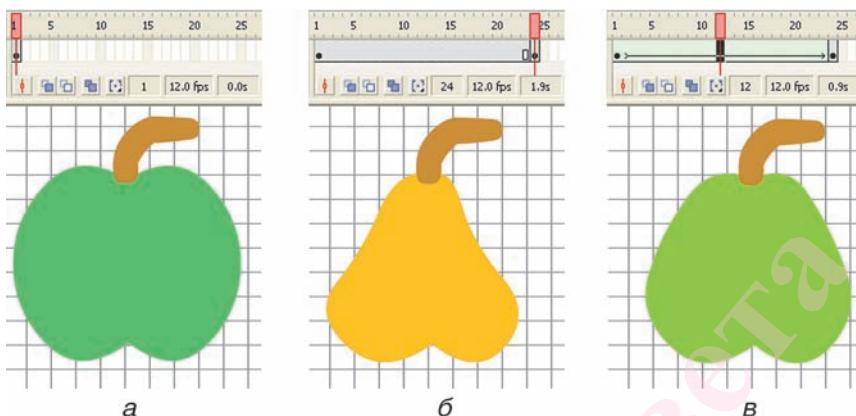


Рис. 3.57

струментов выделения придадим ему форму яблока. Инструментом **Кисть (Brush)** дорисуем веточку (рис. 3.57, *а*).

Выделим 24-й кадр и преобразуем его в ключевой с заполнением. В этом кадре придадим изображению форму груши и изменим цвет заливки (рис. 3.57, *б*).

Произведем автозаполнение кадров типа **Tween Shape**. Для этого выделим любой промежуточный кадр (например, 12-й) и в поле **Tween** панели **Свойства** установим **Shape (Форма)** (рис. 3.58). Светло-зеленая окраска и стрелка от 1-го кадра к 24-му указывает на то, что промежуточные кадры сгенерированы (рис. 3.57, *в*). (В случае ошибки стрелка рисуется пунктиром.)



Рис. 3.58

Просмотрим последовательность фаз анимации, перемещая указатель кадров по шкале времени, а также в режиме калькирования.

Сохраним работу под именем **morf1.fla**. Протестируем анимацию.

**!** Анимацию формы нельзя применять к экземплярам символов, а также к сгруппированным объектам.

Рассматриваемый вид анимации позволяет не только изменять форму объекта, но и превращать одни объекты в другие. Нередко при этом получаются интересные и даже неожиданные эффекты.

**Пример 2.** Создать анимацию превращения красного круга в синий квадрат за 1,5 с.

В первом кадре единственного слоя нарисуем красный круг (рис. 3.59, *а*).

Выделим 18-й кадр и превратим его в пустой ключевой нажатием клавиши F7 или командой Вставка → Пустой Ключевой кадр (Insert → Blank Keyframe).

В этом кадре того же слоя нарисуем синий квадрат (рис. 3.59, *б*).

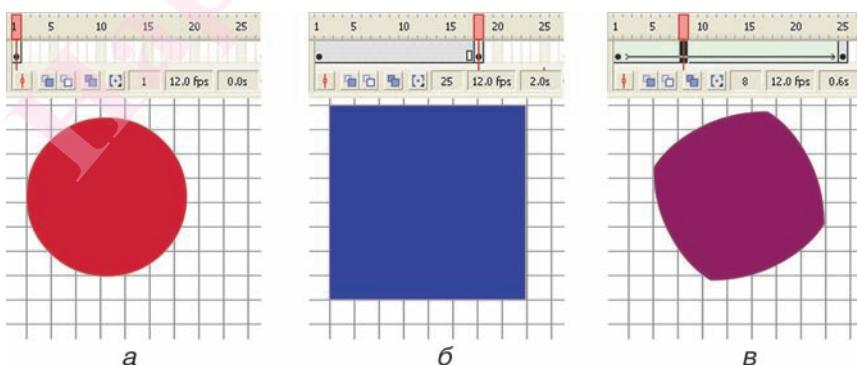


Рис. 3.59

Произведем автозаполнение кадров, выбрав любой промежуточный кадр, например 8-й (рис. 3.59, в), и в поле **Tween** панели **Свойства** установив **Shape**.

Сохраним работу под именем **morf2.fla**. Протестируем анимацию.

Промежуточные фазы трансформации обводки и заливки фигур могут различаться. Для получения более предсказуемых трансформаций проще использовать фигуры без обводки, а также располагать их составные части на разных слоях.

**Пример 3.** Анимировать рост цветка: росток первую секунду увеличивается в размерах, за вторую секунду у него вырастают листья и бутон, за третью секунду распускаются лепестки.

В первом кадре единственного слоя (назовем его «Росток») инструментом **Кисть** без обводки изобразим маленький стебелек, бутон и листочки (рис. 3.60, а). При рисовании следует установить увеличение 400 %.

Преобразуем 12-й кадр в ключевой. Выделим все изображение (стебелек, бутон и листочки) и с помощью инструмента **Трансформация** увеличим его (рис. 3.60, б).

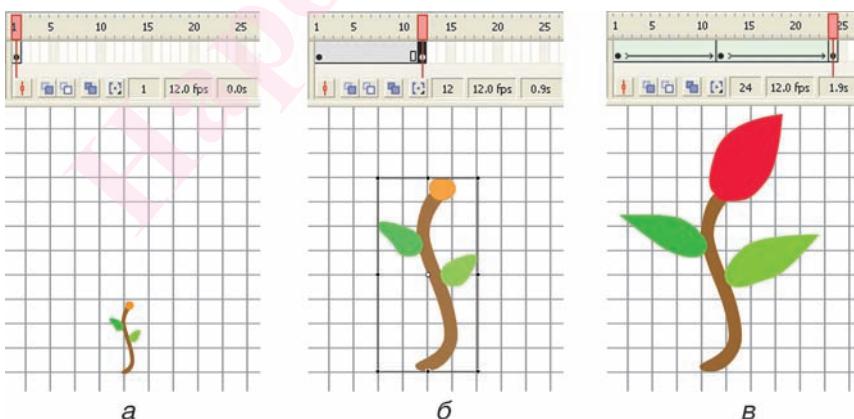


Рис. 3.60

Преобразуем 24-й кадр в ключевой. Инструментами выделения придадим листьям и бутону требуемые размеры и форму (рис. 3.60, в).

Произведем автозаполнение кадров Tween Shape, выделив промежуточные кадры для каждого из двух этапов анимации (например, 7-й и 18-й).

Сохраним работу под именем *rostok.fla*. Протестируем анимацию.

Продолжим создание анимации. Добавим новый слой «Лепестки». Дальнейшие трансформации будут происходить на этом слое, поэтому слой «Росток» заблокируем. На время рисования покажем его содержимое в виде контуров, щелкнув мышью по квадратику рядом с именем этого слоя.

Преобразуем 24-й кадр слоя «Лепестки» в ключевой. На контуре бутона цветка инструментом Кисть без обводки изобразим два маленьких лепестка (рис. 3.61, а).

Преобразуем 36-й кадр обоих слоев в ключевой.

Инструментами выделения и трансформации придадим лепесткам требуемые размеры и форму (рис. 3.61, б).

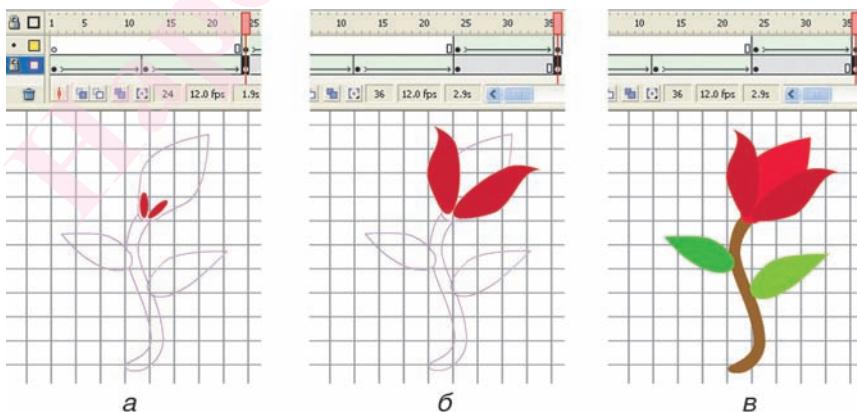


Рис. 3.61

Произведем автозаполнение кадров Tween Shape, выделив промежуточный кадр третьего этапа анимации на слое «Лепестки» (например, 32-й).

Вернем видимость содержимого слоя «Росток» (рис. 3.61, *в*).

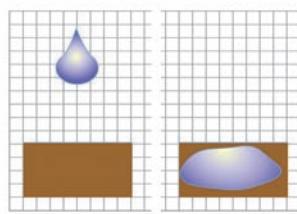
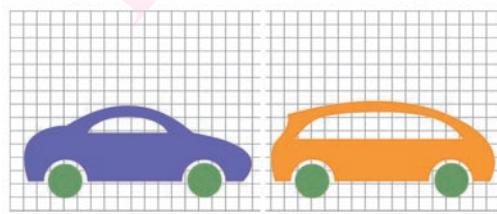
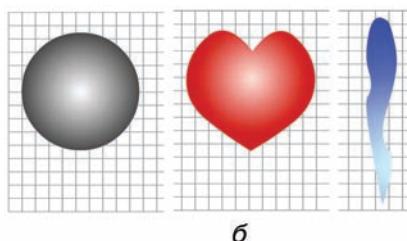
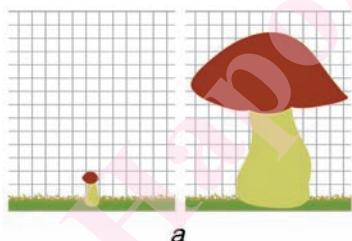
Сохраним работу под именем **rostok2.fla**. Протестируем анимацию.

- ? 1. Что понимают под анимацией формы?  
2. К каким объектам нельзя применять анимацию формы?

### Упражнение

Создайте анимацию по образцу:

- а) рост гриба;
- б) превращение серого шарика в красное сердечко, а затем в сосульку;
- в) превращение синего легкового автомобиля в оранжевый микроавтобус;
- г) падение и растекание капли воды (2 слоя, 3 ключевых кадра).



## § 21. Работа с текстом

### 21.1. Ввод и форматирование

В анимациях нередко используется текст. В редакторе Flash можно создавать и редактировать тексты, подобно тому как это делается в обычных текстовых редакторах. Кроме того, текстовые объекты можно превращать в графические и работать с ними, как с изображениями. При этом, однако, утрачивается возможность редактировать и форматировать текст (исправлять ошибки, изменять шрифт и т. п.). Таким образом, существует принципиальное различие между текстовым и графическим объектом.

**Пример 1.** Ввести текст ПРИВЕТ (шрифт Times, 40 пт, цвет зеленый). Отформатировать его (шрифт Arial, 50 пт, цвет синий).

Возьмем инструмент **Текст (Text)**. Укажем на рабочем поле место ввода щелчком мыши в точке, где будет верхний левый угол прямоугольной области текста.

Параметры текста настраиваются на панели **Свойства** (рис. 3.62) перед его вводом или при последующем форматировании. Цвет можно задавать на панели **Смеситель Цветов**.

Установим шрифт Times, 40 пт, цвет зеленый. Тип текста должен быть установлен по умолчанию **Статичный (Static)**. Введем текст (рис. 3.63, а).

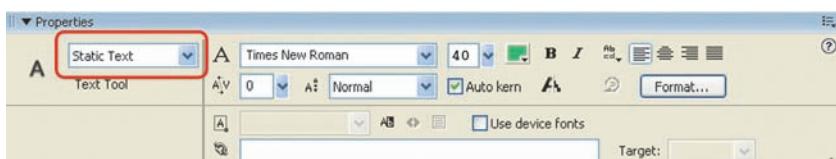


Рис. 3.62

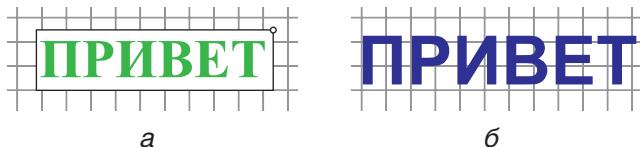


Рис. 3.63

Для форматирования выделим текст и на панели **Свойства** установим шрифт Arial, 50 пт, цвет синий. По окончании щелкнем мышью в любом месте вне текстовой области, и выделение будет снято. Результат показан на рисунке 3.63, б.

К текстовым объектам применимы лишь некоторые трансформации: **Масштаб (Scale)**, **Поворот (Rotate)** и **Наклон (Skew)**. Преобразования **Искажение (Distort)** и **Огибание (Envelope)** недоступны. Нельзя применять градиентную заливку, но можно задавать прозрачность параметром **Alpha**.

**Пример 2.** Трансформировать текстовый объект из примера 1 в соответствии с рисунком 3.64.

Выделим текстовый объект, скопируем в буфер обмена и вставим две его копии.

С помощью инструмента **Трансформация** первую копию уменьшим и повернем (см. рис. 3.64, а), а ко второй применим деформацию **Наклон** (см. рис. 3.64, б).

Текст можно превратить в графический объект. Тогда его нельзя будет редактировать и форматиро-

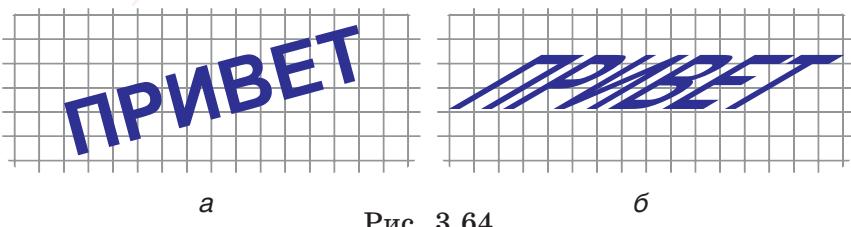


Рис. 3.64

вать как текстовый объект, но можно будет применять все инструменты, предназначенные для графики.

**Пример 3.** Превратить текстовый объект из примера 1 в графический и трансформировать в соответствии с рисунком 3.65.



Рис. 3.65

С помощью буфера обмена вставим на рабочее поле три копии текста.

Превратим их в графические объекты, дважды применив команду **Модифицировать → Разделить** (**Modify → Break Apart**) (или дважды нажав клавиши **Ctrl + B**). При первом применении текст делится на отдельные буквы (рис. 3.66), а при втором — каждая буква превращается в графический объект.

Инструментом **Трансформация** к первой копии применим преобразование **Искажение** (**Distort**) (см. рис. 3.65, *а*), а ко второй — **Огибание** (**Envelope**) (см. рис. 3.65, *б*).

В третьей копии с помощью инструментов выделения и трансформации изменим форму каждой буквы и применим градиентные заливки (см. рис. 3.65, *в*).

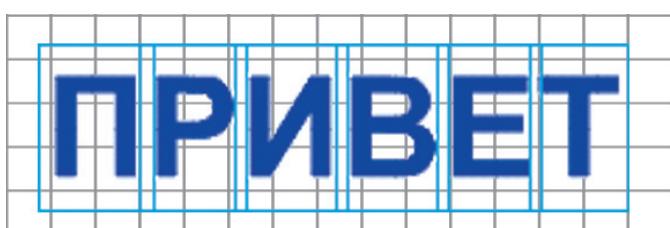


Рис. 3.66

## 21.2. Анимация текста

Рассмотрим особенности анимации текста.

Для текста как сгруппированного объекта можно применять анимацию движения.

**Пример 4.** Создать анимацию «Бегущая строка».

Находясь в первом кадре единственного слоя, введем слово **Движение**.

Превратим 24-й кадр в ключевой.

В 1-м кадре переместим слово за пределы рабочего поля вправо, а в 24-м — влево.

Произведем автозаполнение кадров типа **Motion Tween**. При этом текст превратится в библиотечный символ типа **Графика** (как сгруппированный объект).

Сохраним работу. Протестируем анимацию.

**Пример 5.** Создать анимацию «Рассыпанные буквы».

Находясь в первом кадре единственного слоя, введем слово **Привет**.

Для разделения слова на буквы применим операцию **Разделить (Break Apart)** из меню **Modify** (или комбинацию клавиш **Ctrl + B**). Чтобы обеспечить независимость движения каждой буквы, распределим их по отдельным слоям командой **Шкала времени → Распределить по слоям (Timeline → Distribute to Layers)** того же меню. При этом каждая буква превратится в библиотечный символ типа **Графика** и будет помещена на отдельный слой (рис. 3.67, *a*).

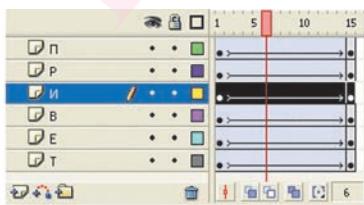
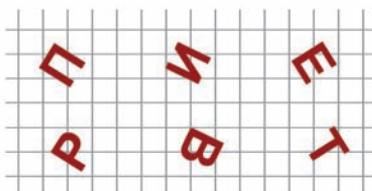
*a**б*

Рис. 3.67

Превратим 15-й кадр всех слоев в ключевой.

В 1-м кадре переместим буквы в разные места рабочего поля и повернем их (рис. 3.67, б).

Выделим промежуточный кадр во всех слоях и произведем автозаполнение кадров типа **Motion Tween**.

Сохраним работу. Протестируем анимацию.

! Анимация формы неприменима для текста. Сначала необходимо превратить текст в графические объекты.

**Пример 6.** Создать анимацию превращения буквы А в букву Б за 1,5 с.

Находясь в первом кадре единственного слоя, напишем букву А синего цвета размером 96 пт (рис. 3.68, а).

Выделим 12-й кадр и превратим его в пустой ключевой, нажав клавишу F7. В этом кадре напишем букву Б красного цвета такого же размера (рис. 3.68, б).

Инструментом выделения **Стрелка** выделим буквы в первом и последнем кадрах и применим операцию **Разделить (Break Apart)** с помощью комбинации клавиш Ctrl + B.

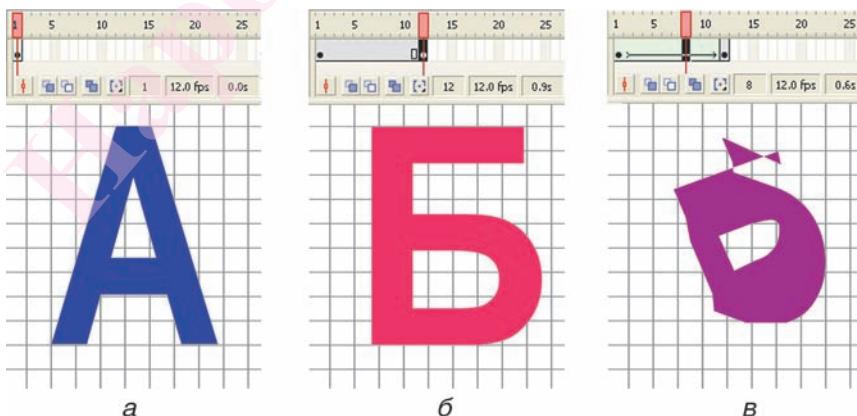


Рис. 3.68

После этого произведем автозаполнение кадров типа **Tween Shape** (рис. 3.68, в).

Сохраним работу под именем **ab.fla**. Протестируем анимацию.

! К словам операция разделения применяется дважды, поскольку слова сначала разделяются на отдельные буквы.

**Пример 7.** Создать анимацию превращения слова **МУХА** в слово **СЛОН**.

В первом кадре напишем слово **МУХА**. Превратим 20-й кадр в пустой ключевой и напишем слово **СЛОН**.

Преобразуем слова в графические объекты, дважды применив операцию **Разделить (Break Apart)**. После этого произведем автозаполнение кадров типа **Tween Shape**.

Сохраним работу. Протестируем анимацию.

? 1. Каким образом можно превратить в графический объект:  
а) букву;      б) слово?

### Упражнения

1. Создайте анимацию всплывающего снизу и увеличивающегося слова **ШКОЛА** (пример показан в режиме калькирования).

2. Создайте анимацию превращения слова **ПРИВЕТ** в слово **ПРОЩАЙ**.

3. Создайте поздравительную открытку с ани-





мированным текстом и изображением.

**4.** Создайте анимацию движения слова **МИНСК** (бегущая строка) над словом с видом Минска (рисунок *minsk.jpg*).

## § 22. Создание и использование клипов

Создание анимаций с несколькими повторяющимися элементами и движениями значительно упрощается при разделении фильма на составные части и анимировании их по отдельности. Экземпляры клипов, как и любых библиотечных образцов, могут использоваться многократно.

В общем случае создание анимационного фильма содержит два основных этапа: разработка клипов и сборка всего фильма.

**Пример 1.** Создать клип: колобок открывает и закрывает рот.

С помощью меню **Вставка → Новый Символ (Insert → New Symbol)** или клавиш **Ctrl + F8** вызовем диалоговое окно **Создать новый символ (Create New Symbol)**. Установим его тип **Клип (Movie Clip)** и назовем «Колобок». В результате этих действий откроется рабочее поле (сцена) клипа со своей шкалой времени.

В центре рабочего поля клипа (отмечен крестиком) нарисуем шар (круг с заливкой радиальным градиентом). С помощью инструментов выделения трансформируем его (рис. 3.69, кадр 1).

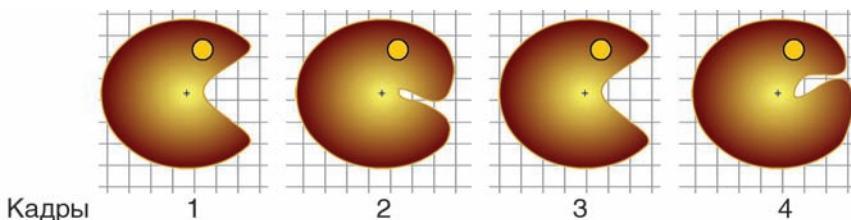


Рис. 3.69

Создадим четырехкадровую мультипликацию. Для этого продублируем содержимое первого ключевого кадра еще на три кадра. Теперь во всех четырех кадрах будет одинаковый рисунок.

В кадрах 2 и 4 с помощью инструментов выделения изменим изображение в соответствии с рисунком 3.69. Просмотрим последовательность полученных кадров, перемещая указатель кадра вдоль шкалы времени клипа (рис. 3.70).

Вызовем окно библиотеки с помощью меню **Окно → Библиотека** (**Window → Library**) или клавиш **Ctrl + L**, и, нажав кнопку , просмотрим клип в динамике в окне библиотеки (рис. 3.71).

Итак, клип «колобок» создан и находится в библиотеке. Рассмотрим второй этап создания фильма — его сборку.



Рис. 3.70

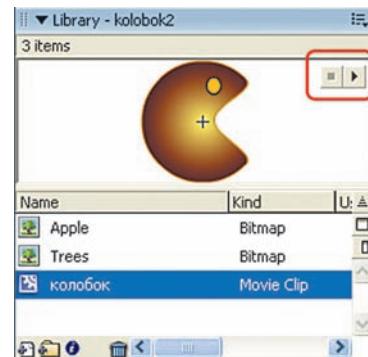


Рис. 3.71

**Пример 2.** Создать фильм длительностью 2 с: из-за деревьев появляется колобок, 1 с он движется к яблоку, съедает его, разворачивается и движется обратно.

Прежде всего, вернемся на основную сцену, щелкнув по надписи **Сцена 1 (Scene 1)** на рамке шкалы времени клипа.

Находясь в кадре 1, расставим «декорации» на основной сцене. Импортируем рисунок *trees.gif*. Подкорректируем его положение и размеры. Назовем слой «фон». Добавим еще два слоя. Назовем их «яблоко» и «колобок». На слой «колобок» слева поместим экземпляр клипа «колобок» из библиотеки (при этом отображается только его первый кадр без анимации). На слой «яблоко» в центр рабочего поля импортируем изображение яблока *apple.gif*. Чтобы колобок оказался за деревьями, поместим слой «фон» выше слоя «колобок» и заблокируем его (рис. 3.72).

Теперь займемся расстановкой ключевых кадров.

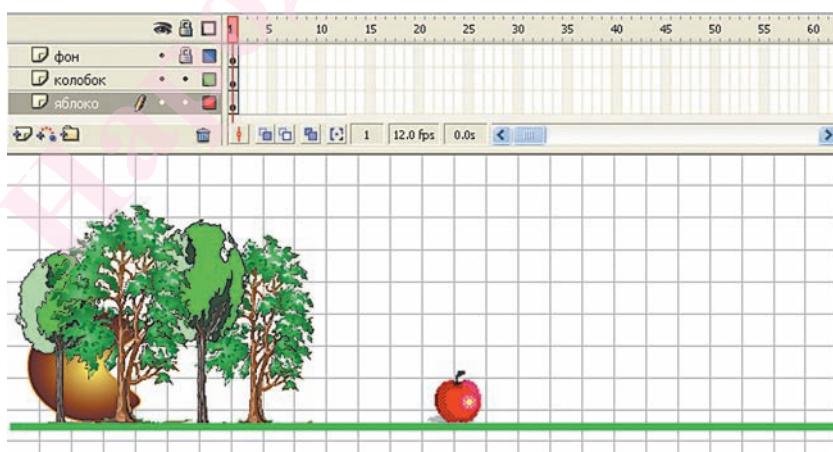


Рис. 3.72

Фон неподвижен на протяжении всего фильма, поэтому превратим 24-й кадр этого слоя в ключевой с заполнением.

Яблоко покоится 1 с, а затем исчезает. Превратим 12-й кадр слоя «яблоко» в ключевой.

Колобок имеет три этапа движения: первую секунду (с 1-го кадра по 12-й) он движется вправо к яблоку, в 13-м кадре разворачивается, затем движется в обратном направлении. Поэтому в слое «колобок» вставим ключевой кадр 12, в котором подберем положение колобка так, чтобы он закрывал яблоко. Затем вставим ключевой кадр 13. В нем с помощью инструмента Трансформация или меню Модифицировать → Трансформировать → Перевернуть горизонтально (Modify → Transform → → Flip Horizontal) развернем колобка на 180°. Наконец, вставим ключевой кадр 24 и установим колобка слева за деревьями.

Теперь можно произвести автозаполнение кадров выделив промежуточные кадры этого слоя, соответствующие этапам движения вперед и назад, например 9 и 16 (рис. 3.73).

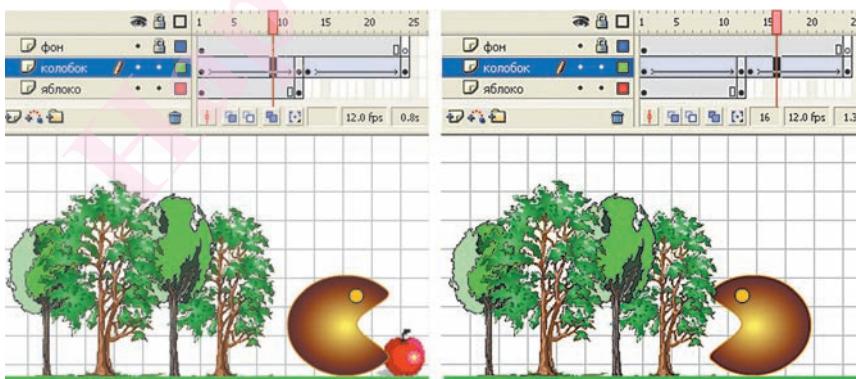


Рис. 3.73

Сохраним работу под именем **kolobok.fla**. Протестируем анимацию.

- ? 1. В каких случаях и с какой целью создают библиотечные символы типа Клип?
- 2. Как просмотреть библиотечный клип в динамике?

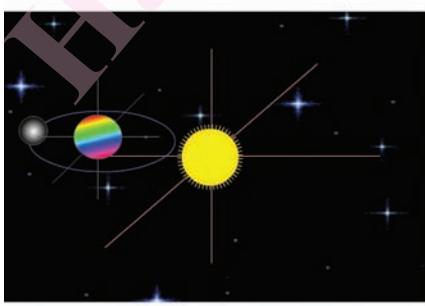
### Упражнение

Создайте анимацию с использованием клипов по образцу:

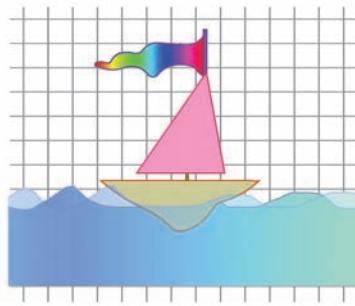
а) плывущая рыба (прозрачность воды **alpha = 30**, библиотечные символы (клипы): рыба, двигающая хвостом и плавниками, колеблющиеся водоросли);



а



б



в

- б) Луна вращается вокруг Земли, а Земля — вокруг Солнца по эллиптическим орбитам (библиотечные символы: Земля (графика), Луна (графика), система Земля — Луна (клип));
- в) лодка движется в горизонтальном направлении и качается на волнах. На мачте развевается вымпел (библиотечные символы: лодка с вымпелом (клип), вымпел (клип), волна (графика)).

# Глава 4

## ИНФОРМАЦИОННЫЕ РЕСУРСЫ СЕТИ ИНТЕРНЕТ

### § 23. Подключение к сети Интернет

С каждым днем все больше разнообразных устройств получают возможность подключаться к сети Интернет. Это персональные и мобильные компьютеры, серверы и локальные сети, телефоны, игровые приставки и музыкальные центры. Их доступ к Сети и обслуживание обеспечивают **интернет-провайдеры** — организации, предоставляющие услуги Интернет.

Подключение к сети Интернет может осуществляться различными способами (рис. 4.1). Наиболее распространенными являются:

- коммутируемый доступ по телефонной линии Dial-Up;
- доступ по цифровой абонентской линии ADSL;
- доступ по выделенному каналу связи;

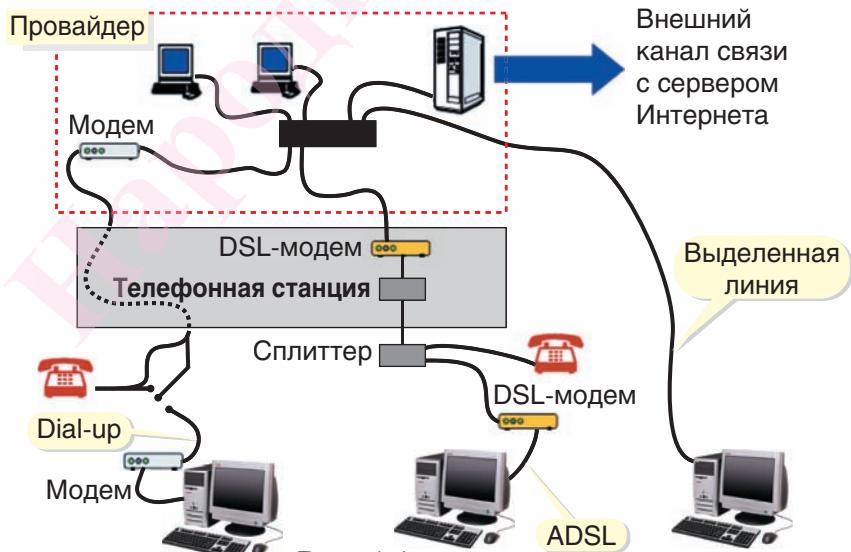


Рис. 4.1

- беспроводная цифровая связь;
- беспроводное подключение по мобильному телефону.

Преобразование информации из цифровой формы в электрические сигналы для передачи их по линиям связи и наоборот осуществляется с помощью **модема** (сокращение от слов «модулятор-демодулятор»).

Выбор типа модема определяется способом подключения к сети: аналоговый — для обычного подключения по телефону, цифровой — для ADSL-подключения, радио-модем — для беспроводного подключения (рис. 4.2, а).

Для персональных компьютеров чаще всего используют внутренний модем, который с помощью стандартного разъема подключается к слоту на материнской плате (рис. 4.2, б). Большинство современных ноутбуков имеют встроенный модем.

Важной характеристикой любой компьютерной сети является скорость передачи информации, т. е. количество информации, которое передается в единицу времени. За единицу скорости передачи принят 1 бит/с. Скорость передачи информации в современных компьютерных сетях достигает сотен миллионов битов в секунду. Поэтому используют производные единицы: килобиты в секунду (кбит/с) или мегабиты в секунду (Мбит/с). При этом  $1 \text{ кбит/с} = 10^3 \text{ бит/с}$ ,  $1 \text{ Мбит/с} = 10^3 \text{ кбит/с} = 10^6 \text{ бит/с}$ .

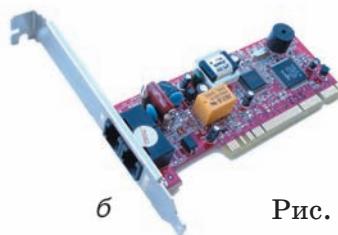


Рис. 4.2

При коммутируемом доступе по телефонной линии (Dial-Up) модем по команде компьютера набирает номер телефона провайдера и, если телефонная линия не занята, устанавливает соединение с модемом провайдера. Поэтому такой способ подключения называют Dial-Up (по звонку). Затем проверяется имя пользователя и пароль. Если авторизация прошла успешно, осуществляется подключение к Интернету. На время сеанса компьютеру присваивается временный адрес. Во время сеанса работы в Интернете никто не может дозвониться до абонента — телефонная линия занята.

Скорость приема и передачи данных для аналоговых модемов достигает 56 кбит/с. Реальная скорость при коммутируемом доступе зависит от многих факторов: пропускной способности внешнего канала связи, количества одновременно подключенных пользователей, состояния телефонной линии.

Чем больше пользователей подключается, тем больше должна быть пропускная способность внешнего канала. Поэтому провайдер должен иметь многоканальный телефон и скоростной внешний канал связи, например со скоростью 30—60 Мбит/с и выше.

Более перспективной является технология ADSL (Asymmetric Digital Subscriber Line — асимметрическая цифровая абонентская линия), которая позволяет передавать данные по телефонным сетям со скоростью до 8 Мбит/с к абоненту и до 1,5 Мбит/с от абонента (отсюда и название асимметричная).

Реальная скорость передачи зависит от протяженности и качества линии. Обмен данными не мешает телефонному разговору благодаря разделению диапазонов частот сигналов в телефонной линии.

Для подключения к ADSL требуется цифровой (DSL) модем и устройство разделения сигналов (сплиттер).

**Доступ по выделенному каналу связи** осуществляется путем постоянного подключения компьютера пользователя к серверу провайдера. Провайдер проводит к компьютеру абонента выделенную линию (витая пара или оптоволокно) и выдает постоянный адрес. Пользователь получает постоянную связь с Интернетом, высокое качество соединения и передачи данных, высокую скорость (до 100 Мбит/с). Из оборудования необходима только сетевая карта (в настоящее время она имеется на большинстве материнских плат).

Стоимость прокладки выделенной линии зависит от расстояния до точки подключения провайдера. Для индивидуальных пользователей этот способ доступа оправдан лишь тогда, когда на прокладку самой линии не требуется значительных затрат. Оптимальным вариантом является подключение к **домашним или городским локальным сетям**, в которых организован общий доступ к высокоскоростным интернет-соединениям. Подключаться к подобным сетям можно различными способами — фактически это обычные локальные сети.

В последнее время получило широкое распространение подключение к **Интернету по сетям кабельного телевидения**. Возможны два варианта. При индивидуальном варианте кабельный модем устанавливается отдельно у каждого компьютера. При коллективном варианте в доме на несколько пользователей устанавливается один модем. Затем проводится локальная сеть и устанавливается необходимое оборудование. Достоинства: хорошая скорость, возможность просматривать цифровые каналы кабельного телевидения.

**Беспроводная цифровая связь WiFi** позволяет получить доступ к Интернету с помощью специального адаптера. В большинство современных ноутбуков он встроен.

Существует несколько стандартов WiFi. Они различаются скоростью передачи данных, которая может достигать 50 Мбит/с. Реальная скорость передачи данных значительно ниже (как минимум в несколько раз), но и этого достаточно для удобной работы в Интернете.

Использование WiFi возможно, если Вы находитесь в зоне действия приемных устройств (эти устройства называют точками доступа). Достоинства: мобильность, неплохая скорость, минимум оборудования. Недостатки: небольшой радиус зоны покрытия одной точки доступа, проблема прямой видимости, ограничение количества пользователей в зоне действия WiFi.

**Беспроводное подключение к Интернету по мобильному телефону** осуществляется с помощью специальных протоколов мобильной связи.

Один из сервисов наиболее распространенных в настоящее время сетей сотовой связи — **GPRS** (General Packet Radio Service). Максимальная скорость передачи данных по стандарту GPRS достигает 170 кбит/с. Реальная скорость обычно не превышает 30—40 кбит/с и зависит от загруженности и возможностей сети сотового оператора, расстояния до антенны, характеристик конкретного мобильного телефона.

Иногда GPRS-сети называют сетями второго поколения. Начинают развиваться и сети третьего поколения. Например, к ним относятся стандарты мобильного подключения CDMA и EDGE.

- ?
- 1. Какие устройства могут подключаться к Интернету?
- 2. Какие способы подключения к сети Интернет наиболее распространены?
- 3. Для чего используется модем?
- 4. При каких способах подключения к сети Интернет модем не нужен?

## § 24. Организация службы WWW

Служба Всемирной информационной паутины (World Wide Web, или WWW) объединяет большинство услуг сети Интернет. Создание Всемирной информационной паутины было вызвано необходимостью объединения различных сетей и услуг, а также расширения возможностей доступа к различным информационным ресурсам.

Локальные сети с помощью устройств подключения (маршрутизаторов) через линии связи объединяются в пределах одного района, города, области в региональные сети, которые в свою очередь подсоединяются к другим региональным сетям, образуя Всемирную глобальную сеть (рис. 4.3). В глобальных компьютерных сетях ограничений на расстояние между компьютерами не существует.



Рис. 4.3

Различают два типа компьютерных сетей: *одноранговые сети и сети с выделенным сервером*. В одноранговых сетях все компьютеры равноправны. Такое построение используется для небольших локальных сетей.

Сеть с *выделенным сервером* имеет один высокопроизводительный компьютер (*сервер*), который управляет работой сети. Он предоставляет свои ресурсы для совместного использования остальным компьютерам сети, которые называются *клиентами*. Технология *клиент — сервер* лежит в основе функционирования WWW.

Основой программного обеспечения компьютерной сети являются операционные системы. Современные операционные системы (Windows, системы семейства Unix, MacOS и др.) поддерживают функционирование компьютерной сети.

Все остальное сетевое программное обеспечение можно разделить на две большие группы: обеспечение компьютеров-серверов и обеспечение компьютеров-клиентов.

Сетевое программное обеспечение компьютеров-серверов позволяет выполнять услуги, предоставляемые сервером клиентам, например хранение и пересылку ресурсов по запросу, поддержку работы электронной почты, все сервисы Всемирной паутины.

Пользователь работает с программным обеспечением компьютеров-клиентов:

- программами для навигации в сети интернет — *браузерами*, или *веб-обозревателями*, например Microsoft Internet Explorer, Mozilla Firefox, Opera;

- почтовыми клиентами, например Microsoft Outlook Express, The Bat!;
- программами для общения в сети, например ICQ, Skype.

Функционирование разнообразных компьютерных сетей и служб было бы невозможно без принятия единых стандартов, называемых *протоколами*. Протокол — это набор соглашений и правил, определяющих порядок обмена информацией в сети.

Сетевые протоколы определяют формат и правила передачи и приема сообщений различными устройствами компьютерной сети.

Наиболее важными протоколами в Интернете являются протокол TCP (Transmission Control Protocol — Протокол управления передачей) и протокол IP (Internet Protocol — межсетевой Интернет-протокол). Вместе они составляют семейство сетевых протоколов TCP/IP.

Протокол **TCP** предназначен для установки логического соединения между компьютерами клиентов и сервером, разделения сообщений на пакеты и контроля доставки этих пакетов к пункту назначения.

Файл делится на пакеты размером до 1,5 Кбайт, которые передаются независимо друг от друга. В месте назначения пакеты снова собираются в один файл, для чего в каждом пакете наряду с данными передается служебная информация.

Протокол **IP** задает маршрут движения пакетов: определяет адресацию при передаче информации и обеспечивает организацию транспортировки этой информации в пункты назначения по определенным маршрутам.

Все компьютеры, подключенные к сети, имеют свои **IP**-адреса. Каждый IP-адрес является уникаль-

ным и состоит из четырех восьмиразрядных двоичных чисел (байтов), разделенных точкой, например

**11000000.00110010.11010101.00010010.**

Часто их записывают с помощью четырех десятичных чисел (от 0 до 255), например **192.50.213.18**.

Три первых числа обозначают адрес сети (они выдаются специальными службами), а последнее — адрес (номер) компьютера в этой сети (присваивается администратором данной сети).

С помощью четырех байтов можно создать не более  $2^{32} = 4\ 294\ 967\ 296$  адресов. При такой адресации к Интернету может быть подключено чуть больше 4 миллиардов устройств. При бурном развитии Интернета довольно скоро адресов может не хватить. Поэтому уже разработана система адресации, включающая 6 байт ( $2^{48}$  адресов).

Цифровые IP-адреса неудобны для запоминания. Поэтому была создана другая система адресации — доменная система имен — DNS (Domain Name Service). Домен — это группа компьютеров, объединенных по некоторому признаку.

Доменный адрес складывается из символьных имен доменов. Число доменов и символов в их именах может быть различным. При записи доменных адресов применяется принцип последовательных уточнений справа-налево (как в обычном почтовом адресе: имя, дом, улица, город, страна), например:

**www.lisa.nora.les.by**

Домен 4-го уровня

Домен 3-го уровня

Домен 2-го уровня

Домен 1-го уровня

Домен первого уровня записывается в крайней правой части адреса. Это индекс страны или обозначение принадлежности к организации, например:

www.adu.by www.yandex.ru

www.kosht.com www.harward.edu

Левее перед индексом страны последовательно записывают имена доменов следующих уровней. Это могут быть названия провайдера, города, организации и, наконец, имя компьютера, например

www.bspu.unibel.by или www.mysite.bn.by.

Доменные имена первого уровня регистрируются международной службой имен, остальные — региональными службами.

Примеры индексов некоторых стран и организаций приведены в таблице.

Страна	Индекс	Организация	Индекс
Беларусь	.by	Коммерческая	.com
Россия	.ru	Образование	.edu
Украина	.ua	Бизнес	.biz
Польша	.pl	Информацион-ные сайты	.info
Германия	.de		
Италия	.it		
Китай	.cn	Разные органи-зации	.org
Япония	.jp		.net

Служба WWW использует гипертекстовую организацию разнообразной информации. Напомним, что под гипертекстом понимают способ организации информации с помощью гиперссылок. Гиперссылки позволяют связывать данный документ с различными объектами сети Интернет. Размещенный во Всемирной паутине гипертекстовый документ называют

**веб-страницей.** Совокупность связанных гиперссылками веб-страниц, объединенных общими целями и содержанием, называют **веб-сайтом**.

Все веб-страницы сайта, как правило, хранятся на одном компьютере, который имеет соответствующее программное обеспечение и называется **веб-сервером**. Иногда страницы сайта и даже отдельные объекты размещают на разных серверах.

Правила передачи веб-страниц по сети и их просмотра на экране компьютера определены в протоколе передачи гипертекста **HTTP** (HyperText Transfer Protocol). HTTP определяет формат и порядок обмена сообщениями между клиентом и сервером.

Для организации доступа к веб-странице одного доменного имени недостаточно. Необходимо знать не только адрес веб-сервера, но и расположение страницы на нем, т. е. путь к ней.

Полный адрес, состоящий из доменного имени компьютера и пути к файлу, называется **универсальным указателем ресурса URL** (Uniform Resource Locator). В общем случае URL имеет следующий вид:

**[протокол]://[доменное имя]/[путь к файлу].**

Например, URL-адрес

**http://www.bspu.unibel.by/phys/main.html**

содержит имя протокола **http**, затем после символов **://** указывается служба **www**, записывается доменное имя **bspu.unibel.by** и, наконец, путь к веб-странице (файлу) **/phys/main.html**.

Заметим, что при вводе адреса в адресной строке браузера имя протокола **http://** нередко опускают, его добавляет сам браузер.

Поскольку компьютер обрабатывает информацию только в цифровой форме, при пересылке со-

общений все URL-адреса преобразуются в цифровые IP-адреса. Это преобразование производит служба доменных имен. Поэтому IP-адреса часто сравнивают с почтовыми индексами, которые удобны для автоматической обработки.

- ?
- 1. Что понимают под сетевым протоколом?
- 2. Как записывается IP-адрес?
- 3. Что такое доменный адрес? Какие части он содержит?
- 4. Для чего предназначен URL? Какие части он содержит?

### Упражнения

1. Какие IP-адреса являются неправильными и почему?

- а) 85.192.102.111
- б) 131.1.12.45.17
- в) 137.102.205.18
- г) 34.567.78.207

2. Какие DNS-адреса могут быть неправильными и почему?

- а) www.school.by.newserver
- б) www.petr.unibel.by
- в) www.son76.com.pl
- г) www.ru.gov.ministerstvo

## § 25. Поиск информации в сети Интернет

В глобальной компьютерной сети Интернет размещены миллионы сайтов и миллиарды веб-страниц. Выполнение эффективного поиска информации в Интернете позволяет сократить затраты времени и повысить качество отбора нужной информации. Так, чаще всего пользователю нужен не весь найденный источник информации, а только отдельные фрагменты, например термины, определения, фотографии, научные и исторические справки.

*Информационные ресурсы* Интернета различаются как по формам представления информации, так и по методам доступа. Источником информации могут служить веб-сайты и отдельные страницы, тематические каталоги и поисковые системы, электронные библиотеки и файловые архивы, электронные журналы, материалы телеконференций и форумов, ленты новостей, каталоги интернет-магазинов. Способы поиска нужной информации также разнообразны. Выделяют два основных способа: *поиск по адресу* и *тематический поиск*.

Самый простой способ заключается в обращении к нужному ресурсу *по известному адресу*. Для этого достаточно ввести его в адресной строке браузера. С этим способом Вы уже знакомы. Его недостаток: необходимость знать адреса — обуславливает неэффективность поиска новой информации. Тем не менее следует накапливать адреса полезных и регулярно посещаемых сайтов и страниц, сохраняя их в папке **Избранное**.

**Тематический поиск** информации в Интернете выполняется с помощью *поисковых систем*, которые делятся на *тематические каталоги* и *поисковые машины*.

**Тематические каталоги** содержат *разделы*, информация в которых упорядочена *по категориям*, например областям человеческой деятельности: наука, политика, экономика, здоровье, семья. Они имеют древовидную структуру. На нижних уровнях располагаются ссылки на искомые ресурсы. Каждая ссылка обычно снабжена аннотацией.

Например, каталог системы **Google** содержит разделы: **Бизнес, Наука, Искусство, Компьютеры, Дом и семья, Общество, Страны и регионы** и др. (рис. 4.4).

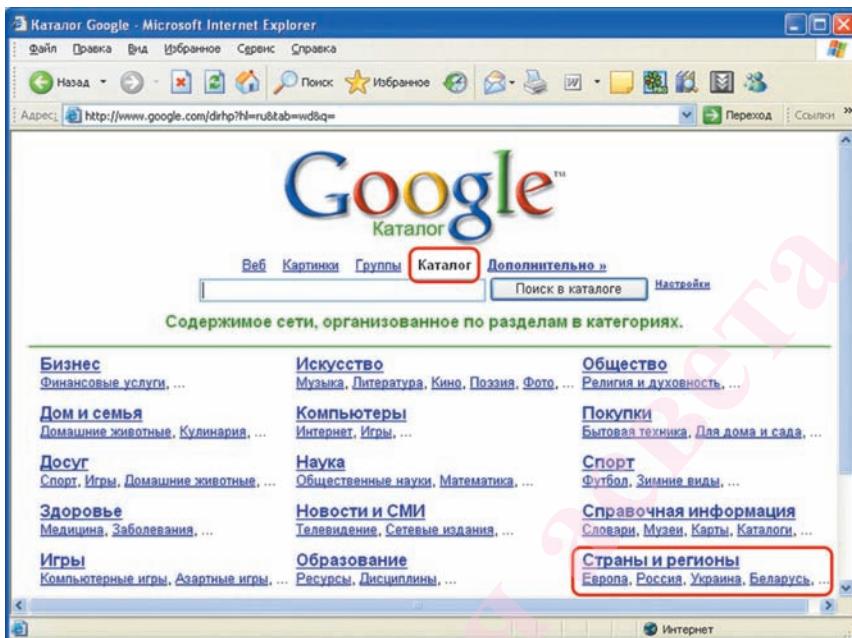


Рис. 4.4

Каталоги наполняют информацией сами разработчики при регистрации сайтов в поисковой системе, а также редакторы, просматривая новые сайты. Объем каталогов относительно невелик, но информация в них хорошо структурирована. Наиболее популярные каталоги **Rambler**, **Yahoo!**, **Google**.

**Пример 1.** С помощью каталога поисковой системы **Google** просмотреть ссылки с информацией о Беларуси в разделах **Образование**, **Наука** и **природа**.

Откроем в браузере каталог системы **Google**. Для этого наберем в адресной строке [www.google.com.by](http://www.google.com.by).

Выберем в разделе **Страны и регионы** (см. рис. 4.4) слово **Беларусь**.

Просмотрим содержимое раздела **Наука** и **природа**. По щелчку левой кнопкой мыши на заголовке

раздела **Наука и природа** (рис. 4.5) в браузере открывается страница со ссылками на веб-сайты данной тематики (рис. 4.6).

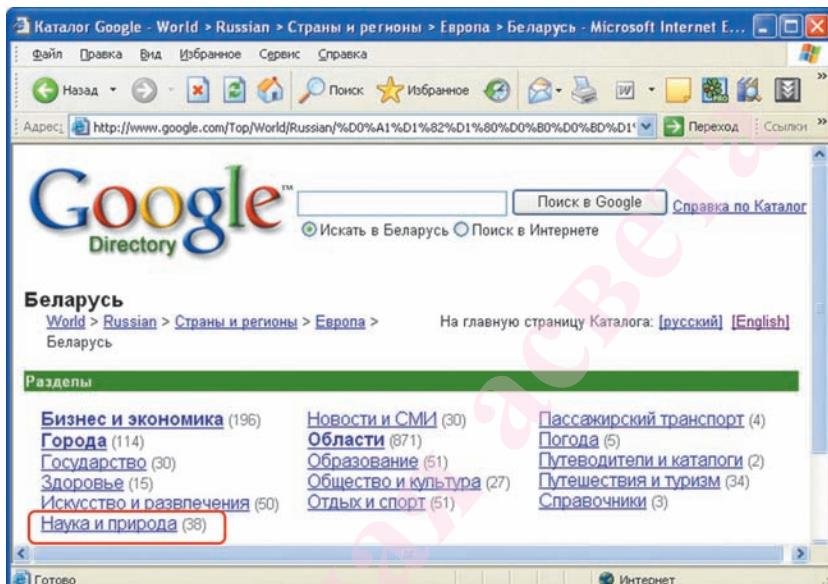


Рис. 4.5

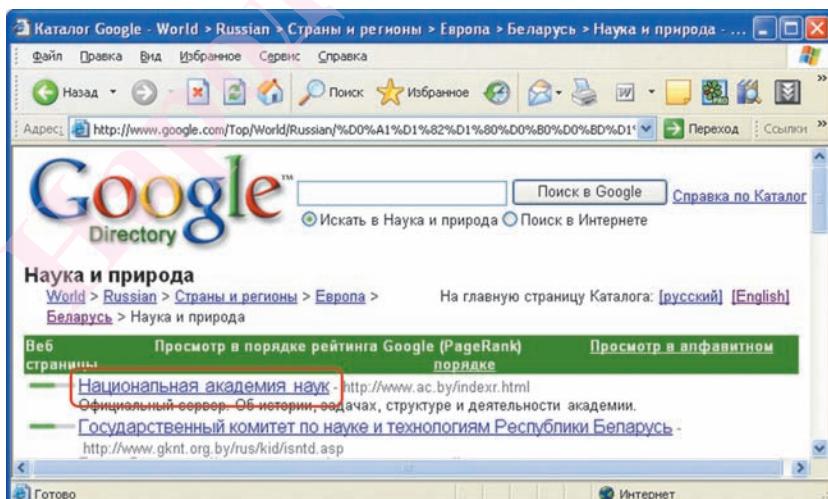


Рис. 4.6

Щелкнув на ссылке **Национальная академия наук**, можно попасть на главную страницу сайта Академии (рис. 4.7).

Автоматическое заполнение баз данных поисковых систем производят *поисковые машины* (search engines). Специальные программы-роботы (spiders — пауки) постоянно обходят Интернет в поисках новых страниц. Выполняется их обработка: индексирование (опись содержания, выделение ключевых слов), фильтрация (отсев повторяющихся ссылок), ранжирование (упорядочение по некоторому критерию, например цитируемости) и занесение в базу данных. Эта база данных все время расширяется

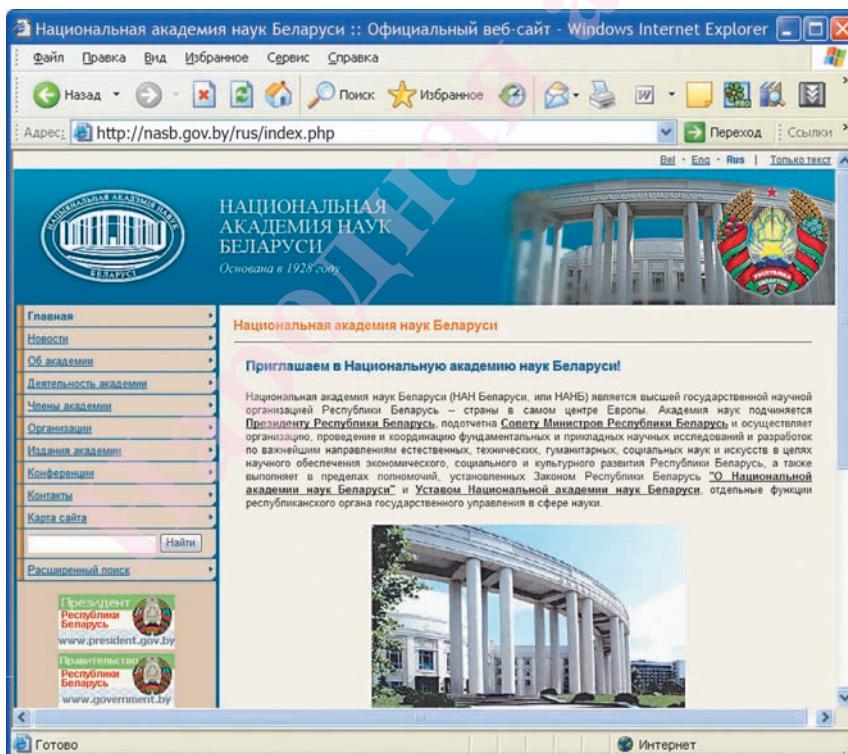


Рис. 4.7

и обновляется. В ней хранятся URL-адреса проиндексированных ресурсов.

Поиск информации в такой системе выполняется по текстовому запросу. Под **запросом** в поисковой системе понимается комбинация ключевых слов и команд, с помощью которых описывается искомый ресурс.

Поисковая система анализирует базу **данных** и выдает список ресурсов, удовлетворяющих запросу, с краткими сведениями о них. Благодаря индексации запрос выполняется очень быстро. Наиболее популярные системы с поисковыми машинами — **Яндекс**, **Google**.

Отметим, что современные поисковые системы используют и машины и тематические каталоги.

GOOGLE www.google.com	YAHOO www.yahoo.com
Яндекс www.yandex.ru	RAMBLER www.rambler.ru
APORT www.aport.ru	ALL.BY www.all.by

Эффективность поиска в значительной степени определяется умением корректно формулировать запрос.

**Пример 2.** С помощью поисковой системы **Яндекс** найти информацию по теме **Почтовые марки**.

Откроем главную страницу поисковой системы **Яндекс** в браузере, набрав в адресной строке [www.yandex.ru](http://www.yandex.ru).

В строке запроса наберем ключевые слова **почтовые марки** (рис. 4.8) и щелкнем левой кнопкой мыши на кнопке **Найти** (или нажмем клавишу Enter).

После выполнения запроса на экране появится список найденных ссылок. В верхней его части под строкой запроса представлена суммарная информация о результатах поиска, например **Нашлось 8 млн страниц** (см. рис. 4.8).

Подходящие ссылки можно выбрать, просматривая комментарии. Заметим, что окончания слов в найден-

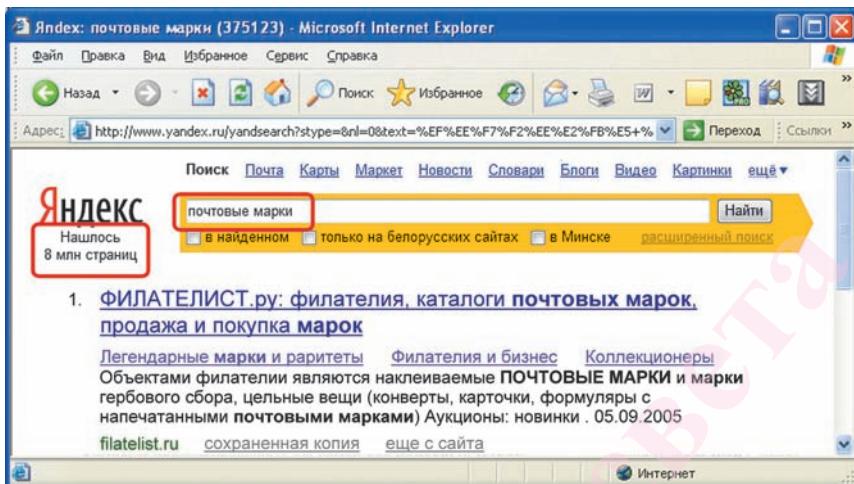


Рис. 4.8

ных ресурсах могут отличаться от указанных в запросе (например, **почтовых марок, почтовая марка**).

Конечно, просмотреть такое огромное количество страниц невозможно. Уменьшить количество предлагаемых ссылок можно, ограничив регион поиска. Для этого установим флажок **только на белорусских сайтах** (рис. 4.9).

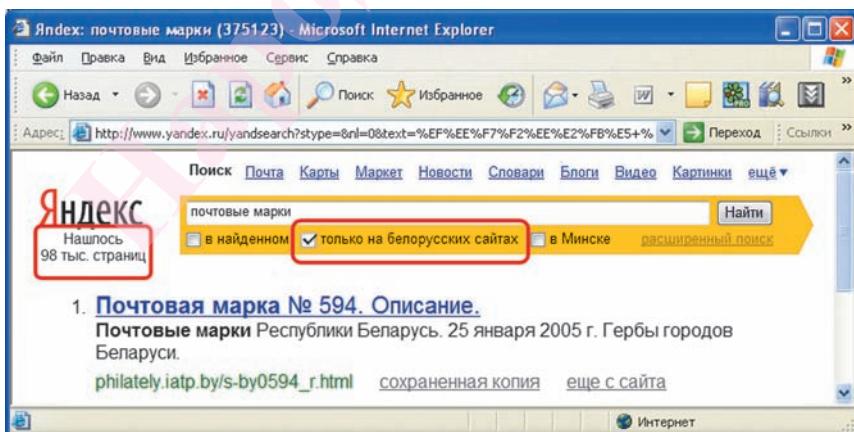


Рис. 4.9

Наибольшая эффективность поиска достигается при использовании в запросах логических операций.

Их обозначение в разных поисковых системах может несколько отличаться.

В системе **Яндекс**, например, соединив слова запроса символом & можно ограничить поиск только теми страницами, где эти слова находятся в пределах одного предложения. Для исключения страниц, где встречается указанное слово, слева перед ним без пробела набирают знак «минус». Более подробная информация о языке запросов содержится в разделе **Справка** каждой поисковой системы.

**Пример 3.** С помощью поисковой системы Яндекс найти информацию по теме **Почтовые марки Беларусь**.

В строке запроса наберем **почтовые марки & Беларусь** (рис. 4.10). Будут отобраны ссылки на те

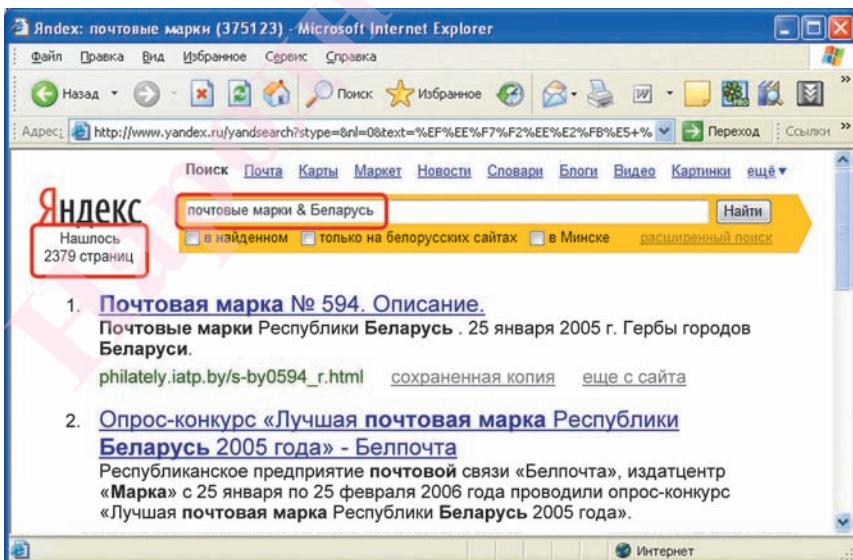


Рис. 4.10

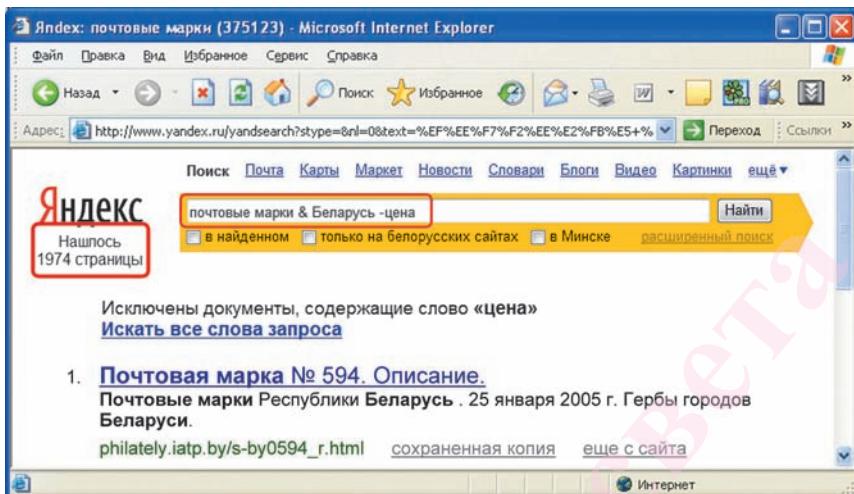


Рис. 4.11

страницы, где встречаются все три слова: **почтовые, марки и Беларусь** (см. рис. 4.10).

Можно еще больше сузить поиск, исключив не имеющую для нас значения информацию. Пусть, например, нас не интересует цена почтовых марок.

В этом случае введем **почтовые марки & Беларусь -цена**. Количество найденных страниц и сайтов еще больше сократится (рис. 4.11).

- ? 1. Какие источники информации в Интернете Вам известны?  
 2. В чем недостатки поиска информации по адресу?  
 3. Какие действия производят поисковые машины при автоматическом заполнении баз данных?  
 4. Что понимается под запросом в поисковой системе?

### *Упражнения*

1. Проведите поиск информации в одном из тематических каталогов на тему:

- a) Беловежская пуща;

б) озеро Нарочь;

в) спорт в Беларуси.

2. С помощью поисковой системы найдите:

а) кулинарные рецепты различных блюд;

б) время отправления поездов Минск — Брест;

в) сведения о заданной модели мобильного телефона.

3. Найдите ссылки на источники информации, содержащие ответы на вопросы:

а) какой прогноз погоды в Беларуси на завтра;

б) кто и когда построил Мирский замок;

в) как называлась первая белорусская газета;

г) как называется самая высокая гора на Земле.

## § 26. Сохранение информации из сети Интернет

Поиск информации в Интернете обычно заканчивается ее сохранением на диске.

Сохранение веб-страницы в браузере Internet Explorer выполняется с помощью меню **Файл → Сохранить как**. При этом в списке **Тип файла** диалогового окна **Сохранение веб-страницы** предлагается выбрать формат сохраняемого документа (рис. 4.12):

- Веб-страница, полностью (\*.htm, \*.html);
- Веб-архив, один файл (\*.mht);
- Веб-страница, только HTML (\*.htm, \*.html);
- Текстовый файл (\*.txt).

Документы, сохраняемые в первых трех форматах, в последующем можно просматривать в браузере без подключения к сети Интернет (в автономном режиме).

**Пример 1.** Сохранить на диске открытую веб-страницу Гавайские острова (рис. 4.13) разными способами.

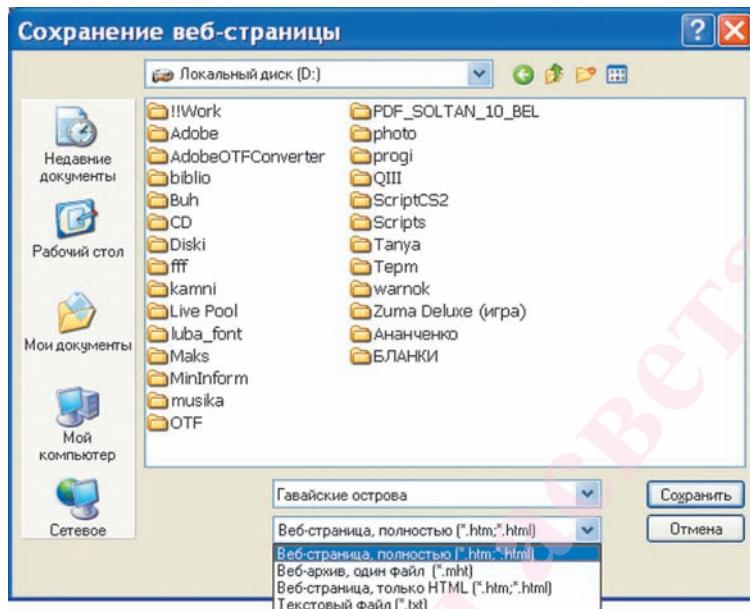


Рис. 4.12

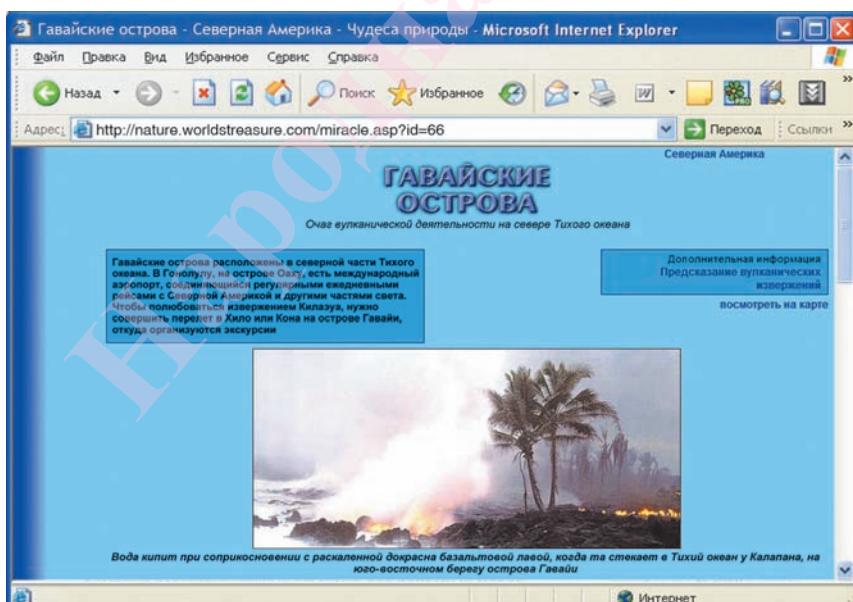


Рис. 4.13

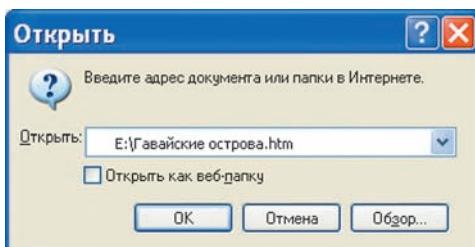


Рис. 4.14

Выберем тип файла **Веб-страница, полностью**. Выберем папку и введем имя файла. Страница сохранится в формате **.html**, при этом рисунки и другие объекты сохранятся в папке с тем же именем и расширением **.files**).

В дальнейшем сохраненную страницу можно открыть и просмотреть в браузере с помощью меню **Файл → Открыть**. Путь к ней указывается в диалоговом окне **Открыть** с помощью кнопки **Обзор** (рис. 4.14).

Выберем тип файла **Веб-архив, один файл**. Страница со всеми объектами будет упакована в один архивный файл с расширением **.mht**.

Вид страницы, сохраненной в этих двух форматах, при просмотре в автономном режиме не отличается от исходного.

Выберем тип файла **Веб-страница, только HTML**. В этом случае сохранится только текст страницы в формате **.html**, а оформление, рисунки и другие объекты — нет. Вид страницы изменится (рис. 4.15, сравните с рис. 4.13).

Наконец выберем тип файла **Текстовый файл**. В этом случае сохранится только текст в формате **.txt**, который можно просматривать и редактировать с помощью текстового редактора **Блокнот**.

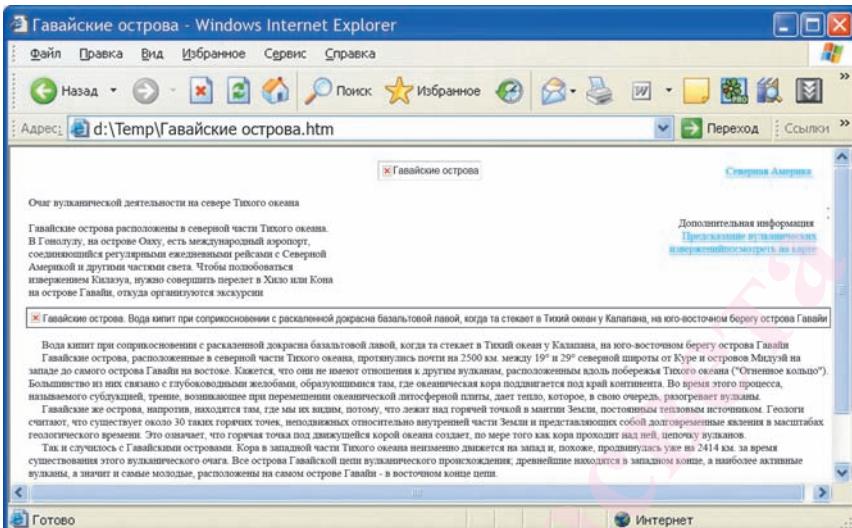


Рис. 4.15

Иногда требуется сохранить или скопировать в другие документы отдельные фрагменты веб-страницы, например рисунки, фрагменты текста.

### **Пример 2.** Сохранить рисунок с веб-страницы Гавайские острова.

Щелкнем правой кнопкой мыши на изображении.

В открывшемся контекстном меню выберем **Сохранить рисунок как**.

Выберем папку и введем имя файла.

Выделенный объект можно скопировать в буфер обмена. Для этого в контекстном меню надо выбрать пункт **Копировать**. Из буфера обмена объект может быть вставлен, например, в документ Word.

### **Пример 3.** Скопировать в документ Word рисунок и подпись к нему с веб-страницы Гавайские острова.

Выделим рисунок и подпись. С помощью меню **Правка → Копировать** или комбинации клавиш

Ctrl + C скопируем выделенный фрагмент в буфер обмена.

Вставим его из буфера обмена в документ Word с помощью меню **Правка** → **Вставить** или комбинации клавиш Ctrl + V.

- ? 1. Какими способами веб-страница может быть сохранена на диске?
2. С помощью каких действий в документ Word можно скопировать с веб-страницы:
- фрагмент текста;
  - рисунок?

### *Упражнение*

Откройте с помощью браузера одну из известных Вам поисковых систем. Найдите с помощью этой системы и сохраните на диске Вашего компьютера:

- карту Беларуси;
- портрет и биографию известного ученого (писателя, музыканта, художника) по указанию учителя;
- текст по истории города Полоцка (Турова, Несвижа);
- изображения обитающих в Беларуси животных (птиц, рыб);
- текст гимна Республики Беларусь;
- расписание движения автобусов (поездов, самолетов).

## **§ 27. Национальные информационные ресурсы**

Годом рождения белорусского сегмента сети Интернет можно считать 1991, когда была основана первая национальная компьютерная сеть BAS-NET (Byelorussian Academy of Sciences Network),

объединившая 10 научных учреждений национальной Академии наук. В 1993 г. создается государственная сеть **Белпак**. Начинается работа по организации национальной образовательной сети **Юнибел** (Unibel). В 1994 г. зарегистрирован первый белорусский домен **minsk.by**.

Быстрое развитие Белорусского сегмента Интернета начинается с 1996 г. Если на начало 1996 г. в Беларуси было 4 домена (ac.by, bas-net.by, minsk.by, unibel.by), то к середине 1997 г. — уже 28. Начинает предоставлять услуги доступа в Интернет сеть **Белпак**. Растет и посещаемость информационных ресурсов белорусской адресной зоны **.by**. Если за 1996 г. домены **.by** запрашивались менее 500 раз, то за 1997 г. — уже более 3200.

К середине 2009 г. в Республике работают более 180 операторов связи, предоставляющих услуги доступа в Интернет. Число пользователей Сети в Беларуси превышает 3 млн. Количество информационных ресурсов в зоне **.by** стремительно увеличивается. Национальный каталог белорусских сайтов [www.all.by](http://www.all.by) содержит более 30 000 ссылок, сгруппированных в 8 разделах и 65 рубриках. Примеры некоторых наиболее значимых и информативных национальных интернет-ресурсов приведены в Приложении 2.

В настоящее время все органы государственной власти от республиканского до районного масштаба имеют свои сайты, на которых наряду с другой информацией размещаются официальные документы соответствующего уровня. Так, на сайтах Совета Республики [www.sovrep.gov.by](http://www.sovrep.gov.by) и Палаты Представителей [www.house.gov.by](http://www.house.gov.by) размещены законодательные акты, на сайте Президента [www.president.gov.by](http://www.president.gov.by)

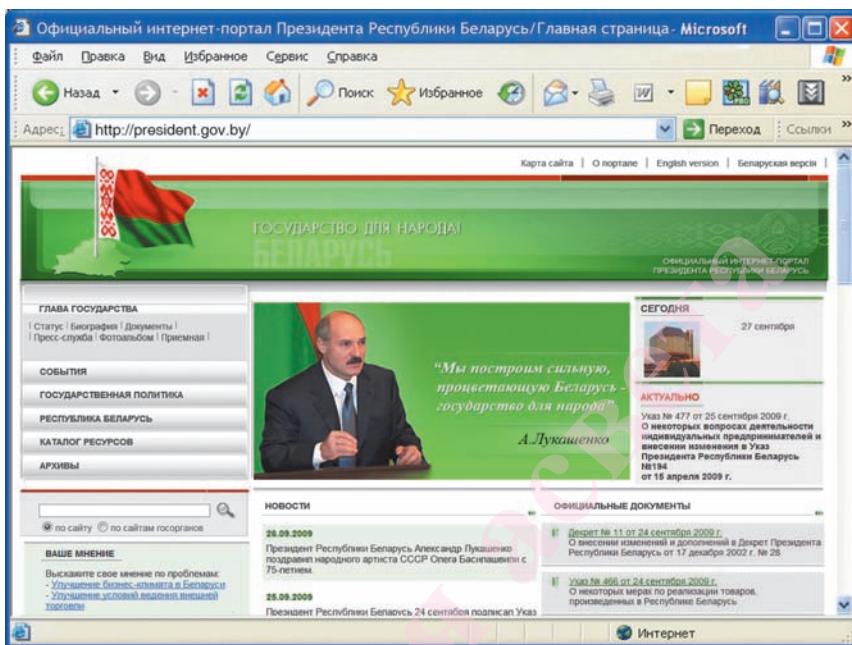


Рис. 4.16

(рис. 4.16) — подписанные им указы, а на сайте Министерства образования [www.minedu.unibel.by](http://www.minedu.unibel.by) — документы, относящиеся к сфере образования.

Важную роль в распространении правовой информации играют Национальный правовой интернет-портал Республики Беларусь [www.pravo.by](http://www.pravo.by) и Национальный центр правовой информации <http://ncpi.gov.by>.

Всесторонняя информация о нашей стране представлена на сайте Беларусь (**Belarus**) [www.belarus.by](http://www.belarus.by) (рис. 4.17). В четырех разделах сайта (О Беларуси, Туризм, Бизнес, Власть) информация сгруппирована по рубрикам. Приведем некоторые из них: География, Население, Природа, Климат и погода, История, Религия, Образование, Культура, Кухня, Спорт.



Рис. 4.17

Большое количество сайтов посвящено природе и истории родного края. Среди них сайты: **Дикая природа Беларуси** <http://www.wild.by>, **Города Беларуси** <http://belaruscity.net>, **Радзіма** <http://www.radzima.org>.

Огромный материал содержит сайт **История Беларуси IX—XVIII вв. Первоисточники**, размещенный по адресу <http://starbel.narod.ru>. На нем представлены археологические находки (оружие, предметы прикладного искусства), произведения изобразительного искусства (живопись, скульптура, иконы, фрески), книжные гравюры и иллюстрации. Особый интерес представляет портретная галерея деятелей Великого княжества Литовского.

Национальный геральдический портал [www.geraldika.by](http://www.geraldika.by) содержит символы, флаги и гербы городов и старинных родов. Там же размещена интерактивная карта памятных мест Беларуси.

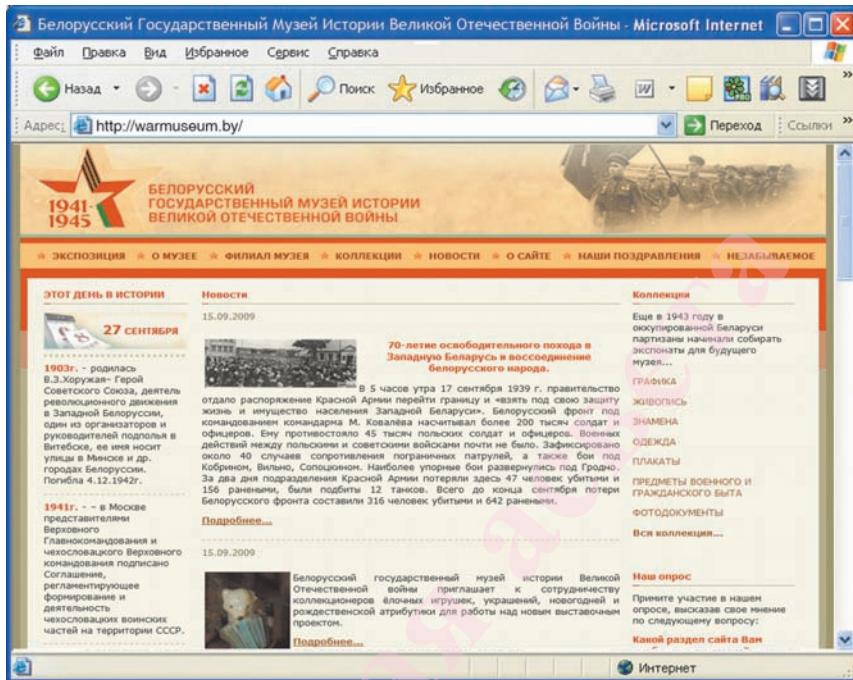


Рис. 4.18

Сайт Белорусского государственного музея истории Великой Отечественной войны <http://warmuseum.by> (рис. 4.18) знакомит посетителей с малоизвестными материалами, а также фотографиями, документами и реликвиями: военной формой, оружием, предметами военного быта, картами, плакатами, листовками, календарем памятных дат периода войны.

Сайт Национального художественного музея <http://artmuseum.by> приглашает с помощью эффектов анимации на виртуальную экскурсию (рис. 4.19).

Весьма динамично развивается Белорусский сектор Интернета в сфере культуры. Интересны и по-

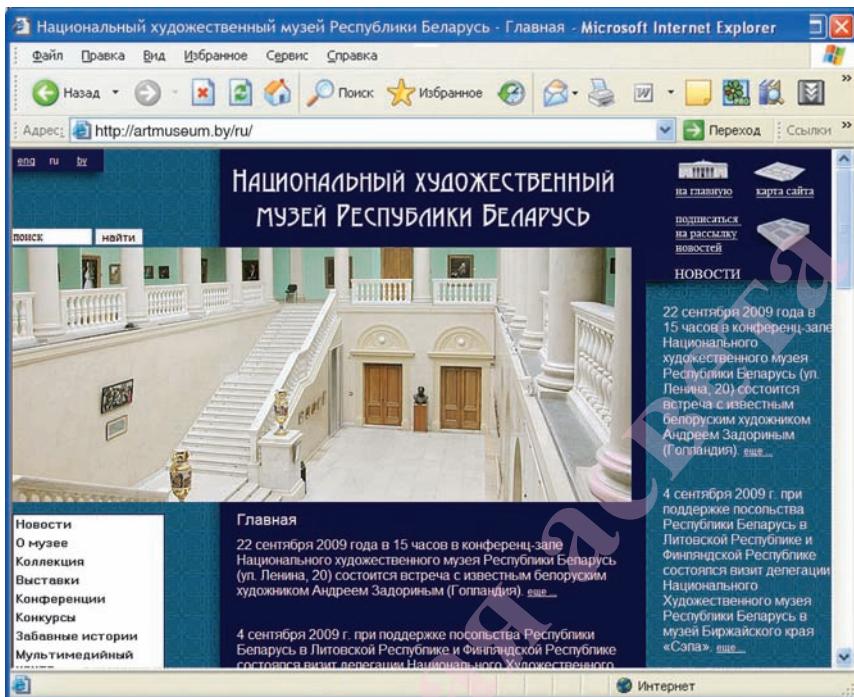


Рис. 4.19

знавательны сайты по искусству, музыке, народному творчеству, например сайт Международного фестиваля искусств «Славянский базар» в Витебске <http://festival.vitebsk.by> или сайт музея старинных народных ремесел и технологий «Дудутки» <http://dudutki.by>.

На сайте Национального академического театра имени Янки Купалы [www.kupala-theatre.by](http://www.kupala-theatre.by) можно просмотреть фотографии актеров и сцен из спектаклей, познакомиться с предстоящими премьерами (рис. 4.20).

Большую помощь в информационном обслуживании образования оказывают сайты библиотек. Эти сайты предоставляют ряд услуг:

- доступ к электронному каталогу и поиск в нем;

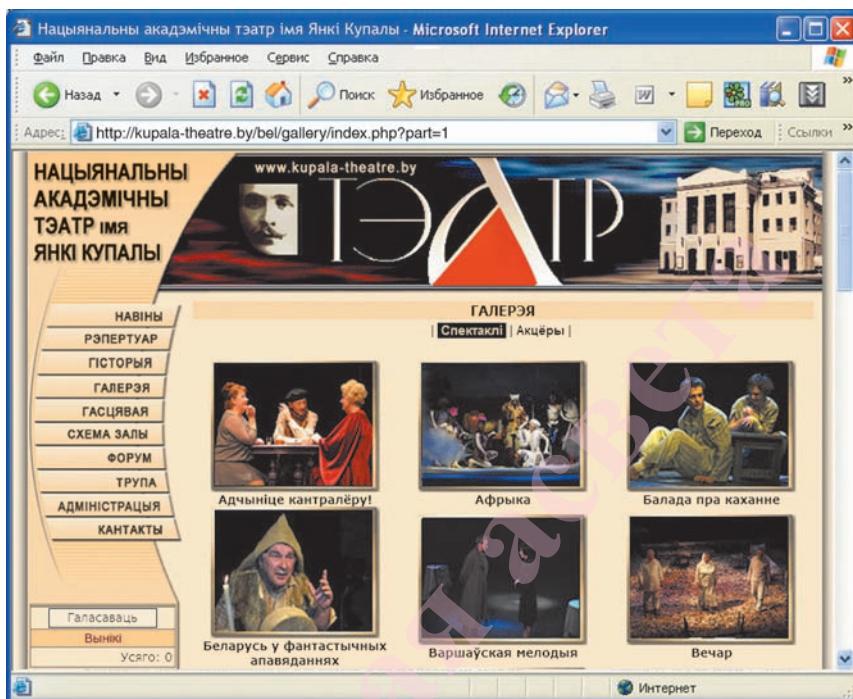


Рис. 4.20

- просмотр электронных документов, журналов, газет, книг и др.;
- автоматический заказ требуемой литературы;
- информирование пользователей о конференциях, семинарах, выставках.

Полезным сервисом является поиск литературы по электронным каталогам.

**Электронный каталог** Национальной библиотеки Беларусь является информационно-поисковой системой. Он постоянно пополняется и отражает старопечатные и редкие издания; книги, изданные в Беларусь и странах СНГ, книги зарубежных издательств на иностранных языках, начиная с 2000 г.; документы международных организаций, нотные и изобрази-

тельные издания, авторефераты диссертаций, карты, атласы, аудио- и видеодокументы, электронные ресурсы и др.

Поиск документов в каталоге возможен в следующих режимах: **Базовый поиск**, **Поиск по видам документов**, **Поиск по словарям** (рис. 4.21).

Базовый поиск позволяет находить документы как по всем элементам библиографической записи, так и по их сочетанию (автор, заглавие, тема и т. д.). Результаты поиска могут также ограничиваться видом документа, языком и годом издания.

**Пример 1.** Выполнить базовый поиск книг Валерия Быкова на белорусском языке, изданных в Беларуси и других странах в период с 2008 по 2009 г. и хранящихся в библиотеке.

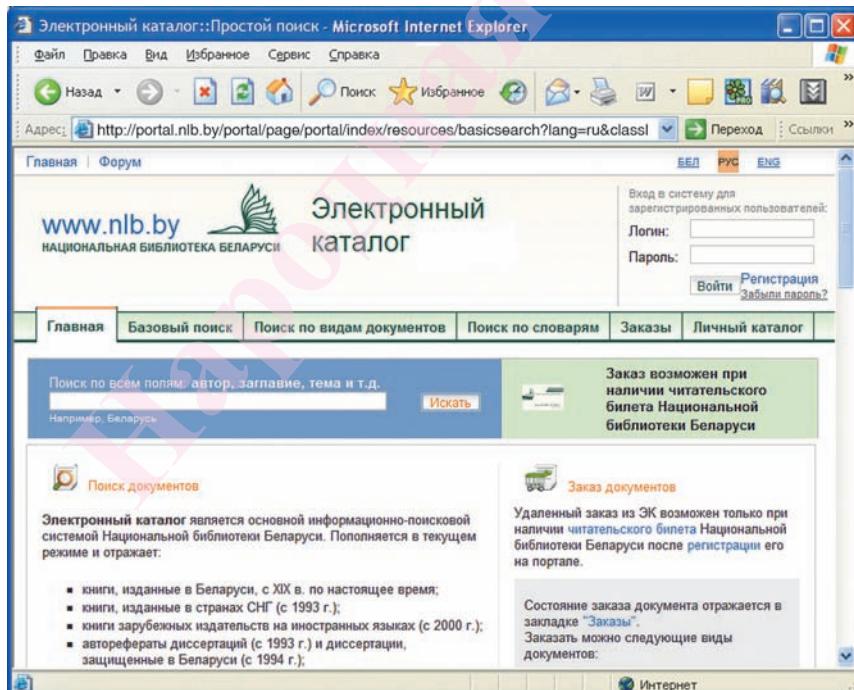


Рис. 4.21

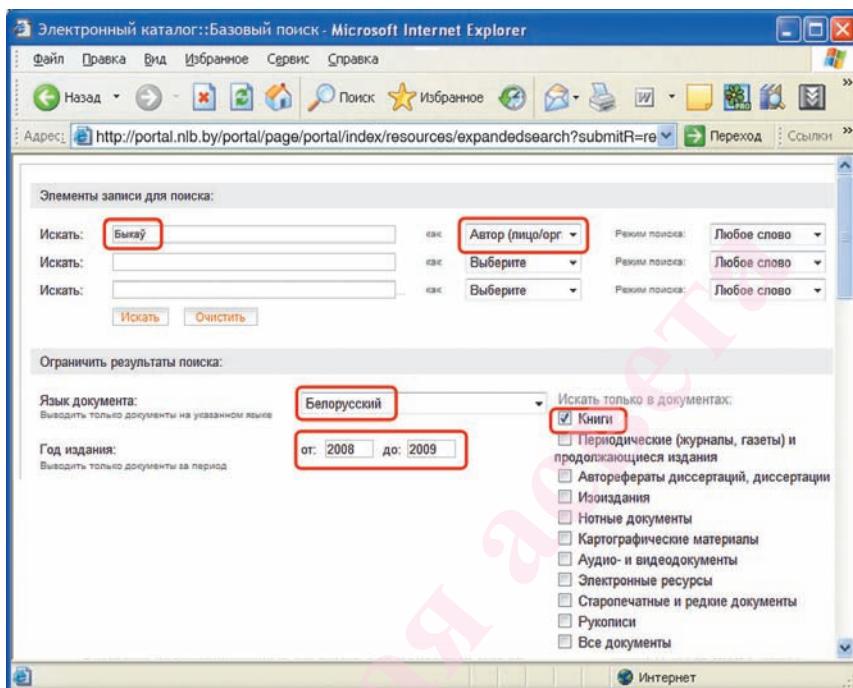


Рис. 4.22

Для того чтобы найти книги, удовлетворяющие этим требованиям, в специальной форме базового поиска электронного каталога следует ввести или выбрать элементы, как показано на рисунке 4.22.

В результате поиска мы получим список источников, представленный на рисунке 4.23.

**Поиск по видам документов** позволяет находить документы с учетом их вида: книги, нотные издания, диссертации, электронные ресурсы и др.

**Пример 2.** С помощью Поиска по видам документов найти изобразительные документы (Изодокументы) белорусского художника Михаила Савицкого за период с 2005 по 2009 г.

Заполним запрос, как показано на рисунке 4.24. Результат поиска представлен на рисунке 4.25.

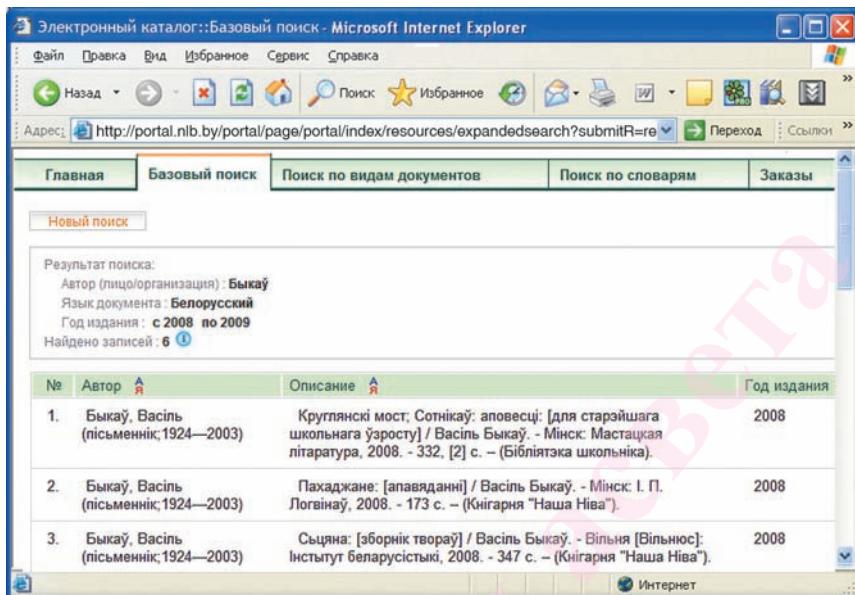


Рис. 4.23

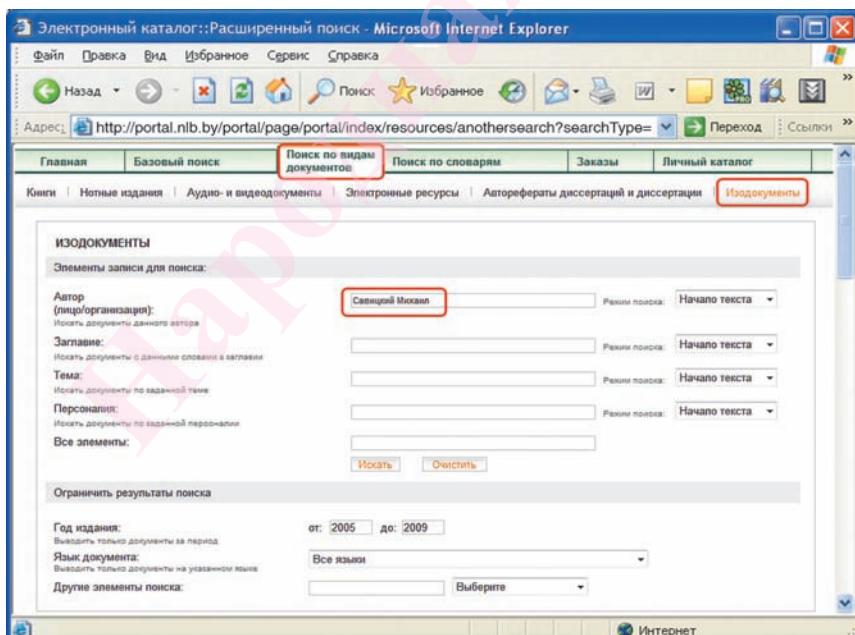


Рис. 4.24

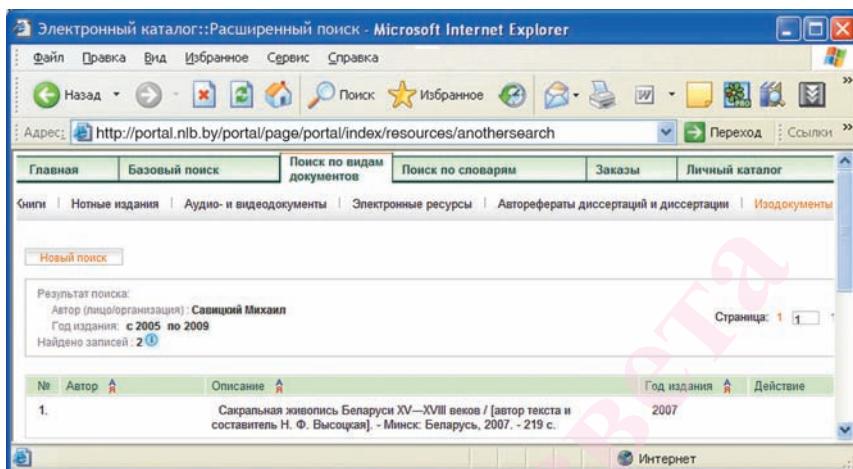


Рис. 4.25

**Поиск по словарям** предусматривает поиск документов на основе специальных записей, которые называют авторитетными записями (АЗ). Исходным элементом АЗ является принятый заголовок объекта: лица, организации, географического объекта, предмета.

- ? 1. Приведите примеры национальных информационных ресурсов.
- 2. Какие возможности предоставляют пользователям сайты библиотек?

### ***Упражнения***

1. Проведите поиск национальных информационных ресурсов на тему:
  - а) природа родного края;
  - б) история родного края (города, района).
2. С помощью каталогов белорусских сайтов (см.

Приложение 2) проведите поиск ресурсов на тему:

- а) библиотеки;
- б) театры;
- в) народные ремесла;
- г) спорт.

## § 28. Образовательные ресурсы сети Интернет

К образовательным ресурсам относят материалы учебно-познавательного характера. В сети Интернет накоплено огромное количество образовательных ресурсов, которые можно использовать на занятиях, при самостоятельной работе, а также для самообразования. Они размещаются на сайтах учебных заведений, библиотек, музеев, театров, а также отдельных разработчиков: преподавателей, работников науки и культуры, школьников и студентов.

Примеры некоторых образовательных интернет-ресурсов приведены в Приложении 3.

Широкое применение находят ресурсы справочного характера: электронные энциклопедии, словари и справочники. Большой популярностью пользуется Википедия — свободная энциклопедия <http://ru.wikipedia.org> (рис. 4.26). В ней статьи создаются и редактируются самими пользователями, что позволяет весьма оперативно добавлять и уточнять термины.

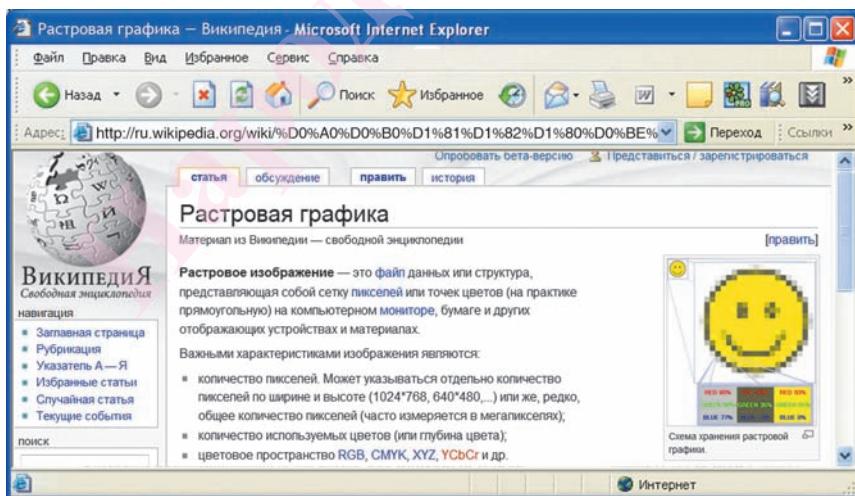


Рис. 4.26

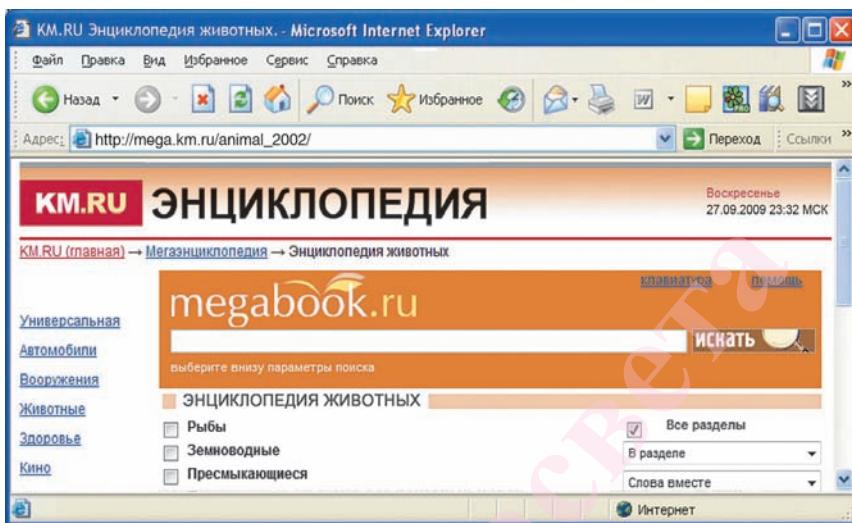


Рис. 4.27

Высоким качеством и развитой графикой отличаются материалы порталов **Кирилл и Мефодий** [www.km.ru](http://www.km.ru), энциклопедии и словари <http://mega.km.ru/> (рис. 4.27), а также виртуальные уроки сайта **Школьный клуб** <http://school-club.ru/vschool> (рис. 4.28).

На справочно-информационном интернет-портале **Русский язык** <http://www.gramota.ru> можно не только воспользоваться орфографическими и терминологическими словарями, но и в интерактивном режиме проверить правописание (рис. 4.29).

Особый интерес вызывают интерактивные модели и анимации процессов и явлений, которые невозможно воспроизвести в классе или дома, а также виртуальные лабораторные работы. С ними можно познакомиться на интернет-портале **Открытый колледж** [www.college.ru](http://www.college.ru), который осуществляет дистанционное обучение школьников по различным учеб-

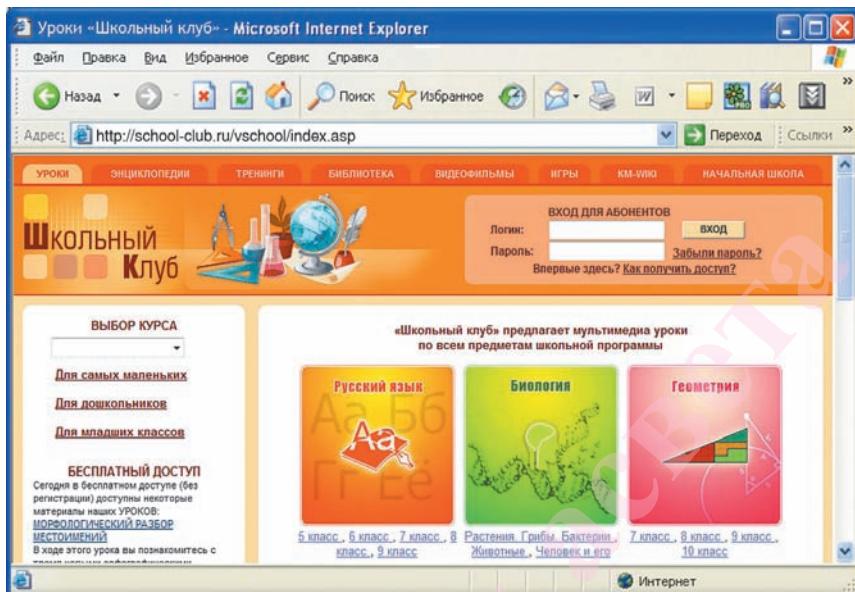


Рис. 4.28

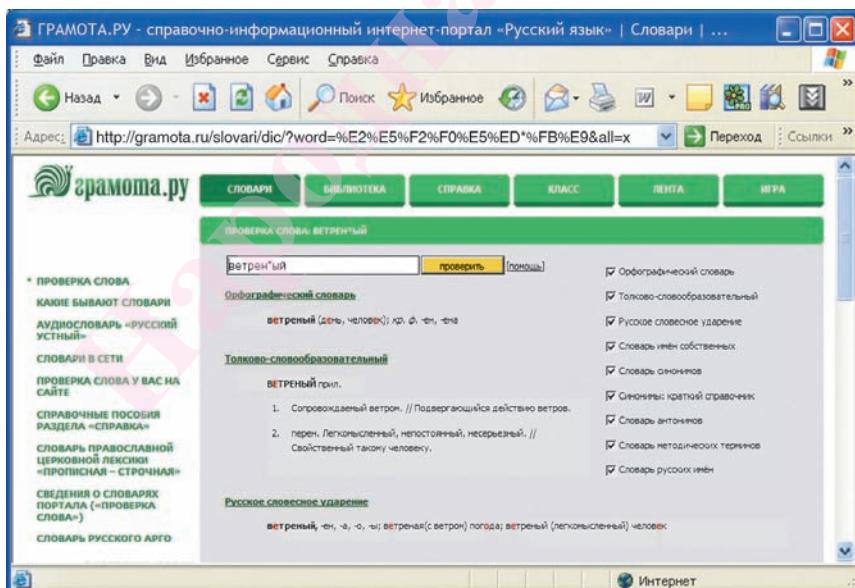


Рис. 4.29



Рис. 4.30

ным предметам: **Математика** <http://mathematics.ru>, **Физика** <http://physics.ru>, **Биология** <http://biology.ru>, **Химия** <http://chemistry.ru> (рис. 4.30).

Многие образовательные сайты содержат учебные пособия, тесты, рефераты, экзаменационные и олимпиадные задачи и их решения, а также разнообразные материалы познавательного характера.

Использование материалов образовательных сайтов позволяет расширить и углубить знания по различным учебным предметам и повысить общекультурный уровень.

Для поиска требуемых образовательных ресурсов можно использовать известные поисковые системы. Наиболее просто использовать специализированные тематические каталоги, а также коллекции ссылок, например **Республиканский информаци-**

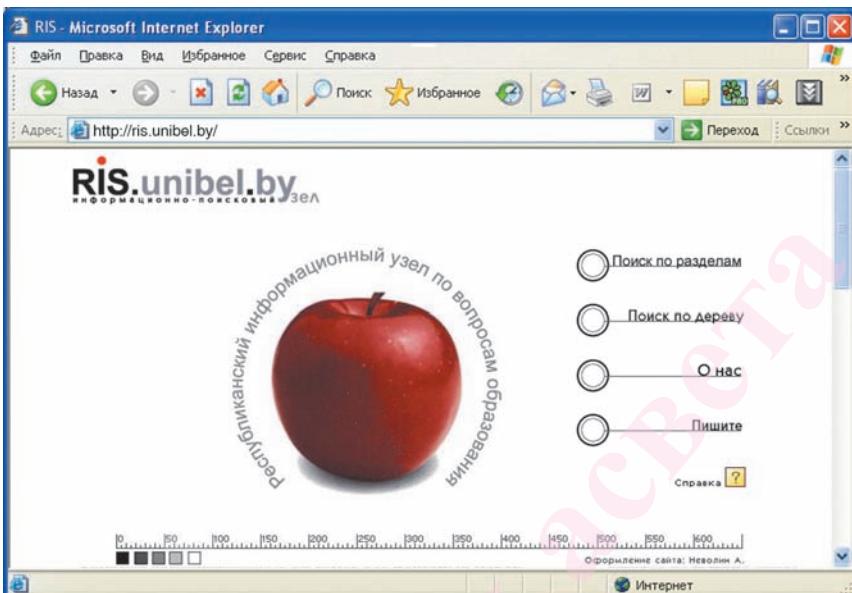


Рис. 4.31

**онный поисковый узел по вопросам образования** <http://ris.unibel.by> в Республике Беларусь (рис. 4.31), **Единое окно доступа к образовательным ресурсам** <http://window.edu.ru>, **Единая коллекция цифровых образовательных ресурсов** <http://school-collection.edu.ru> в России.

- ? 1. Какие ресурсы сети Интернет относятся к образовательным?  
2. Какие материалы могут содержать образовательные сайты?

### **Упражнение**

Используя адреса образовательных сайтов (см. Приложение 3) и поисковые системы, найдите и сохраните на диске Вашего компьютера материалы 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. Максім Багдановіч.

**Русская литература:** Слово о полку Игореве. Литературное творчество М. В. Ломоносова. Г. Р. Державин. А. С. Грибоедов. 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.

**Всемирная история:** Наполеоновские войны. Франция: от Июльской революции до Парижской Коммуны. Гражданская война в США. Отечественная война 1812 года. Декабристы. Народничество. Революция 1905—1907 гг. Первая мировая война.

**Гісторыя Беларусі:** Беларусь у перыяд Айчыннай вайны 1812 г. Гарады і мястэчкі Беларусі ў XIX ст. Маствацтва XIX ст. Тэатр і музыка XIX ст. М. К. Агінскі. Шляхецкае паўстанне 1863—1864 гг. у Беларусі. К. Каліноўскі. Рэвалюцыйныя падзеі 1905 г. у Беларусі.

**География:** Региональный обзор земного шара. Природные условия. Рельеф. Климат. Почвы. Растительный и животный мир. Население. Промышленность. Сельское хозяйство. Экологические проблемы. Охрана природы.

**Биология:** Заповедники и национальные парки Беларуси. Человек и окружающая среда. Биосфера. Влияние различных факторов среды на человека.

Загрязнение окружающей среды. Глобальное потепление. Парниковый эффект.

**Математика:** История числа  $\pi$ . Длина окружности и дуги. Площадь круга и его сектора. Системы уравнений с двумя неизвестными. Квадратные неравенства. Графики функций. Арифметическая и геометрическая прогрессии. Замечательные точки треугольника. Теоремы синусов и косинусов. Решение треугольников. Правильные многоугольники.

**Физика:** Система отсчета. Относительность движения. Равномерное и равноускоренное движение. Сложение скоростей. Движение по окружности. Центростремительное ускорение. Силы в природе. Законы Ньютона. Закон всемирного тяготения. Невесомость. Упругие деформации. Трение в природе и технике. Законы сохранения импульса. Реактивное движение. Работа и энергия. Законы сохранения энергии. Простые механизмы.

**Химия:** Кислород в природе. Сера. Азот. Фосфор. Углерод. Кремний. Химическое строение органических соединений. Углеводороды. Переработка нефти. Кислородсодержащие органические соединения. Применение жиров. Синтетические моющие средства. Углеводы. Азотсодержащие органические соединения. Аминокислоты. Синтетические высокомолекулярные соединения. Применение полимеров. Белки.

## ПРИЛОЖЕНИЯ

### ПРИЛОЖЕНИЕ 1

#### Кодовая таблица символов

Код	Символ	Код	Символ	Код	Символ	Код	Символ
0		32		64	@	96	`
1		33	!	65	А	97	а
2		34	"	66	В	98	б
3	♥	35	#	67	С	99	с
4	♦	36	\$	68	Д	100	д
5	♣	37	%	69	Е	101	е
6	♠	38	&	70	Ф	102	ф
7	•	39	'	71	Г	103	г
8		40	(	72	Х	104	х
9		41	)	73	І	105	і
10		42	*	74	҃	106	ј
11	♂	43	+	75	К	107	к
12	♀	44	,	76	Л	108	л
13		45	-	77	М	109	м
14	¤	46	.	78	Н	110	н
15		47	/	79	О	111	о
16	►	48	0	80	Р	112	р
17	◀	49	1	81	Q	113	қ
18	↑	50	2	82	Р	114	ր
19	!!	51	3	83	С	115	ս
20	¶	52	4	84	Т	116	տ
21	§	53	5	85	Ւ	117	ւ
22	■	54	6	86	Վ	118	վ
23		55	7	87	Ո	119	ո
24	↑	56	8	88	Խ	120	խ
25	↓	57	9	89	Յ	121	յ
26	→	58	:	90	Զ	122	զ
27	←	59	;	91	[	123	{
28	∟	60	<	92	\	124	
29	↔	61	=	93	]	125	}
30	▲	62	>	94	^	126	~
31	▼	63	?	95	—	127	◊

## Продолжение

Код	Символ	Код	Символ	Код	Символ	Код	Символ
128	А	160	а	192	Л	224	р
129	Б	161	б	193	Л	225	с
130	В	162	в	194	Т	226	т
131	Г	163	г	195	Г	227	у
132	Д	164	д	196	—	228	Ф
133	Е	165	е	197	+	229	х
134	Ж	166	ж	198	Г	230	ц
135	З	167	з	199		231	ч
136	И	168	и	200	Л	232	ш
137	Й	169	й	201	Г	233	щ
138	К	170	к	202	Л	234	ъ
139	Л	171	л	203	Г	235	ы
140	М	172	м	204	Г	236	ь
141	Н	173	н	205	=	237	э
142	О	174	о	206	Г	238	ю
143	П	175	п	207	Л	239	я
144	Р	176	р	208	Л	240	Ё
145	С	177	с	209	Г	241	ё
146	Т	178	т	210	Г	242	Є
147	У	179	у	211	Л	243	є
148	Ф	180	ф	212	Л	244	Ї
149	Х	181	х	213	Г	245	ї
150	Ц	182	ц	214	Г	246	Ӯ
151	Ч	183	ч	215	Г	247	ӹ
152	Ш	184	ш	216	+	248	◦
153	Щ	185	щ	217	Г	249	·
154	Ъ	186	ъ	218	Г	250	·
155	Ы	187	ы	219	■	251	√
156	Ь	188	ь	220	■	252	№
157	Э	189	э	221	■	253	¤
158	Ю	190	ю	222	■	254	■
159	Я	191	я	223	■	255	

## ПРИЛОЖЕНИЕ 2

## Национальные информационные ресурсы

Администрация Президента Республики Беларусь	<a href="http://www.president.gov.by">www.president.gov.by</a>
Совет Республики Национального собрания Республики Беларусь	<a href="http://www.sovrep.gov.by">www.sovrep.gov.by</a>
Палата представителей Национального собрания Республики Беларусь	<a href="http://www.house.gov.by">www.house.gov.by</a>
Совет Министров Республики Беларусь	<a href="http://www.government.by">www.government.by</a>
Официальные документы. Декреты, Указы, Постановления	<a href="http://www.law.sb.by">www.law.sb.by</a>
Город Минск: история и современность	<a href="http://www.minskportal.com">www.minskportal.com</a>
Настоящее и прошлое Минска: исторические фотографии	<a href="http://www.minsk-old-new.com">www.minsk-old-new.com</a>
Збор помнікаў зямлі беларускай	<a href="http://www.radzima.org">www.radzima.org</a>
Страна замков	<a href="http://www.zamak.by">www.zamak.by</a>
Исторический проект «Мир-Несвиж.com»	<a href="http://www.mir-nesvizh.com">www.mir-nesvizh.com</a>
Музеи Беларуси	<a href="http://museum.by">http://museum.by</a>
Беларускі дзяржаўны музей народнай архітэктуры і быту	<a href="http://etna.of.by">http://etna.of.by</a>
Информационно-справочный портал Открытый контакт	<a href="http://www.open.by">www.open.by</a>
Беларусь — новости	<a href="http://news.belta.by">http://news.belta.by</a>
Белорусская информационная компания	<a href="http://belapan.by">http://belapan.by</a>

Про белорусский интернет	<a href="http://www.probelnet.com">www.probelnet.com</a>
Гидрометцентр Республики Беларусь	<a href="http://www.pogoda.by">www.pogoda.by</a>
Национальный институт образования	<a href="http://www.adu.by">www.adu.by</a>
Образование и обучение в Беларуси	<a href="http://www.obrazovanie.by">www.obrazovanie.by</a>
Централизованное тестирование — on line	<a href="http://www.testirovanie.org">www.testirovanie.org</a>
Детский правовой сайт	<a href="http://mir.pravo.by">http://mir.pravo.by</a>
Белорусский государственный университет	<a href="http://www.bsu.by">www.bsu.by</a>
Белорусский национальный технический университет	<a href="http://www.bntu.by">www.bntu.by</a>
Белорусский государственный экономический университет	<a href="http://www.bseu.by">www.bseu.by</a>
Белорусский государственный университет информатики и радиоэлектроники	<a href="http://www.bsuir.by">www.bsuir.by</a>
Белорусский государственный педагогический университет имени М. Танка	<a href="http://www.bspu.unibel.by">www.bspu.unibel.by</a>
Белорусский государственный университет культуры и искусств	<a href="http://www.buk.by">www.buk.by</a>
Брестский государственный университет	<a href="http://www.brsu.brest.by">www.brsu.brest.by</a>
Витебский государственный университет	<a href="http://www.vsu.by">www.vsu.by</a>
Гомельский государственный университет	<a href="http://www.gsu.by">www.gsu.by</a>
Гродненский государственный университет	<a href="http://www.grsu.by">www.grsu.by</a>
Могилевский государственный университет	<a href="http://msu.mogilev.by">http://msu.mogilev.by</a>

Мозырский государственный педагогический университет имени И. П. Шамякина	<a href="http://www.mgpu.gomel.by">www.mgpu.gomel.by</a>
Полоцкий государственный университет	<a href="http://www.psu.by">www.psu.by</a>
Белорусский школьный портал	<a href="http://www.school-city.by">www.school-city.by</a>
Беларускія слоўнікі і энцыкляпедыі	<a href="http://slounik.org">http://slounik.org</a>
Словари и справочники	<a href="http://o-db.ru/ru/dictionaries.html">http://o-db.ru/ru/dictionaries.html</a>
Педагагічная майстэрня	<a href="http://nastaunik.info">http://nastaunik.info</a>
Беларуская літаратура	<a href="http://www.bellit.narod.ru">www.bellit.narod.ru</a>
Электронная библиотека «Беларуская палічка»	<a href="http://www.knihicom">www.knihicom</a>
Поиск бывших одноклассников, однокурсников, учителей	<a href="http://www.parta.by">www.parta.by</a>
Музыкальный портал	<a href="http://www.music.oko.by">www.music.oko.by</a>
Архив нот и табулатур для гитары	<a href="http://www.notesgallery.com">www.notesgallery.com</a>
Театральная Беларусь	<a href="http://www.belarus-theatre.net">www.belarus-theatre.net</a>
Белорусские актеры театра и кино	<a href="http://belactors.info">http://belactors.info</a>
Белорусский республиканский театр юного зрителя	<a href="http://beltuz.by">http://beltuz.by</a>
Белорусская ассоциация «Конкурс»	<a href="http://www.bakonkurs.org">www.bakonkurs.org</a>
Школа фотографии	<a href="http://sp.znyata.com">http://sp.znyata.com</a>
Сайт, посвященный различным видам рукоделия	<a href="http://www.businka.by">www.businka.by</a>
Вязание и вышивание	<a href="http://www.beloshveika.by">www.beloshveika.by</a>
Кулинарный портал	<a href="http://www.vkus.by">www.vkus.by</a>
Детям об армии	<a href="http://www.deti.mil.by">www.deti.mil.by</a>

## ПРИЛОЖЕНИЕ 3

<b>Каталоги и коллекции электронных образовательных ресурсов</b>	
Единое окно доступа к образовательным ресурсам	<a href="http://window.edu.ru">http://window.edu.ru</a>
Каталог Российского общебазовательного портала	<a href="http://www.school.edu.ru">www.school.edu.ru</a>
Каталог «Образовательные ресурсы Интернета»	<a href="http://catalog.iot.ru">http://catalog.iot.ru</a>
Центр информационных образовательных ресурсов РФ	<a href="http://fcior.edu.ru">http://fcior.edu.ru</a>
<b>Образовательные ресурсы по предметам</b>	
Единая коллекция цифровых образовательных ресурсов	<a href="http://school-collection.edu.ru">http://school-collection.edu.ru</a>
Школьный портал	<a href="http://www.portalschool.ru">www.portalschool.ru</a>
Портал информационной поддержки ЕГЭ	<a href="http://ege.edu.ru">http://ege.edu.ru</a>
<b>Информатика</b>	
Портал «Информационные и коммуникационные технологии»	<a href="http://www.ict.edu.ru">www.ict.edu.ru</a>
Газета «Информатика»	<a href="http://inf.1september.ru">http://inf.1september.ru</a>
Журнал «Информатика и образование»	<a href="http://www.infojournal.ru">www.infojournal.ru</a>
Интернет-университет информационных технологий (ИНТУИТ)	<a href="http://www.intuit.ru">www.intuit.ru</a>
Виртуальный компьютерный музей	<a href="http://www.computer-museum.ru">www.computer-museum.ru</a>
Библиотека алгоритмов Alglib	<a href="http://alglib.sources.ru">http://alglib.sources.ru</a>
Алгоритмы и методы AlgoList	<a href="http://algolist.manual.ru">http://algolist.manual.ru</a>
Компьютер на уроках	<a href="http://www.klyaksa.net">www.klyaksa.net</a>

Задачи по информатике	<a href="http://www.problems.ru/inf">www.problems.ru/inf</a>
Онлайн-тестирование и сертификация по информационным технологиям	<a href="http://test.specialist.ru">http://test.specialist.ru</a>
Олимпиадная информатика	<a href="http://www.olypiads.ru">www.olypiads.ru</a>
Всероссийская командная олимпиада школьников по программированию	<a href="http://neerc.ifmo.ru/school/">http://neerc.ifmo.ru/school/</a>
Всероссийская олимпиада школьников по информатике	<a href="http://info.rusolymp.ru">http://info.rusolymp.ru</a>
Всероссийская интернет-олимпиада школьников по информатике	<a href="http://olymp.ifmo.ru">http://olymp.ifmo.ru</a>
Задачи соревнований по спортивному программированию	<a href="http://acm.timus.ru">http://acm.timus.ru</a>
Олимпиады по информатике в Беларуси	<a href="http://byoi.narod.ru">http://byoi.narod.ru</a>

**Физика**

Газета «Физика»	<a href="http://fiz.1september.ru">http://fiz.1september.ru</a>
Журнал «Квант»	<a href="http://www.kvant.info">www.kvant.info</a>
Астрофизический портал	<a href="http://www.afportal.ru">www.afportal.ru</a>
Лауреаты Нобелевской премии по физике	<a href="http://n-t.ru/nl/fz/">http://n-t.ru/nl/fz/</a>
Мир физики: демонстрации физических экспериментов	<a href="http://demo.home.nov.ru">http://demo.home.nov.ru</a>
Физика.ру: сайт для учащихся и преподавателей	<a href="http://www.fizika.ru">www.fizika.ru</a>
Школьная физика для учителей и учеников	<a href="http://www.alsak.ru">www.alsak.ru</a>
Виртуальный методический кабинет физики и астрономии	<a href="http://www.gomulina.orc.ru">www.gomulina.orc.ru</a>

Занимательная физика в вопросах и ответах	<a href="http://elkin52.narod.ru">http://elkin52.narod.ru</a>
Физика для всех: задачи по физике с решениями	<a href="http://fizzika.narod.ru">http://fizzika.narod.ru</a>
Обучающие трехуровневые тесты по физике	<a href="http://www.physics-regelman.com">www.physics-regelman.com</a>
Самотестирование школьников 7—11-х классов	<a href="http://barsic.spbu.ru/www/tests/">http://barsic.spbu.ru/www/tests/</a>
Всероссийская олимпиада школьников по физике	<a href="http://phys.rusolymp.ru">http://phys.rusolymp.ru</a>
Открытые интернет-олимпиады по физике	<a href="http://barsic.spbu.ru/olymp">http://barsic.spbu.ru/olymp</a>
<b>Математика</b>	
Портал Math.ru	<a href="http://www.math.ru">www.math.ru</a>
Вся элементарная математика: математическая интернет-школа	<a href="http://www.bymath.net">www.bymath.net</a>
Газета «Математика»	<a href="http://mat.1september.ru">http://mat.1september.ru</a>
Задачи по геометрии: информационно-поисковая система	<a href="http://zadachi.mccme.ru">http://zadachi.mccme.ru</a>
Задачи по математике	<a href="http://www.problems.ru">www.problems.ru</a>
ЕГЭ по математике: подготовка к тестированию	<a href="http://www.uztest.ru">www.uztest.ru</a>
Образовательный математический сайт Exponenta	<a href="http://www.exponenta.ru">www.exponenta.ru</a>
Портал «Вся математика в одном месте»	<a href="http://www.allmath.ru">www.allmath.ru</a>
Детская математика	<a href="http://www.kidmath.ru">www.kidmath.ru</a>
Занимательная математика: олимпиады, конкурсы, игры	<a href="http://www.math-on-line.com">www.math-on-line.com</a>
Математические олимпиады для школьников	<a href="http://www.olimpiada.ru">www.olimpiada.ru</a>

# Содержание

От авторов . . . . .	3
<b>Г л а в а 1. Представление информации в компьютере</b>	
§ 1. Кодирование информации . . . . .	4
§ 2. Единицы измерения объема информации . . . . .	6
§ 3. Понятие системы счисления. Двоичная система счисления . . . . .	8
§ 4. Представление различных видов информации . . . . .	14
4.1. Кодирование текстовой информации . . . . .	—
4.2. Кодирование графической информации . . . . .	16
4.3. Кодирование звуковой информации . . . . .	18
<b>Г л а в а 2. Основы алгоритмизации и программирования</b>	
§ 5. Структурированный тип данных: массив . . . . .	21
§ 6. Ввод и вывод элементов массива . . . . .	25
§ 7. Арифметические действия над элементами массива	32
§ 8. Преобразование элементов массива . . . . .	38
§ 9. Поиск элементов с заданными свойствами . . . . .	45
§ 10. Решение задач с использованием массивов . . . . .	56
<b>Г л а в а 3. Основы анимации</b>	
§ 11. Основные понятия. Виды анимации . . . . .	67
§ 12. Работа в редакторе Flash . . . . .	71
12.1. Основные элементы интерфейса . . . . .	—
12.2. Рисование в редакторе Flash . . . . .	74
§ 13. Преобразование объектов . . . . .	80
13.1. Преобразование формы . . . . .	—
13.2. Преобразование заливки . . . . .	83
§ 14. Использование слоев и библиотеки объектов . . . . .	87
14.1. Создание и использование слоев . . . . .	—
14.2. Создание и использование библиотечных образцов . . . . .	90
§ 15. Основы Flash-анимации . . . . .	96
15.1. Основные понятия . . . . .	—
15.2. Покадровая анимация . . . . .	98
§ 16. Анимация движения . . . . .	102
§ 17. Движение нескольких объектов. Звуковое сопровождение . . . . .	109
§ 18. Движение по траектории . . . . .	114
§ 19. Использование маски . . . . .	117

§ 20. Анимация формы . . . . .	122
§ 21. Работа с текстом . . . . .	128
21.1. Ввод и форматирование . . . . .	—
21.2. Анимация текста . . . . .	131
§ 22. Создание и использование клипов . . . . .	134
<b>Г л а в а 4. Информационные ресурсы сети Интернет</b>	
§ 23. Подключение к сети Интернет . . . . .	140
§ 24. Организация службы WWW . . . . .	145
§ 25. Поиск информации в сети Интернет . . . . .	151
§ 26. Сохранение информации из сети Интернет . . . . .	160
§ 27. Национальные информационные ресурсы . . . . .	164
§ 28. Образовательные ресурсы сети Интернет . . . . .	175
Приложения . . . . .	182

---

(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 /			

Учебное издание  
Заборовский Георгий Александрович  
Лапо Анжелика Ивановна  
Пупцов Александр Евгеньевич

## ИНФОРМАТИКА

Учебное пособие для 9 класса  
общеобразовательных учреждений  
с русским языком обучения

Зав. редакцией В. Г. Бехтина. Редактор Н. М. Алганова. Оформление Ю. В. Прохорчика. Художественный редактор Л. А. Дашкевич. Технический редактор М. И. Чепловодская. Компьютерная верстка Е. Ю. Гурченок.

Корректоры Д. Р. Лосик, В. С. Бабеня, А. В. Аleshko.

Подписано в печать 19.10.2009. Формат 60 × 90 <sup>1</sup>/<sub>16</sub>. Бумага офсетная. Гарнитура школьная. Офсетная печать. Усл.-печ. л. 12. Уч.-изд. л. 7,6. Тираж 96 300 экз. Заказ 113.

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.

Правообладатель Народная асвета